

Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Клиент-серверное приложение для анализа
системных журналов операционной
системы Linux

Студент

(подпись, дата)

Можаев Кирилл Олегович
(ФИО полностью)

Руководитель

(подпись, дата)

Афонин А.Ю.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ Козлов А.Ю.
«__» _____ 2023 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

1. Студент Можаев Кирилл Олегович гр. 19ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Клиент-серверное приложение для анализа системных журналов
операционной системы Linux

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 2019 г.

2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Афонин А.Ю.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение: автоматизированная информационная система предназначена для
для анализа системных журналов*

2) *Среда выполнения: операционная система Linux*

*Средства разработки: языки программирования Java Script, C++, Java, Фреймворки
React, Spring Boot, язык разметки HTML и язык стилей CSS; инструментальные средства
«IntelliJ Idea», «Postgresql».*

3) *Основные функции автоматизированной информационной системы:*

а) Добавление и удаление правил

б) Модификация существующих правил в реальном времени.

в) Просмотр зарегистрированных срабатываний правил .

г) Просмотр работы системы в реальном времени

д) Просмотр данных подключенных станций.

е) Авторизация.

ж) Поиск и фильтрация зарегистрированных срабатываний правил .

4. Перечень подлежащих разработке вопросов

1) Анализ предметной области

2) Анализ требований

3) Проектирование

4) Реализация

5) Тестирование

6) План разработки и оценка бюджета

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований	10	06.06.23-07.06.23	
2	Проектирование	25	08.06.23-11.06.23	
3	Реализация	35	12.06.23-14.06.23	
4	Контроль качества	15	15.06.23-16.06.23	
5	План разработки бюджета	5	17.06.23-18.06.23	
6	Оформление пояснительной записки	10	19.06.23-20.06.23	

Дата выдачи задания " ____ " _____ 2023 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 95 листов, 30 рисунков, 9 таблиц, 16 использованных источников, 3 приложения.

SPRING BOOT, JSON, JAVA, JAVASCRIPT, БАЗА ДАННЫХ, АНАЛИЗ ЖУРНАЛОВ, LOGS, UML, POSTGRESQL, СУБД, WEB-ПРИЛОЖЕНИЕ, ИНФОРМАЦИОННАЯ СИСТЕМА

Объектом разработки является программное обеспечение для анализа системных журналов в ОС Linux

Цель выпускной квалификационной работы – разработать программное обеспечение, позволяющее проводить анализ журналов в ОС Linux

Результат работы – программное обеспечение «Анализ системных журналов в ОС Linux», отвечающее всем требованиям технического задания.

Для разработки программного продукта применялись языки Java Script, JAVA, HTML, CSS, IDE «IntelliJ Idea», система управления базами данных POSTGRESQL и операционная система Linux.

					ПГУ 09.03.02 – 8БР19118.09 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Можасев К.О.			Клиент-серверное приложение для анализа системных журналов операционной системы Linux.	Лит.	Лист	Листов
Провер.		Афонин А.Ю.					4	95
Реценз.						группа 19ВИ1		
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований.....	8
1.1 Анализ предметной области.....	8
1.2 Анализ программ аналогов.....	9
1.2.1 Анализ «Logstash».....	9
1.2.2 Анализ «Graylog»	11
1.2.3 Анализ «Fluentd»	12
1.2.4 Сравнение разрабатываемого приложения с аналогами.....	13
1.3 Постановка задачи	15
1.3.1 Назначение.....	15
1.3.2 Предполагаемая аудитория	15
1.3.3 Общий взгляд на продукт.....	16
1.3.4 Выбор инструментальных средств.....	16
1.4 Постановка задачи на разработку	18
1.5 Анализ требований на разработку	20
1.5.1 Требования к функциональности программных средств	20
1.5.2 Требования к пользовательскому интерфейсу.....	21
2 Проектирование программного обеспечения	22
2.1 Базовая архитектура	22
2.2 Логическая диаграмма	23
2.2.1 Диаграмма деятельности	24
2.2.2 Диаграмма последовательности	25

2.3	Пользовательский интерфейс клиентского приложения	27
2.4	Планирование разработки и оценка бюджета	29
3	Реализация программного обеспечения	34
3.1	Физическая модель базы данных	34
3.2	Серверная часть приложения	35
3.3	Диаграмма развертывания	36
4	Контроль качества программного обеспечения	39
4.1	Функциональное тестирование	39
4.2	Определение классов эквивалентности.....	39
4.3	Анализ граничных условий	40
4.4	Построение функциональной диаграммы.....	43
4.5	Парное тестирование.....	44
4.6	Метрики программного кода проекта	45
	Заключение	48
	Список использованных источников	49
	Приложение А Листинг клиентского приложения.....	51
	Приложение Б Листинг серверного приложения	71
	Приложение В Результаты тестирования	87

Введение

Анализ системных журналов в операционной системе Linux является важным инструментом для контроля и мониторинга компьютерной системы. Системные журналы содержат подробную информацию обо всех событиях, происходящих в системе, и могут помочь выявлять ошибки, слабые места и другие проблемы в работе системы.

Однако, в процессе работы с системными журналами возникает ряд сложностей, связанных с необходимостью анализа большого объема информации, нахождением конкретных событий, их интерпретацией и выделением ключевых показателей. В связи с этим, для решения указанных задач могут быть использованы специальные инструменты и технологии, которые позволяют автоматизировать процесс анализа журналов и упростить его для пользователя.

Кроме того, разработка и использование таких инструментов является актуальной темой для системных администраторов и пользователей Linux, которые хотят убедиться в работоспособности своей системы или выявить возможные проблемы.

Целью данной работы является исследование существующих инструментов и технологий для анализа системных журналов в Linux, а также разработка и тестирование собственной программной системы, основанной на современных подходах и инструментах, для упрощения анализа журналов в Linux.

В процессе исследования был проведен анализ процесса обработки журнальных записей, а также разработаны алгоритмы и программный код. В результате данной работы будет создана программа, которая позволит проводить более эффективный анализ системных журналов в Linux, выявлять возможные проблемы и ошибки, а также повышать работоспособность системы в целом.

1 Анализ предметной области и требований

1.1 Анализ предметной области

Логи (log) – это специальные текстовые файлы, в которых в хронологическом порядке фиксируется информация обо всех действиях программы или пользователей. Проще говоря, это журнал регистрации всех событий, происходивших в системе:

- ошибки сервера (сбои), возникающие при обращении к некоторым функциям сайта или задачам;
- данные о доступе – запись о подключении (или попытке входа) каждого пользователя, откуда и как он попал на сайт;
- прочие, записывающие информацию о работе компонентов сервера.

Логи доступа указывают на уязвимые места сайта (в случае взлома), помогают собирать статистику посещаемости, узнавать откуда проводились запросы и какие ресурсы ссылаются на этот сайт, оценивать популярность страниц.

По файлам ошибок проще найти источник проблемы и оперативно устранить баги и сбои. Журналы сервера облегчают контроль рабочего процесса серверной машины.

В файлах логов записывается и отслеживается история работы всего программного комплекса.

Поэтому специалисты рекомендуют периодически просматривать их, даже если никаких подозрительных моментов не произошло. И тем более немедленно обратиться к ним, если резко возросло количество ошибок, посыпался спам или заметно увеличилась нагрузка на сервер.

Логи используются для мониторинга работы программ, идентификации ошибок, отладки и анализа производительности. Анализ логов позволяет

выявлять причины ошибок, улучшать производительность приложений и оптимизировать работу системы.

Для анализа логов часто используются специальные программы, называемые «системами лог-анализа».

Они позволяют автоматически обрабатывать большое количество лог-данных, выделять в них важные события (например, ошибки или определенные действия пользователей), и отображать результаты в удобном для аналитика виде. Это может быть в виде отчетов, дашбордов, графиков, алертов и т. д.

Программы для анализа логов могут обеспечить следующие преимущества:

- ускорение поиска и анализа лог-данных;
- автоматическая выявление и оповещение об ошибках;
- анализ производительности и оптимизации приложений;
- упрощение процесса отладки;
- мониторинг и анализ поведения пользователей;

Таким образом, логи являются важной и необходимой частью работы компьютерных систем и приложений, а программы для анализа логов позволяют использовать эту информацию максимально эффективно.

1.2 Анализ программ аналогов

Системы для анализа системных журналов, такие как «Logstash», «Graylog» и «Fluentd» представляют собой клиент-серверное приложение.

1.2.1 Анализ «Logstash»

Logstash представляет собой открытое программное обеспечение, которое служит для сбора, фильтрации и обработки журнальных файлов. Название Logstash состоит из двух слов – «log» и «stash», и означает «склад журналов».

Кроме журнальных файлов, Logstash может также обрабатывать данные различных форматов и приложений.

Logstash обычно используется в составе стека Elastic Stack в качестве системы сбора и анализа данных. Она может интегрироваться с другими приложениями Elastic Stack, такими как Elasticsearch и Kibana, и предоставлять возможность провести визуальный анализ, поиск и мониторинг журнальных файлов, на основе собранных данных.

Основными особенностями Logstash являются:

- Множество входных и выходных плагинов: Logstash предоставляет более 200 плагинов для сбора и обработки данных, в том числе поддержку протоколов, таких как syslog, слежения за файлами и базами данных.
- Гибкие возможности фильтрации: Logstash имеет мощную систему фильтрации, которая позволяет производить специфичную обработку данных для каждого типа записи.
- Система обработки данных в реальном времени: Logstash позволяет обрабатывать и анализировать большие объемы данных в реальном времени.
- Встроенные механизмы обработки ошибок: Logstash имеет встроенные механизмы для отслеживания ошибок и обработки непредвиденных ситуаций.
- Открытый и расширяемый исходный код: Logstash имеет открытый исходный код и позволяет легко расширять его возможности путем создания собственных плагинов или дополнительной настройки имеющихся.

Основными преимуществами Logstash являются:

- Высокая производительность: Logstash была разработана с учетом требований к высокой производительности и обработке больших объемов данных.

- Легкость использования: благодаря наличию готовых плагинов и инструментов, Logstash имеет простой и интуитивно понятный интерфейс.
- Расширяемость: благодаря открытому исходному коду и возможности написания собственных плагинов, Logstash может быть легко расширен для удовлетворения дополнительных требований.
- Поддержка большого количества форматов: благодаря широкому выбору плагинов, Logstash поддерживает большое количество различных форматов данных при сборе и обработке информации.
- Интеграция с Elastic Stack: Logstash может быть легко интегрирована с другими приложениями Elastic Stack, такими как Elasticsearch и Kibana, что позволяет проводить более глубокий анализ и управление собранными данными.

В целом, Logstash является мощным и гибким инструментом для сбора и обработки данных, который может быть использован для различных задач, связанных с обработкой журнальных файлов и анализом данных в Linux.

1.2.2 Анализ «Graylog»

«Graylog – это открытое программное обеспечение для сбора, хранения, поиска и анализа журнальных файлов и других видов данных. Она была разработана с учетом требований к высокой производительности и эффективности обработки больших объемов данных.

Основными преимуществами Graylog являются:

- Высокая производительность: Graylog была разработана с учетом требований к высокой производительности в обработке больших объемов данных.
- Гибкие возможности настройки: Graylog позволяет настраивать и адаптировать систему в соответствии с требованиями и потребностями пользователя.

- Визуализация данных: Graylog позволяет создавать графики и диаграммы для более удобного просмотра и анализа данных.
- Расширяемость: Graylog может быть расширена для удовлетворения дополнительных требований.
- Интеграция с другими приложениями: Graylog может быть легко интегрирована с другими приложениями, в том числе с Elasticsearch, MongoDB и Kafka.
- Отчетность: Graylog позволяет создавать отчеты об активности системы, включая периодическую отчетность и оповещения.
- Надежность и безопасность: Graylog обеспечивает надежность и безопасность в хранении, обработке и передаче данных
- Легкость использования: благодаря простому интерфейсу и обширной документации, Graylog легко использовать даже для тех пользователей, которые не имеют опыта в работе с журнальными файлами.

Graylog представляет собой мощный и гибкий инструмент для обработки, анализа и хранения журнальных файлов и других видов данных. Она может быть использована в различных сферах деятельности, включая системное администрирование, мониторинг приложений, безопасность и многие другие.

1.2.3 Анализ «Fluentd»

Fluentd – это система сбора и пересылки данных или журналов, написанная на языке программирования Ruby. Она была разработана для упрощения процесса централизованного сбора логов (данных о процессах, событиях и ошибках) в больших и сложных серверных инфраструктурах. Fluentd позволяет собирать данные из различных источников, таких как устройства, приложения, базы данных и файловые системы, и направлять их в различные хранилища данных или инструменты анализа.

Основные особенности и преимущества Fluentd:

- Широкий выбор плагинов: Fluentd содержит более 700 плагинов, которые обеспечивают поддержку большого количества форматов данных (JSON, XML, CSV и другие).
- Гибкая настройка и конфигурация: Fluentd имеет мощную систему настройки и конфигурации, которая позволяет создавать цепочки обработки данных, а также конвертировать и фильтровать полученные данные.
- Масштабируемость: Fluentd поддерживает масштабирование для обработки больших объемов данных и их пересылки, что позволяет использовать более мощные вычислительные ресурсы.
- Надежность: Fluentd предоставляет механизмы обработки ошибок и сбоев, а также поддерживает ротацию журналов для предотвращения их переполнения и потери данных.
- Высокая производительность: Fluentd была спроектирована с учетом требований к высокой производительности и скорости обработки данных.
- Легкость интеграции: Fluentd может легко интегрироваться с другими инструментами, такими как Elasticsearch, Kibana и Hadoop, для обработки, анализа и визуализации собранных данных.

В целом, Fluentd является мощной и гибкой системой для сбора, обработки и пересылки данных или журналов, которая обладает многими преимуществами и может быть использована для различных задач в Linux-среде в различных инфраструктурах.

1.2.4 Сравнение разрабатываемого приложения с аналогами

Рассмотрев схожие по функционалу программы, было проведено их сравнение, представленное в таблице 1.

Таблица 1 – Сравнение программ аналогов

Программа	Достоинства	Недостатки
Logstash	Высокая производительность; Гибкая настройка; Нет собственной системы аутентификации.	Подходит только для встраивания в уже существующую систему так как нет собственного интерфейса; Нет собственной системы аутентификации.
Graylog	Высокая производительность; Расширяемость; Возможность интеграции.	Сложность настройки и обучения пользователей; Подходит только для встраивания в уже существующую систему так как нет собственного интерфейса
Fluentd	Высокая производительность; Надежность; Гибкая настройка; Высокая расширяемость при помощи плагинов.	Сложность настройки и обучения пользователей; Нет собственной системы аутентификации.
JournalD	Работа в реальном времени; Собственная система аутентификации; Осуществляет хранение всех данных без использования сторонних приложений	Правила поддерживают только регулярные выражения при определении срабатывания правил; Упрощенная система сбора событий; .

По результатам сравнения программ аналогов можно сделать вывод, что каждая из систем имеет свои сильные и слабые стороны и при выборе приложения для анализа системных журналов требуется проанализировать для каких целей будет использоваться данное программное обеспечение.

1.3 Постановка задачи

1.3.1 Назначение

Программа для анализа журналов должна обеспечить системным администраторам и специалистам по информационной безопасности дополнительные инструменты для управления безопасностью и эффективностью работы системы, а также для быстрого выявления возникших проблем и их решения.

1.3.2 Предполагаемая аудитория

Программа анализа системных журналов может быть полезна для различных аудиторий, включая:

- Администраторы систем: они могут использовать программу для анализа журналов безопасности, проблем производительности и других системных событий для обнаружения ошибок в системе и устранения их.
- Разработчики программного обеспечения: они могут использовать программу для анализа журналов приложения для выявления ошибок и улучшения производительности приложений.
- Специалисты по безопасности: они могут использовать программу для анализа журналов безопасности, чтобы обнаружить аномальное поведение и инциденты безопасности.

1.3.3 Общий взгляд на продукт

Разрабатываемое программное обеспечение состоит из клиентской, серверной части и демона. Клиентская часть предназначена для взаимодействия с пользователем. Здесь пользователь может авторизоваться, изменять правила анализа журналов, анализировать текущую работу станций на определенный промежуток времени или в реальном времени. На серверной части происходит хранение учетных данных пользователей и правила. Также серверная часть отвечает за коммуникацией между станциями на которых работают демоны для анализа журналов. Демон для анализа журналов – это программа которая работает незаметно пользователя и собирает данные о работе системы из системных журналов.

1.3.4 Выбор инструментальных средств

В качестве фреймворка был выбран «Spring Boot Framework». Spring – это один из самых популярных фреймворков для разработки приложений на Java. Многие компании выбирают Spring для разработки своих приложений по следующим причинам:

- Инверсия управления (IoC) и внедрение зависимостей (DI): это основные компоненты Spring, которые позволяют разработчикам сосредоточиться на бизнес-логике приложений, а не на управлении объектами. Spring автоматически инициализирует и управляет объектами, инжектирует зависимости, создает и уничтожает объекты.
- Поддержка аннотаций: Spring поддерживает широкий спектр аннотаций для упрощения создания приложений. Например, аннотация `@Autowired` позволяет автоматически связать объекты, а аннотация `@RequestMapping` позволяет связывать URL-адреса с методами контроллера.

– Модульность: Spring состоит из модульных компонентов, что облегчает использование и переиспользование кода. Это также позволяет разработчикам выбирать только те компоненты, которые им необходимы, и уменьшает зависимость приложения от фреймворка. Для разработки программного обеспечения был выбран текстовый редактор «IntelliJ IDEA». IntelliJ IDEA – это интегрированная среда разработки (IDE), которая является одним из наиболее популярных инструментов для разработки программного обеспечения на Java. Многие разработчики выбирают IntelliJ IDEA по следующим причинам:

– Удобство использования: IntelliJ IDEA предоставляет богатый набор функциональных возможностей и удобный пользовательский интерфейс, который облегчает работу с большим проектом.

– Расширяемость: IntelliJ IDEA имеет множество плагинов, которые позволяют разработчикам расширять функциональные возможности IDE по своему усмотрению. Также IntelliJ IDEA имеет открытый API, который позволяет разработчикам создавать свои собственные плагины.

– Интеграция с другими системами: IntelliJ IDEA легко интегрируется с другими средствами разработки, такими как системы управления версиями (Git, SVN, Mercurial), серверы приложений (Tomcat, GlassFish, WildFly) и многие другие.

– Рефакторинг: IntelliJ IDEA предоставляет мощный набор инструментов рефакторинга кода, который позволяет улучшить структуру кода и уменьшить вероятность ошибок.

– Автоматическое завершение кода и статический анализ: IntelliJ IDEA автоматически завершает код, основываясь на его контексте, что позволяет ускорить процесс написания кода. IntelliJ IDEA также имеет мощные

инструменты статического анализа, которые позволяют обнаруживать потенциальные ошибки в коде до его запуска.

В качестве языков программирования были выбраны Java и JavaScript. Данные языки программирования позволяют использовать современные концепции, технологии и библиотеки.

В качестве языков разметки были выбраны HTML и XSLT. XSLT – язык преобразования XML-документов. Спецификация XSLT входит в состав XSL и является рекомендацией W3C. При применении таблицы стилей XSLT, состоящей из набора шаблонов, к XML-документу образуется конечное дерево, которое может быть сериализовано в виде XML-документа, XHTML-документа, HTML-документа или простого текстового файла.

В качестве языка моделирования был выбран UML. UML – это универсальный язык визуального моделирования систем. UML объединил лучшие современные технические приемы моделирования и разработки программного обеспечения. UML-диаграммы легко воспринимаются и при этом без труда генерируются компьютерами.

Для проектирования UML-диаграмм использовался продукт StarUML.

Логическое и физическое проектирование базы данных автоматизированной информационной системы выполнялось при помощи case-средства Visual Paradigm.

Для планирования и распределения ресурсов по задачам был выбран продукт Ganttpro.

1.4 Постановка задачи на разработку

Основной целью данной работы является разработка программного средства, позволяющего проводить анализ системных журналов в операционной системе Linux. В наше время информационная безопасность является одной из наиболее актуальных и важных проблем в IT-отрасли. Одним из основных

инструментов, позволяющих следить за событиями в компьютерной системе и находить возможные слабые места, являются системные журналы. Анализ этих журналов позволяет выявлять неисправности, ошибки и уязвимости, которые затем могут быть устранены.

В данной работе будет рассмотрен процесс разработки клиент-серверного приложения, которое представляет собой инновационное решение для анализа системных журналов в Linux. Разрабатываемое программное обеспечение состоит из двух частей: клиентской и серверной. Клиентская часть приложения предназначена для визуализации результатов анализа журналов, а серверная часть отвечает за сбор информации из журналов и обработку полученных данных. Кардинальное отличие разрабатываемого приложения от аналогов заключается в том, что оно позволяет работать с логами системы в реальном времени, а не только с предварительно записанными файлами журналов. Приложение будет оснащено механизмами предобработки данных и инструментами визуализации информации, что позволит увеличить точность и эффективность анализа журналов.

В ходе работы будут использованы инструменты и технологии, такие как языки программирования Java, C++ и JavaScript, фреймворки React и Spring Boot, а также база данных PostgreSQL.

Выводы, полученные в ходе работы, позволят оценить возможности разрабатываемого приложения в сфере информационной безопасности, обосновать его эффективность и выбрать наиболее эффективный набор инструментов для анализа системных журналов в Linux.

1.5 Анализ требований на разработку

1.5.1 Требования к функциональности программных средств

Диаграмма вариантов использования для данного типа пользователей представлена на рисунке. 1

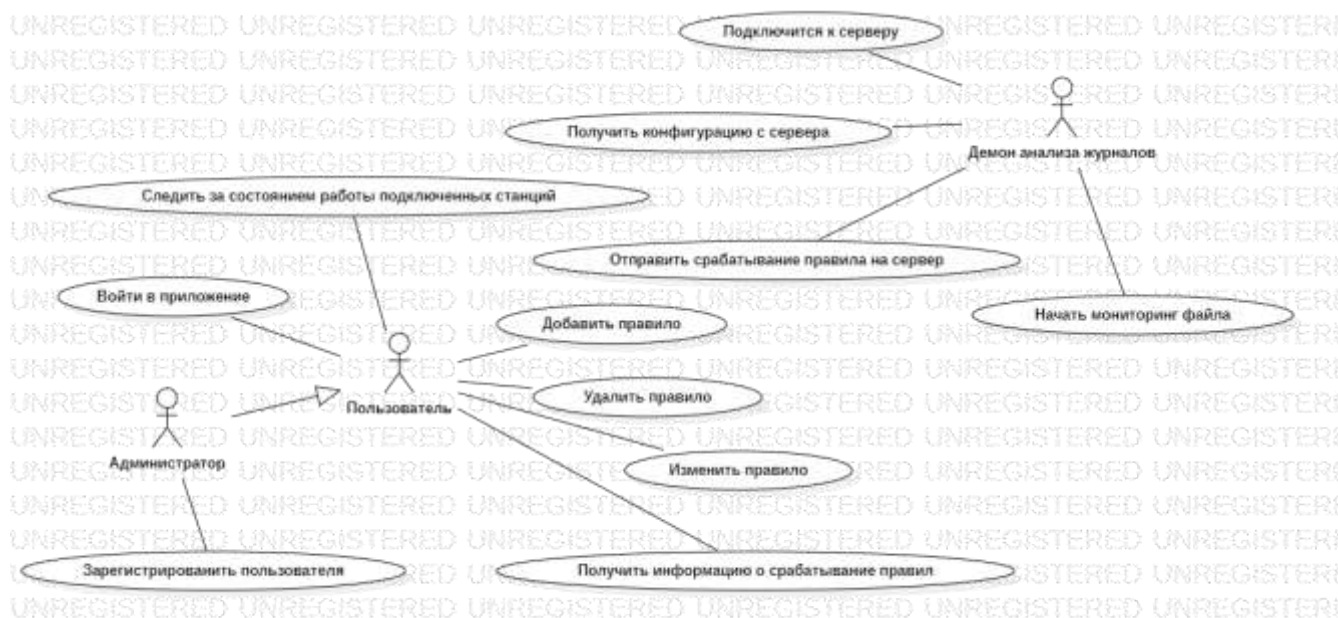


Рисунок 1 – Диаграмма вариантов использования

Сервис анализа журналов должен иметь следующий функционал:

- Добавление, удаление, изменение правила анализа журналов;
- Отображение срабатывания правил в реальном времени;
- Загрузка всех срабатываний правил за определенный период;
- Возможность авторизации пользователя;
- Возможность добавления нового пользователя администратором;

Демона мониторинга файлов должен иметь возможность:

- Осуществлять подключение к серверу;
- При изменении правил перезагружать конфигурацию с сервера;

- При срабатывании правила должен быть отправлен запрос на сервер с информацией об этом правиле;

1.5.2 Требования к пользовательскому интерфейсу

Веб-интерфейс приложения должен быть удобным, интуитивно понятным и простым в использовании, чтобы пользователи могли быстро и легко анализировать журналы и получать необходимую информацию.

Ниже приводятся некоторые основные требования к пользовательскому интерфейсу для приложения лог-анализа:

- Простота использования. Весь процесс анализа должен быть максимально упрощен, чтобы пользователи могли быстро и легко найти нужную информацию. Интерфейс должен быть легким и интуитивно понятным.
- Наглядность представления. Данные должны быть представлены в форме наглядных диаграмм, графиков, таблиц, а также в различных цветовых кодах, а также других визуальных элементов, чтобы пользователи могли быстро и легко интерпретировать информацию.
- Фильтрация данных. Позволяет настраивать условия фильтрации логов, что делает процесс анализа более эффективным и точным. Пользователи могут фильтровать данные по времени, типу событий и другим параметрам.
- Гибкость настроек. Пользователи могут настраивать представление данных и графиков, выбирать и настраивать различные параметры для анализа и отображения, что позволяет делать свой анализ более точным и приспособленным к требованиям.

2 Проектирование программного обеспечения

2.1 Базовая архитектура

Архитектура приложения состоит из нескольких компонентов:

- Фронтенд на React – это компонент, который отвечает за отображение пользовательского интерфейса и взаимодействие с пользователем. Приложение запускается в браузере клиента и общается с бекендом посредством REST API. React обычно построен на основе компонентной архитектуры, где каждый компонент управляет своим состоянием и поведением.
- Бекенд на Spring Boot – это компонент, который отвечает за бизнес-логику приложения и взаимодействие с базой данных. Spring Boot предоставляет удобные инструменты для создания RESTful API, которые могут быть использованы фронтендом React для получения необходимой информации. Взаимодействие с базой данных PostgreSQL происходит через JDBC, а для анализа журналов может использоваться библиотека Log4j.
- База данных PostgreSQL – это сервер базы данных, в которой хранятся данные приложения. База данных взаимодействует с бекендом посредством JDBC и может быть использована для хранения пользовательской информации, настроек и других необходимых данных приложения.
- Демон для анализа журналов на C++ – это кроссплатформенное приложение, которое предназначено для анализа системных журналов и генерации отчетов о возможных проблемах в приложении. Демон может работать на одном из серверов, где запущен бекенд, и реагировать на изменения в журналах с помощью библиотеки Syslog. Отчеты, сгенерированные демоном, могут быть отправлены на почту разработчикам для быстрого определения и устранения возможных проблем.

Таким образом, архитектура приложения является клиент-серверной, где фронтенд на React представляет собой клиентскую часть и бекенд на Spring Boot, база данных PostgreSQL и демон для анализа журналов на C++ представляют серверную часть. Архитектура обеспечивает расширяемость, масштабируемость и надежность приложения.

2.2 Логическая диаграмма

Логическая модель представляет собой модель базы данных, которая не привязана к конкретной СУБД. В ней выделяют основные объекты БД и определяют связи между этими объектами. Иногда определяются типы данных отдельных объектов[16].

Проектирование было проведено с помощью CASE-средства CA Visual Paradigm в нотации IDEF1x.

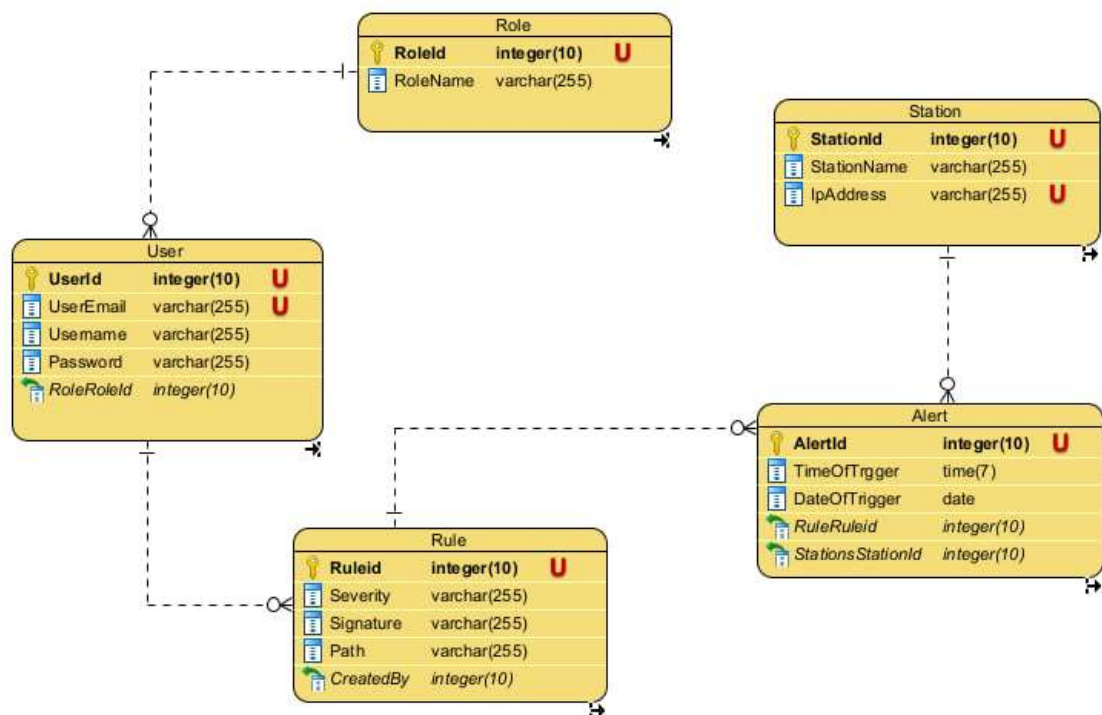


Рисунок 2 – Логическая схема БД

После того, как были выделены сущности и определены отношения между ними, необходимо нормализовать структуру БД.

Спроектированная логическая модель БД не включает в себя неявные сущности, атрибуты, зависящие только от части уникального ключа, и транзитивных зависимостей, следовательно, она соответствует третьей НФ.

2.2.1 Диаграмма деятельности

Для визуализации алгоритма действий при вводе техники в работу была составлена диаграмма деятельности.

Диаграмма деятельности представляет собой граф деятельности, вершинами которого являются состояния действий, а дугами – переходы от одного состояния действия к другому.

Состоянием действия может быть любое действие, записанное на языке программирования, псевдокоде или естественном языке.

Выполнение деятельности начинается с начального состояния и заканчивается конечным.

Переход между состояниями действия происходит последовательно, то есть для начала выполнения следующего действия необходимо завершить текущее.

Это можно обойти, используя разделение и слияние параллельных потоков. В этом случае действия выполняются параллельно друг другу и заканчиваются по завершении всех действий потока.

Помимо этого, имеется механизм ветвления, позволяющий переходить к действию только при выполнении сторожевого условия.

Это помогает, когда необходимо разделить алгоритм на альтернативные ветви. Также есть возможность объединения веток от различных состояний действия в одну.

Диаграмма деятельности представлена на рисунке 3 и в приложении Б.

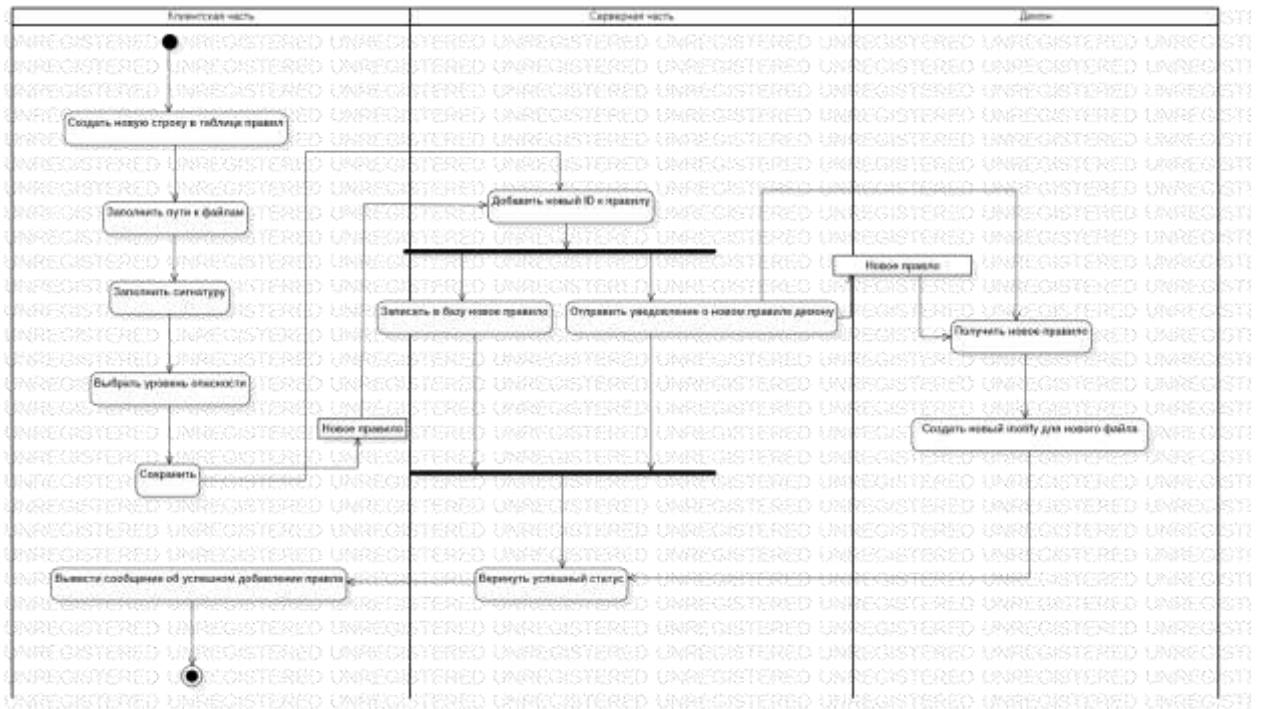


Рисунок 3 – диаграмма деятельности

Для того чтобы создать новое правило необходимо в первую очередь заполнить поле сигнатуры и файлы в которых будет происходить анализ.

После этого из заполненных полей запрос, который отправляется на сервер. На сервере новому правилу присваивается новый уникальный идентификатор после чего производится запись в базу данных.

Сразу же после заполнения правило готово для использования, и готово к началу обработки демона. Получив уведомление о новом правиле, демон загружает это правило и создает новый inotify. После его создания демон возвращается в режим ожидания событий

2.2.2 Диаграмма последовательности

Диаграмма последовательности предназначена для более детального описания вариантов использования. Она позволяет отобразить взаимодействие между объектами, взаимодействующими между собой с помощью сообщений.

Объекты представляют собой экземпляры классов, сущности или пользователей. Сообщения инициируют выполнение какого-либо действия или возвращают его результат.

Отдельные элементы диаграммы можно выделить с помощью фрейма. Например, с помощью фрейма можно задать условия для выполнения отдельных фрагментов или же обозначить их цикличное выполнение.

Также важной особенностью диаграммы является отображение течения времени.

Поэтому важно не только наличие связей между объектами, но и их положения относительно друг друга [12].

В качестве примера была разработана диаграмма последовательности для добавления нового правила. Диаграмма представлена на рисунке 4.

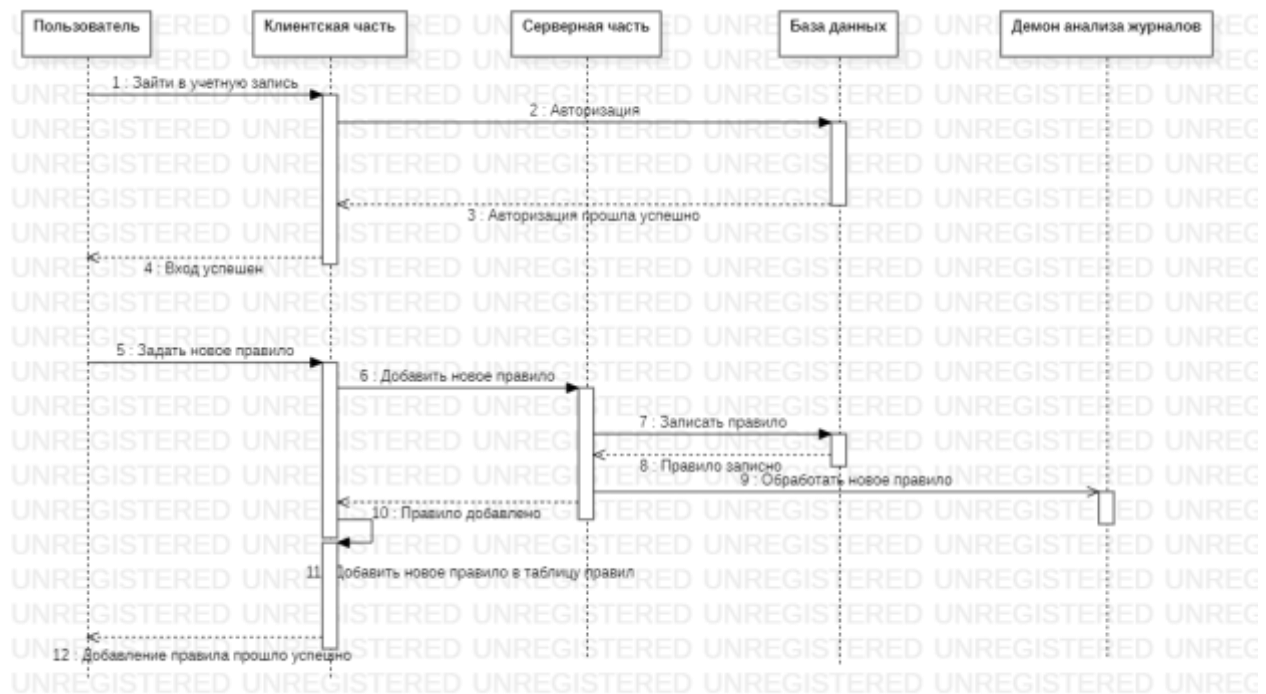


Рисунок 4 – диаграмма последовательности

Для того что бы взаимодействовать с приложением необходимо войти или зарегистрироваться, после чего пользователю будет отправлен jwt токен с

помощью которого он может использовать приложение. При добавлении нового правила запрос отправляется на серверную часть.

На сервере новому правилу присваивается новый уникальный идентификатор после чего производится запись в базу данных. Сразу же после заполнения правило готово для использования, и готово к началу обработки демона.

Получив уведомление о новом правиле, демон загружает это правило и создает новый inotify. После его создания демон возвращается в режим ожидания событий.

2.3 Пользовательский интерфейс клиентского приложения

The image shows a registration form with the following elements and numbered callouts:

- 1**: Points to the 'Имя *' (Name) input field containing 'Александр'.
- 2**: Points to the 'Фамилия *' (Surname) input field containing 'Морозов'.
- 3**: Points to the 'Почта *' (Email) input field containing 'alexandr.v.mzv@gmail.com'.
- 4**: Points to the 'Пароль *' (Password) input field, which is masked with dots.
- 5**: Points to the 'ЗАРЕГИСТРИРОВАТЬСЯ' (REGISTER) button.
- 6**: Points to the text 'Уже есть аккаунт? Войти' (Already have an account? Log in).

Рисунок 5 – Страница регистрации

Основные элементы интерфейса, изображенные на рисунке 5:

- 1 – имя пользователя;
- 2 – фамилия пользователя;
- 3 – e-mail пользователя;
- 4 – пароль пользователя;
- 5 –кнопка регистрации;
- 6 – кнопка входа.

Страница приложения для авторизованного пользователя представлена на рисунке 6.

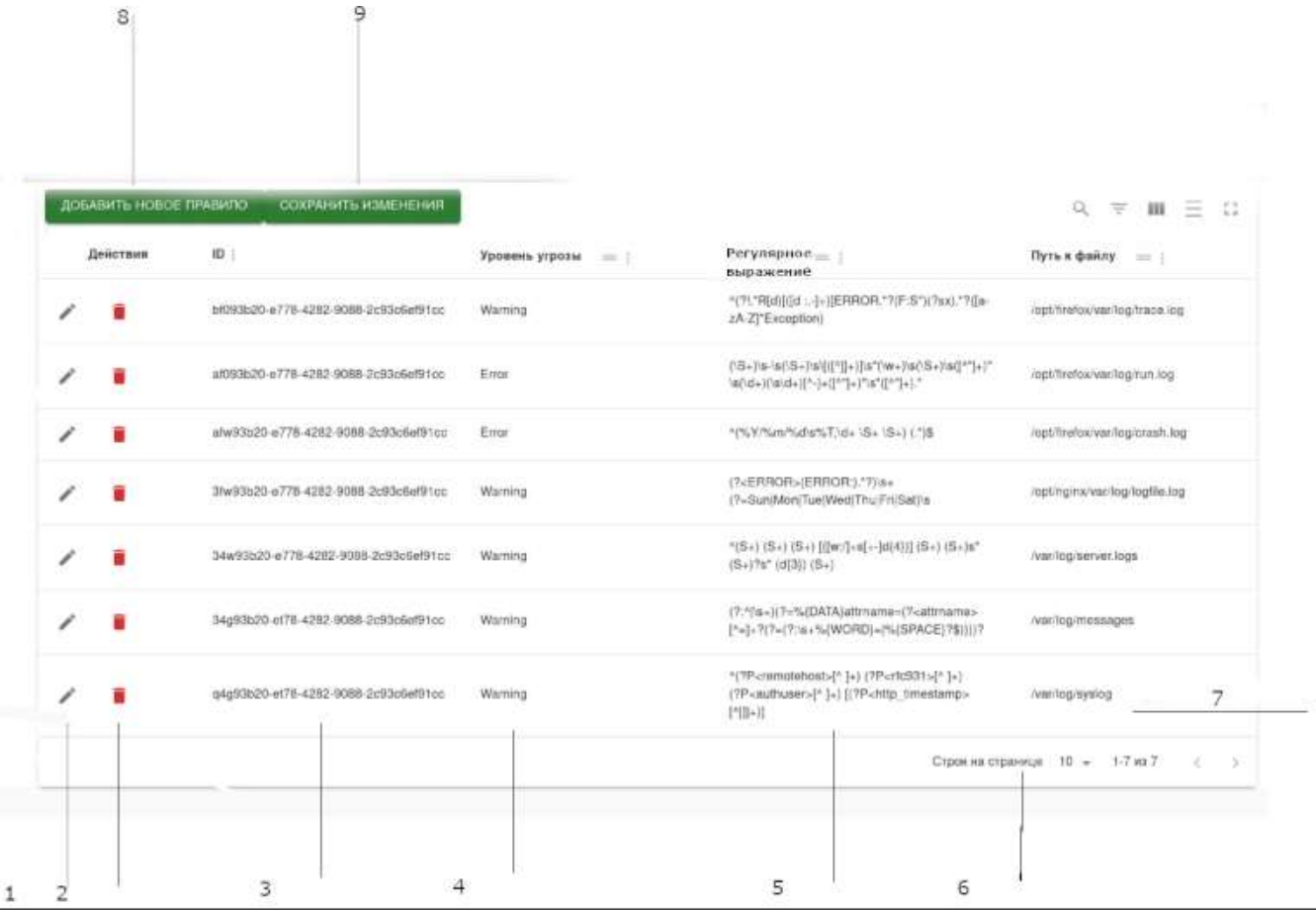


Рисунок 6 – Страница создания правил

На странице, показанной на рисунке 6 мы можем видеть таблицу для заполнения правил, каждое добавленное правило будет отправлено на рабочие станции при нажатии кнопки «Сохранить изменения». Подробно все элементы перечислены ниже:

- 1 – изменить правило
- 2 – удалить правило
- 3 – id правила
- 4 – уровень угрозы, используемый при срабатывании правила
- 5 – регулярное выражение которые используется для работы правила
- 6 – количество правил на странице
- 7 – путь к системному журналу, в котором будет проводится анализ
- 8 – кнопка добавить новое правило
- 9 – кнопка сохранить изменения в правилах

2.4 Планирование разработки и оценка бюджета

Планирование бюджета для разработки программного обеспечения включает определение необходимых финансовых и временных ресурсов, необходимых для завершения проекта в заданные сроки и бюджет.

Эффективное распределение ресурсов позволяет уменьшить риски и избежать проблем, связанных с дефицитом каких-либо ресурсов.

Для планирования разработки и расчета бюджета можно использовать диаграмму Ганта.

Она позволяет визуализировать временные рамки задач и мероприятий, необходимых для завершения проекта, и облегчает контроль за выполнением задач внутри проекта

Ключевым фактором в организации эффективного управления проектом является наличие заранее определенного плана и минимизация отклонений от него в процессе управления.

В ходе планирования проекта было выделено 10 задач. Список задач проекта, их виды, длительность и предшественники приведены в таблице 2.

Таблица 2 – Задачи проекта

№	Название	Назначено	Длительность	Предшественник
1	Начало проекта			
2	Анализ предметной области	Аналитик	16	1
3	Обзор аналогичных программных средств	Аналитик	16	2.1
4	Подстановка задачи на разработку	Аналитик	16	2.2
5	Анализ требований на разработку	Аналитик	8	2.3
6	Проектирование интерфейса пользователя	Архитектор	20	2.4
7	Проектирование авторизации	Архитектор	20	3.1
8	Реализация клиентской части	Программист	20	3.1
9	Проектирование конфигурации правил	Архитектор	20	3.2
10	Реализация авторизации	Программист	20	4.1, 3.2
11	Проектирование демона мониторинга	Архитектор	20	3.3
12	Реализация конфигурации правил	Программист	20	4.2, 3.3
13	Реализация демона мониторинга файлов	Программист	20	4.3, 3.4
14	Функциональное тестирование	Тестировщик	40	4.4
15	Завершение проекта			5.2
16	Оформление документации	Архитектор	20	6
17	Конец проекта			7

Создадим проект и занесем выше перечисленные задачи в созданный проект.

Выделим необходимые ресурсы для выполнения каждой из задач данного проекта. Ресурсы и затраты представлены на рисунке 7.

☐	Ресурс	Стоимость
☐	<div><div>А</div>Аналитик</div>	в час ▼ 300
☐	<div><div>П</div>Программист</div>	в час ▼ 800
☐	<div><div>Т</div>Тестировщик</div>	в час ▼ 700
☐	<div><div>А</div>Архитектор</div>	в час ▼ 1200

Рисунок 7 – Распределение ресурсов по задачам представлено в таблице 2.

Таблица 3 – Задачи и затраты проекта

Задача	Исполнитель	Стоимость
Функциональное тестирование	Тестировщик	56000
Реализация конфигурации правил	Программист	16000
Реализация клиентской части	Программист	16000
Реализация демона мониторинга файлов	Программист	16000
Реализация авторизации	Программист	16000
Проектирование интерфейса пользователя	Архитектор	24000
Проектирование конфигурации правил	Архитектор	24000
Проектирование демона мониторинга	Архитектор	24000
Проектирование авторизации	Архитектор	24000

Продолжение таблицы 3

Задача	Исполнитель	Стоимость
Подстановка задачи на разработку	Аналитик	4800
Оформление документации	Архитектор	24000
Обзор аналогичных программных средств	Аналитик	4800
Анализ требований на разработку	Аналитик	2400
Анализ предметной области	Аналитик	4800
Функциональное тестирование	Тестировщик	56000
Реализация конфигурации правил	Программист	16000

Далее закрепим за задачами соответствующие ресурсы. После, программа предложит наглядную иллюстрацию календарного плана в виде диаграммы Ганта. Полученная диаграмма Ганта приведена на рисунке 8.

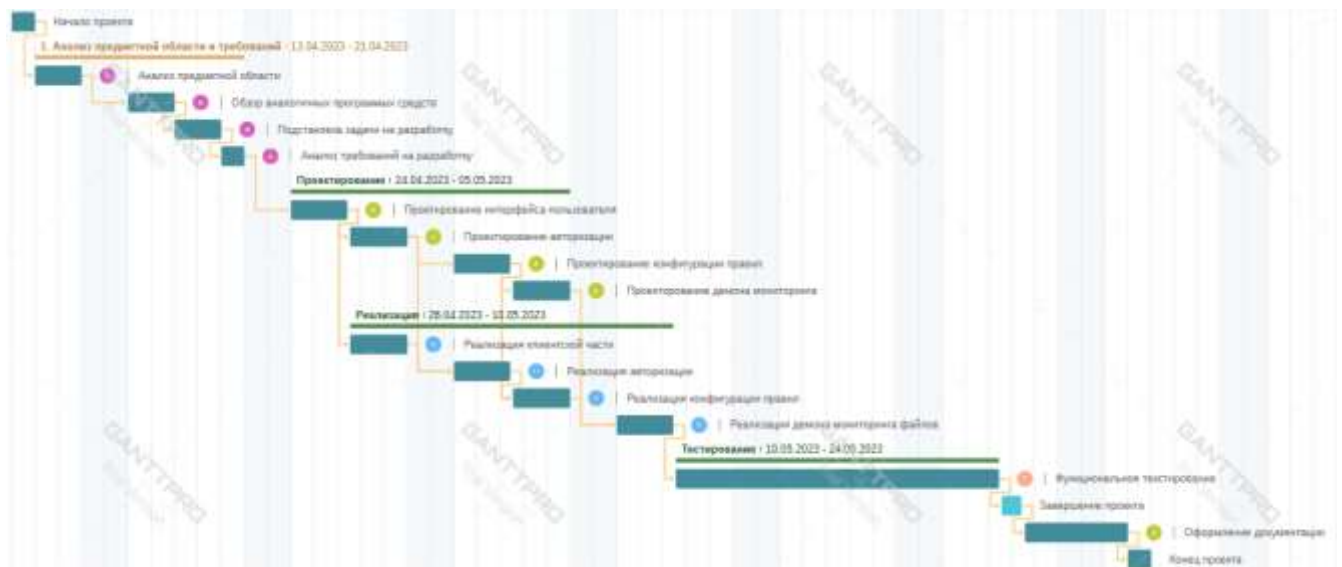


Рисунок 8 – Диаграмма Ганта

На рисунке 8 видно, что проект разделен на 5 фаз:

- «Анализ предметной области и требований»;
- «Проектирование»;
- «Реализация»;
- «Тестирование».

Также на рисунке 8 представлены две вехи: «Начало проекта» и «Завершение проекта».

Длительность выполнения проекта составила 59 дней, а его бюджет составил 256 800 рублей.

3 Реализация программного обеспечения

3.1 Физическая модель базы данных

Физическая модель строится на основе логической модели данных и учитывает особенности целевой СУБД. В качестве СУБД был выбран продукт PostgreSQL.

При помощи case-средства Visual Paradigm была разработана физическая модель, представленная на рисунке 9.

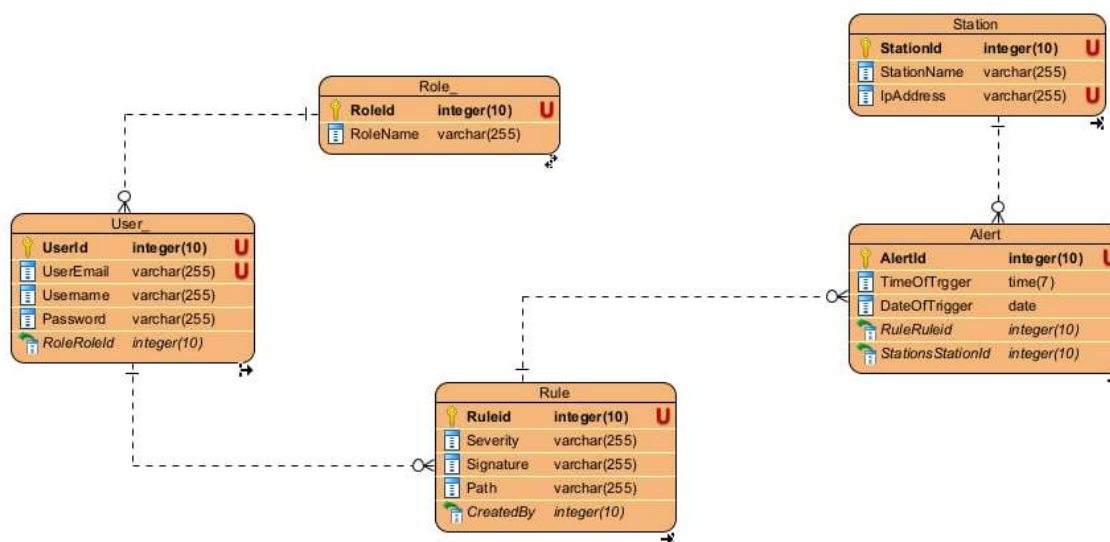


Рисунок 9 – Физическая модель базы данных

Сгенерированный скрипт создания базы данных в Visual Paradigm был выполнен в СУБД PostgreSQL.

3.2 Серверная часть приложения

Диаграмма компонентов описывает физическое представление системы, ее архитектуру, а также устанавливает зависимости между программными компонентами. Данная диаграмма для серверной части представлена на рисунке 10.

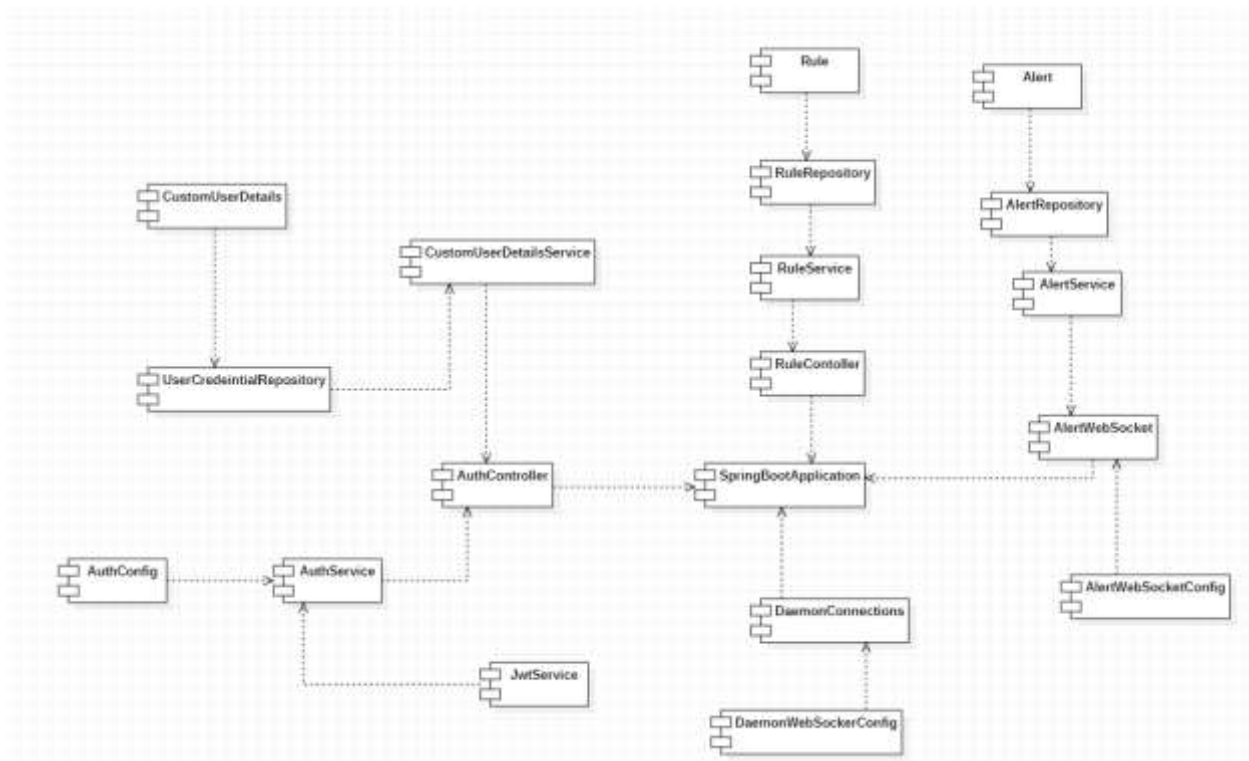


Рисунок 10 – Диаграмма компонентов

На диаграмме изображены три основных компонента: **UserCredentialDetails**, **Rules** и **Alerts**, каждый из которых имеет отдельный класс **Repository** и **Service**.

UserCredentialDetails – компонент, который представляет информацию о пользовательских учетных данных. У него есть отдельный класс **UserCredentialRepository**, который отвечает за хранение и доступ к данным и **UserCredentialService**, который обеспечивает доступ и манипуляции данными.

Rules – компонент, который состоит из правил анализа журналов. У него есть отдельный класс RulesRepository, который отвечает за хранение и доступ к правилам и RulesService, который обеспечивает доступ и манипуляции правилами.

Alerts – компонент, который представляет алерты, сгенерированные при срабатывании правил анализа журналов. У него есть отдельный класс AlertRepository, который отвечает за хранение и доступ к данным по алертам и AlertService, который обеспечивает доступ и манипуляции алертами.

Каждый из компонентов имеет свои отдельные классы Repository и Service для управления данными и бизнес-логикой, что позволяет легко сопровождать и модифицировать систему в дальнейшем.

3.3 Диаграмма развертывания

Диаграмма развертывания представляет собой схематическое изображение компонентов, интерфейсов и связей между ними, которые необходимы для развертывания и функционирования приложения в среде выполнения.

На диаграмме развертывания изображены следующие компоненты и связи между ними:

- Клиентский интерфейс на React – приложение, которое загружается в браузер и предоставляет пользовательский интерфейс для взаимодействия с приложением.
- Бекенд на Spring – приложение, которое предоставляет серверную логику и API для клиентского интерфейса. Взаимодействует с базой данных PostgreSQL и демоном сборки логов.
- База данных PostgreSQL – сервер баз данных, в котором хранятся данные приложения.

– Демон для сборки логов – отдельное приложение, которое отвечает за сбор и хранение логов приложения.

Связь между компонентами осуществляется посредством протоколов и интерфейсов. Клиентский интерфейс взаимодействует с бекендом на Spring через REST API. Бекенд на Spring взаимодействует с базой данных PostgreSQL посредством JDBC. Демон для сборки логов получает события логов из приложения через WebSocket.

Диаграмма развертывания приложения представлена на рисунке 11

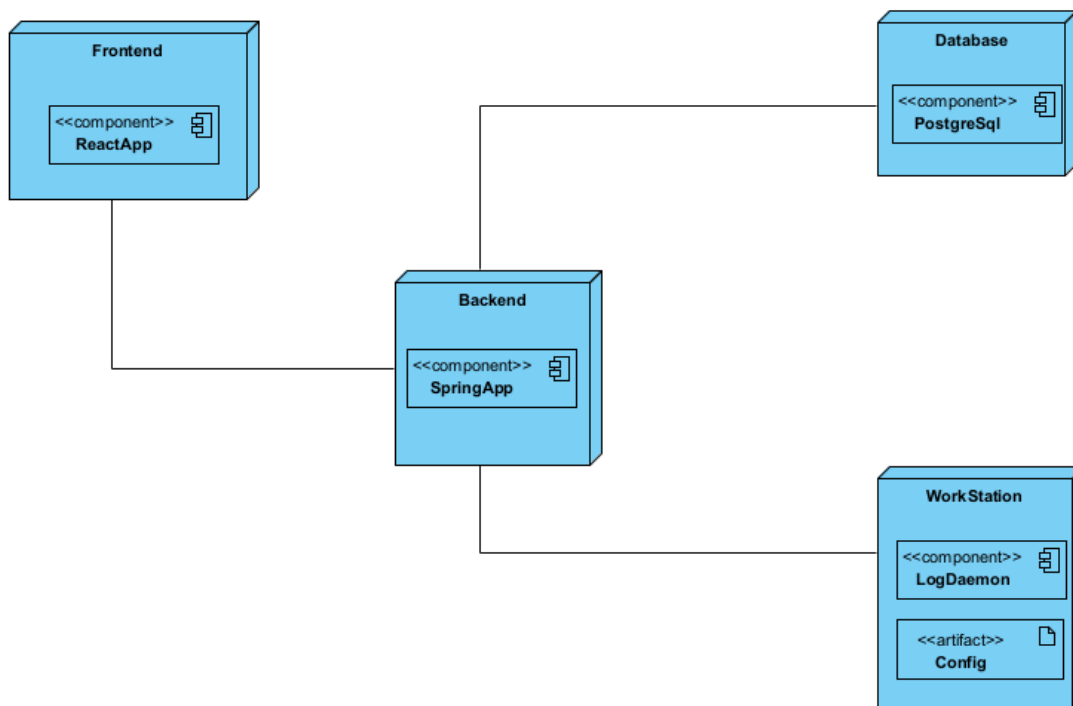


Рисунок 11 – Диаграмма развертывания

Диаграмма развертывания предоставляет визуальный обзор компонентов и их зависимостей, что необходимо для правильного развертывания и

конфигурирования приложения в среде выполнения. Она помогает понять архитектуру системы и определить потребности в ресурсах.

4 Контроль качества программного обеспечения

4.1 Функциональное тестирование

Функциональное тестирование — это процесс обеспечения качества (QA) в рамках цикла разработки программного обеспечения, необходимый для проверки реализуемости функциональных требований, согласно спецификации тестируемого программного обеспечения. Функциональное тестирование проводится для оценки соответствия системы или компонента заданным функциональным требованиям^[1].

Функциональное тестирование является тестированием черного ящика, в связи с чем функциональность программного обеспечения можно протестировать, не зная его внутренней работы. Это снижает требования к тестировщикам в части знания языков программирования или конкретных аспектов реализации программного обеспечения.

4.2 Определение классов эквивалентности

Тестирование программного обеспечения является одним из этапов жизненного цикла ПО, который направлен на проверку соответствия программы требованиям на основе выполнения набора тестов.

В данном тексте рассматриваются функциональное и структурное тестирование приложения для анализа системных журналов.

Таблица 4 – Классы эквивалентности

Поле E-mail	Правильные классы	Неправильные классы
Количество символов в строке	[7,35]	[0,7),(35,+]
Присутствует «@»	Да	Нет
Латинские буквы	[A-Z],[a-z]	[A-Я],[a-я]
Специальные символы	[-],[_],[.]	Все остальные

На основании классов эквивалентности были сформированы возможные ситуации, которые представлены в таблице 15.

Таблица 5 – Функциональное тестирование поля ввода «E-mail»

Входящие значения	Предполагаемый результат	Класс эквивалентности	Результат
Egor.morozov@gmail.com	Успех	1,3,5,7	Успех (рисунок Д.1)
Nosobakasign.com	Ошибка	4	Ошибка (рисунок Д.2)
Почта_наРусском@mail.ru	Ошибка	6	Ошибка (рисунок Д.3)
f@a.d	Ошибка	2	Ошибка (рисунок Д.4)
alexandr-asdf-asdfasdfasdfasdfasdfasdfasdf@gmail.com	Ошибка	2	Ошибка (рисунок Д.5)
qwer#qwer*qwer@gmail.com	Ошибка	8	Ошибка (рисунок Д.6)

Результаты тестирования приведены в приложении Д.

4.3 Анализ граничных условий

Анализ граничных условий может помочь обнаружить ошибки, возникающие на границах классов эквивалентности, а также включать тестирование данных, которые выходят за допустимый диапазон значений. Для проведения анализа было выбрано поле ввода «email». Для этого поля были определены три типа границ: физическая, произвольная и логическая.

Физическая граница устанавливается при нулевом количестве введенных символов.

Произвольная граница определяет, что количество введенных символов должно быть в промежутке от 6 до 35 символов включительно.

Логическая граница устанавливает требование того, что введенное значение должно состоять из латинских букв, цифр и специальных символов, таких как «-«, «_» и «.». Также нужно включить символ «@»

Произвольная граница представлена на рисунке 12.



Рисунок 12 – Произвольная граница ввода поля «Email»

Логическая граница представлена на рисунке 13.

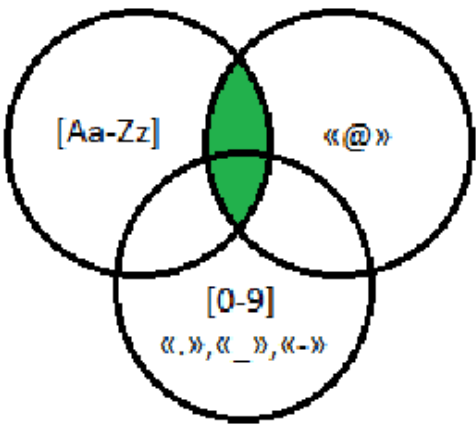


Рисунок 13 – Логическая граница ввода поля «E-mail»

Тестовые наборы приведены в таблице 16.

Таблица 6 – Граничные условия

Граничное условие	Значение	Результат
-------------------	----------	-----------

Пустая строка	[]	Ошибка (рисунок В.7)
Количество символов в строке меньше 7	f@a.d	Ошибка (рисунок В.8)
Количество символов в строке равно 7	fwe@a.d	Пройден успешно (рисунок В.9)
Количество символов в строке равно 35	edogasdf.v.makarosks.11.12.2001@gmail.co m	Пройден успешно (рисунок В.10)
Количество символов в строке больше 35	manymedogasdf.v.001qweqweqweasdfasdfasdf asdfasdfasdf@gmail.com	Ошибка (рисунок В.11)
Строка содержит только допустимые символы	Egor.morozov@gmail.com	Пройден успешно (рисунок В.12)
Строка содержит недопустимые символы	Egor.moroz#####ov@gmail.com	Ошибка (рисунок В.13)
Строка не содержит «@»	Egor.morozovgmail.com	Пройден успешно (рисунок В.14)
Строка содержит русские буквы	русскаяпочта@mail.ru	Ошибка (рисунок В.15)

Результаты тестирования приведены в приложении В.

4.4 Построение функциональной диаграммы

Метод функциональных диаграмм позволяет выбирать высоко результативные тесты.

Для тестирования были выбраны поля ввода «Email» и «Password».

Составляющие функциональной диаграммы:

- был осуществлен вход на сайт;
- не удалось установить соединение с сервером;
- не удалось подключения к интернету;
- вход был осуществлён успешно;
- ошибка установления соединения.

Функциональная диаграмма для выбранных классов эквивалентности представлена на рисунке 14.

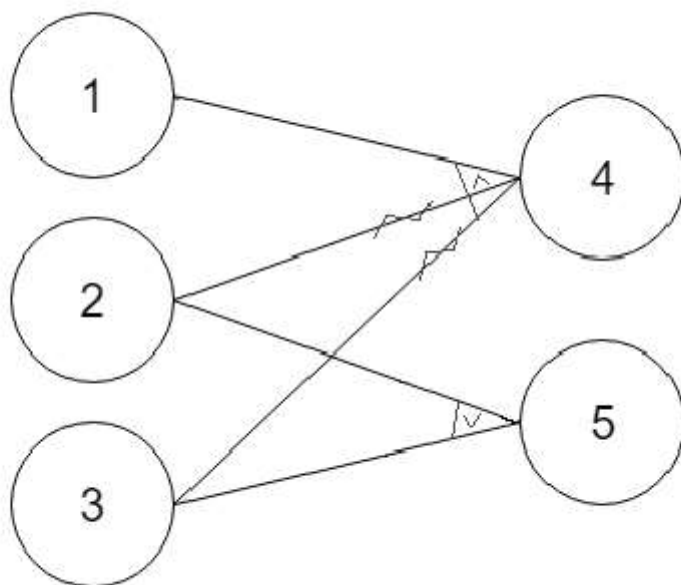


Рисунок 14 – Функциональная диаграмма

Причины и результаты тестов представлены в таблице 7.

Таблица 7 – Причины и тесты

Причины/тесты	1	2	3	4
1	0	0	0	1
2	0	1	1	0
3	1	0	1	0
4	0	0	0	1
5	1	1	1	0

Результатом построения функциональной диаграммы является то, что при отсутствии ошибок соединения и правильно введенных данных, приложение работает корректно, иначе появляется сообщение об ошибке.

4.5 Парное тестирование

Стратегия парного тестирования применяется для сокращения количества тест-кейсов путем использования части входных данных из всего набора, и в данном случае она была использована для поля ввода «email» при помощи программного обеспечения «allpairs». Программа запускается через консоль. В таблице 8 дано описание входных классов эквивалентности для парного тестирования поля ввода «email».

Запуск данного программного обеспечения выполняется из консоли. Описание входных классов эквивалентности для парного тестирования поля ввода «email» представлено в таблице 8.

Таблица 8 – Описание классов эквивалентности

Name	Lastname	Email	Password
goodname	goodlastname	goodemail	goodpassword
badname	badlastname	bademail	badpassword
nullname	nulllastname	[1,6)	[1,6)
othername	otherlastname	[6,35]	[6,35]
defisname	defislastname	(35,+]	(35,+]

Результаты создания тестовых случаев представлены на рисунке 15.

case	name	lastname	email	password	pairings
1	goodName	goodLastname	goodEmail	goodPassword	6
2	goodName	badLastname	badEmail	badPassword	6
3	goodName	nullLastname	[1, 6)	[1, 6)	6
4	goodName	otherLastname	[6, 40]	[6, 40]	6
5	goodName	defisLastname	(40, +]	(40, +]	6
6	badName	goodLastname	badEmail	[1, 6)	6
7	badName	badLastname	goodEmail	[6, 40]	6
8	badName	nullLastname	[6, 40]	goodPassword	6
9	badName	otherLastname	[1, 6)	badPassword	6
10	badName	defisLastname	goodEmail	badPassword	4
11	nullName	goodLastname	[1, 6)	[6, 40]	6
12	nullName	badLastname	[6, 40]	[1, 6)	6
13	nullName	nullLastname	goodEmail	(40, +]	6
14	nullName	otherLastname	badEmail	goodPassword	6
15	nullName	defisLastname	badEmail	[6, 40]	4
16	otherName	goodLastname	[6, 40]	badPassword	6
17	otherName	badLastname	[1, 6)	goodPassword	6
18	otherName	nullLastname	badEmail	[6, 40]	5
19	otherName	otherLastname	goodEmail	[1, 6)	6
20	otherName	defisLastname	[1, 6)	(40, +]	4
21	defisName	goodLastname	(40, +]	goodPassword	5
22	defisName	badLastname	badEmail	(40, +]	5
23	defisName	nullLastname	[6, 40]	badPassword	4
24	defisName	otherLastname	(40, +]	[1, 6)	4
25	defisName	defisLastname	[6, 40]	goodPassword	3
26	badName	goodLastname	(40, +]	(40, +]	3
27	nullName	badLastname	(40, +]	badPassword	4
28	defisName	nullLastname	(40, +]	[6, 40]	3
29	otherName	otherLastname	[6, 40]	(40, +]	2
30	otherName	defisLastname	(40, +]	[1, 6)	2
31	defisName	~goodLastname	goodEmail	~badPassword	1

Рисунок 15 – Результаты создания тестовых случаев

В результате функционального тестирования было выявлено, что при авторизации пользователя в приложении, ошибок не возникает.

4.6 Метрики программного кода проекта

Метрики кода - это числовые значения, которые используются для оценки характеристик программного кода, его свойств и особенностей, связанных с качеством и функциональностью. Метрики кода могут использоваться вместе с

другими методами, такими как тестирование, отладка и анализ кода, для определения качества программного кода и выявления потенциальных проблем.

Они могут охватывать различные аспекты кода, такие как его сложность, размер, эффективность, структуру, повторяемость, надежность и т. д. Обычно метрики кода используются для оценки процесса разработки программного обеспечения и повышения его качества

Для оценки качества кода выбрана метрика Холстеда, которая основывается на четырех параметрах: количество уникальных операторов ($n1$), количество уникальных операндов ($n2$), общее количество операторов ($N1$) и общее количество операндов ($N2$).

С помощью метрики Холстеда проанализируем качество кода для варианта использования «Сохранение». Операнды и операторы приведены в таблице 9.

Таблица 9 – Операнды и операторы

Операторы	Число вхождений	Операнды	Число вхождений
return	2	request	8
=>	4	response	4
map	2	stId	1
stringify	6	ptId	1
toString	3	id	2
json	2	fromDate	6
one	1	toDate	7
{}	5	log	8
.	58	regex	9
declare	2	localhost	8
begin	2	Get_list	7

Продолжение таблицы 9

Операторы	Число вхождений	Операнды	Число вхождений
end	1	error	9
if	2	msg	11
n1 = 12	N1 = 98	n2 = 12	N2=73

На основе данной таблицы рассчитаем метрики кода по Холстеду.

Длина кода программы рассчитывается по формуле:

$$L = N1 + N2 = 98 + 73 = 172.$$

Словарь программы рассчитывается по формуле:

$$V = n1 + n2 = 12 + 12 = 24.$$

Объем кода вычисляется по формуле:

$$P = L \cdot \log(2)V = 175.23.$$

Сложность кода можно рассчитать по формуле:

$$D = (n1 / 2) \cdot (N2 / n2) = (24 / 2) \cdot (172 / 12) = 175$$

Результаты вычисленных метрик показывают, что сложность поддержки кода, для заданного варианта использования, находится на среднем уровне.

Заключение

Разработано веб-приложение анализа системных журналов, имеющее клиент-серверную архитектуру. Клиентская часть была реализована на языке JavaScript с использованием фреймворка React, серверная часть – на языке Java с использованием фреймворка SpringBoot, в качестве СУБД использовался PostgreSQL.

При помощи разработанного приложения можно создавать правила для анализа журналов, подключать к общему серверу и анализировать работу рабочих станций в режиме реального времени. Для приложения спроектирован простой и понятный интерфейс, удобный в использовании пользователю любого уровня подготовки.

Проведено функциональное тестирование, а также определены метрики программного кода проекта по результатам проведенного тестирования можно сделать вывод, что приложение функционирует корректно.

Таким образом, в результате выполнения выпускной квалификационной работы разработано веб-приложение для анализа журналов, отвечающее всем требованиям технического задания.

Список использованных источников

1. Visual Paradigm Community Edition – A Free UML Tool // Visual Paradigm. [Электронный ресурс] URL: <https://www.visual-paradigm.com/solution/freeumltool/> (дата обращения: 25.04.2023)
2. Проектирование информационных систем // КОМПЬЮТЕР ПРЕСС. [Электронный ресурс] URL: <https://compress.ru/article.aspx?id=11764> (дата обращения: 30.04.2023)
3. Разновидности диаграмм UML // Lucidchart. [Электронный ресурс] URL: <https://www.lucidchart.com/blog/ru/types-of-UML-diagrams> (дата обращения: 30.04.2023)
4. Балашова И.Ю., Такташкин Д.В. Современные информационные технологии в проектировании программных средств и комплексов / И.Ю. Балашова, Д.В. Такташкин, под ред. П.П. Макарычев – Пенза: Изд-во ПГУ, 2019. – 121 с.
5. Диаграмма деятельности и особенности ее построения// [Электронный ресурс] / НОУ ИНТУИТ. URL: <https://intuit.ru/studies/courses/32/32/lecture/1020> (дата обращения: 25.05.2023)
6. Data Modeling: An Overview // Toward Data Science. [Электронный ресурс] URL: <https://towardsdatascience.com/data-modeling-an-overview-ed6165a27309> (дата обращения: 26.05.2023)
7. Что такое диаграммы Ганта? // ATLASSIAN. [Электронный ресурс] URL: <https://www.atlassian.com/ru/agile/project-management/gantt-chart> (дата обращения: 07.05.2023)

8. Особенности тестирования «черного ящика» // Лаборатория качества. [Электронный ресурс] URL: <https://quality-lab.ru/blog/key-principles-of-black-box-testing/> (дата обращения: 15.05.2023)
9. Техники тест дизайна // Хекслет. [Электронный ресурс] URL: https://ru.hexlet.io/courses/qa-engineer-workflow/lessons/test-design-techniques/theory_unit (дата обращения: 15.05.2023)
10. Метрики кода и практическая реализация по их сбору и анализу. Часть 1 – метрики // CNews клуб. [Электронный ресурс] URL: https://club.cnews.ru/blogs/entry/metriki_koda_i_prakticheskaya_realizatsiya_po_ih_sboru_i_analizu_chast_1 (дата обращения: 15.05.2023)
11. Тестер // Github. [Электронный ресурс] URL: <https://github.com/grumagargler/tester> (дата обращения: 16.05.2023)
12. Стефанов, С. React.js. Быстрый старт / С. Стефанов. – СПб.: Питер, 2017. – 304 с
13. Javascript-фреймворки: тенденции 2019 года [Электронный ресурс]. – URL: <https://habr.com/ru/company/plarium/blog/433926> (дата обращения 13.06.2023)
14. Аутентификация в приложении с использованием jwt. [Электронный ресурс]. – URL: <https://github.com/auth0/node-jsonwebtoken> (дата обращения: 13.06.2023).
15. Логическая модель базы данных. [Электронный ресурс]. – URL: <https://studfile.net/preview/2083751/page:2/> (дата обращения 13.06.2023)
16. Функциональное тестирование. [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5 (дата обращения 13.06.2023).

Приложение А
ЛИСТИНГ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг А.1 – App

```

import "bootstrap/dist/css/bootstrap.min.css"
import './App.css';
import {BrowserRouter, Navigate, Route, Routes} from "react-router-dom";
import {AuthProvider, useIsAuthenticated} from "react-auth-kit";
import Reg from "./Auth/Reg";
import Auth from "./Auth/Auth";
import {Navbar} from "./components/navbar";
import {ToastContainer} from "react-toastify";
import React from "react";
import {Main} from "./Main/main";
import {Logout} from "./components/signout";
import Example from "./Main/JS";
import Events from "./Events/JS";

function App() {
  const PrivateRoute = ({ Component }) => {
    const isAuthenticated = useIsAuthenticated();
    const auth = isAuthenticated();
    return auth ? <Component /> : <Navigate to="/auth" />;
  };
  return (
    <div className="App">
      <ToastContainer />
      <AuthProvider authType={'localStorage'} authName={'_auth'}>
        <BrowserRouter>
          <Navbar/>
          <Routes>
            <Route path="/reg" element={<Reg/>} />
            <Route path="/auth" element={<Auth/>} />
            <Route path="/" element={<Main/>} />
            <Route path="/events" element={<Events/>} />
            <Route path="/signout" element={<Logout/>} />
          </Routes>
        </BrowserRouter>
      </AuthProvider>
    </div>
  );
}

export default App;

```

Листинг А.2 – Sidemenu

```

import React from "react";
import "bootstrap/dist/js/bootstrap.bundle.js";
import "bootstrap/dist/css/bootstrap.css";
import "./sidemenu.css"

```

```

export function Sidemenu() {
  return (
    <nav id="sidebarMenu" className="collapse d-lg-block collapse bg-
white">
      <div className="position-sticky">
        <a className="list-group-item list-group-item-action py-2
ripple" aria-current="true"
          data-mdb-toggle="collapse" href="#collapseExample1"
          aria-expanded="true"
          aria-controls="collapseExample1">
          <i className="fas fa-tachometer-alt fa-fw me-
3"></i><span>Подключенные станции</span>
        </a>
        <div className="btn-group-vertical w-100 h-100"
          role="group" aria-label="Vertical button group">
          <button type="button" className="btn btn-
primary">Ubuntu 20.04</button>
          <button type="button" className="btn btn-
secondary">Debian 10</button>
          <button type="button" className="btn btn-
secondary">Astra 1.7</button>
          <button type="button" className="btn btn-
secondary">Astra 2.12</button>
        </div>
      </div>
    </nav>
  )
}

return (
  <nav>
    <ul className="pagination">
      <li className="page-item">
        <button
          className="page-link"
          onClick={handleBackButtonClick}
          disabled={previosDisabled}
          aria-disabled={previosDisabled}
          aria-label="previous page"
        >
          Previous
        </button>
      </li>
      {pageItems.map((page) => {
        const className =
          page === currentPage ? "page-item active" :
"page-item";

        return (
          <li key={page} className={className}>
            <button
              className="page-link"
              onClick={handlePageNumber}

```

```

        value={page}
      >
        {page}
      </button>
    </li>
  );
}}}
<li className="page-item">
  <button
    className="page-link"
    onClick={handleNextButtonClick}
    disabled={nextDisabled}
    aria-disabled={nextDisabled}
    aria-label="next page"
  >
    Next
  </button>
</li>
</ul>
</nav>
);
};

const BootyCheckbox = React.forwardRef(({onClick, ...rest}, ref) => (
  <div className="form-check">
    <input
      htmlFor="booty-check"
      type="checkbox"
      className="form-check-input"
      ref={ref}
      onClick={onClick}
      {...rest}
    />
    <label className="form-check-label" id="booty-check"/>
  </div>
));

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
}

function TabPanel(props: TabPanelProps) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={simple-tabpanel-${index}}
      aria-labelledby={simple-tab-${index}}
      {...other}
    >

```

```

        {value === index && (
          <Box sx={{ p: 3 }}>
            <Typography>{children}</Typography>
          </Box>
        )}
      </div>
    );
  }
function allyProps(index: number) {
  return {
    id: simple-tab-${index},
    'aria-controls': simple-tabpanel-${index},
  };
}

export function Main() {
  const [data, setData] = useState([])
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number)
=> {
    setValue(newValue);
  };

  return (
    <div className="container-fluid">
      <div className="row">
        <div className="col-lg-2 sidebar-fixed">
          <Sidemenu/>
        </div>
        <div className="col-lg-10">
          <Example/>
        </div>
      </div>
    </div>
  );
}

```

Листинг A.3 – RulesTable

```

import React, {useEffect, useState} from "react";
import DataTable from 'react-data-table-component';
import "bootstrap/dist/js/bootstrap.bundle.js";
import "bootstrap/dist/css/bootstrap.css";
import "./styles.css";
import {Sidemenu} from "./sidemenu";
import Example from "./JS";
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';

```

```

function getNumberOfPages(rowCount, rowsPerPage) {
  return Math.ceil(rowCount / rowsPerPage);
}

function toPages(pages) {
  const results = [];

  for (let i = 1; i < pages; i++) {
    results.push(i);
  }

  return results;
}

const columns = [
  {
    name: "Title",
    selector: (row) => row.title,
    sortable: true
  },
  {
    name: "Year",
    selector: (row) => row.year,
    sortable: true
  },
  {
    name: "Runtime (m)",
    selector: (row) => row.runtime,
    sortable: true,
    right: true
  },
  {
    button: true,
    cell: () => (
      <div className="App">
        <div class="openbtn text-center">
          <button
            type="button"
            class="btn btn-primary"
            data-bs-toggle="modal"
            data-bs-target="#myModal"
          >
            Open modal
          </button>
          <div class="modal" tabindex="-1" id="myModal">
            <div class="modal-dialog">
              <div class="modal-content">
                <div class="modal-header">
                  <h5 class="modal-title">Modal
title</h5>
                <div class="modal-body">
                  <div class="modal-footer">
                    <button
                      type="button"

```

```

        class="btn-close"
        data-bs-dismiss="modal"
        aria-label="Close"
      ></button>
    </div>
    <div class="modal-body">
      <p>Modal body text goes here.</p>
    </div>
    <div class="modal-footer">
      <button
        type="button"
        class="btn btn-secondary"
        data-bs-dismiss="modal"
      >
        Close
      </button>
      <button type="button" class="btn btn-
primary">
        Save changes
      </button>
    </div>
  </div>
</div>
</div>
</div>
</div>
)
}
];
return (
  <Dialog open={open}>
    <DialogTitle textAlign="center">Добавить новое
правило</DialogTitle>
    <DialogContent>
      <form onSubmit={(e) => e.preventDefault()}>
        <Stack
          sx={{
            width: '100%',
            minWidth: { xs: '300px', sm: '360px', md:
'400px' },
            gap: '1.5rem',
          }}
        >
          {columns.map((column) => (
            <TextField
              key={column.accessorKey}
              label={column.header}
              name={column.accessorKey}
              onChange={(e) =>
                setValues({ ...values,
[e.target.name]: e.target.value })
              }
            </>
          ))}
        </form>
      </DialogContent>
    </Dialog>
  </return>
)

```

```

        ))}
      </Stack>
    </form>
  </DialogContent>
  <DialogActions sx={{ p: '1.25rem' }}>
    <Button onClick={onClose}>Отменить</Button>
    <Button color="secondary" onClick={handleSubmit}
variant="contained">
      Добавить новое правило
    </Button>
  </DialogActions>
</Dialog>
);
};

const validateRequired = (value) => !!value.length;
const validateEmail = (email) =>
  !!email.length &&
  email
    .toLowerCase()
    .match(
      /^(((^<>() [\]\.\.,;:\s@"]+(\. [^<>() [\]\.\.,;:\s@"]+)*)|("."+) )@((\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\]|([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$)/,
    );

export default Example;

```

Листинг A.4 – TableConfig

```

return (
  <>
    <MaterialReactTable
      displayColumnDefOptions={{
        'mrt-row-actions': {
          muiTableHeadCellProps: {
            align: 'center',
          },
          size: 120,
        },
      }}
      columns={columns}
      data={tableData}
      localization={MRT_Localization_RU}
      editingMode="modal" //default
      enableColumnOrdering
      enableEditing
      onEditingRowSave={handleSaveRowEdits}
      onEditingRowCancel={handleCancelRowEdits}
      renderRowActions={({ row, table }) => (

```

```

        <Box sx={{ display: 'flex', gap: '1rem' }}>
          <Tooltip arrow placement="left" title="Edit">
            <IconButton onClick={() =>
table.setEditingRow(row)}>
              <Edit />
            </IconButton>
          </Tooltip>
          <Tooltip arrow placement="right" title="Delete">
            <IconButton color="error" onClick={() =>
handleDeleteRow(row)}>
              <Delete />
            </IconButton>
          </Tooltip>
        </Box>
      )}
      renderTopToolbarCustomActions={() => (
        <Box>
          <Button
            color="success"
            onClick={() => setCreateModalOpen(true)}
            variant="contained"
          >
            Добавить новое правило
          </Button>
          <Button
            color="success"
            onClick={() => setCreateModalOpen(true)}
            variant="contained"
          >
            Сохранить изменения
          </Button>
        </Box>
      )}
      options={{
        actionsColumnIndex: -10
      }}
    />
    <CreateNewAccountModal
      columns={columns}
      open={createModalOpen}
      onClose={() => setCreateModalOpen(false)}
      onSubmit={handleCreateNewRow}
    />
  </>
);
};

//example of creating a mui dialog modal for creating new rows
export const CreateNewAccountModal = ({ open, columns, onClose, onSubmit
}) => {
  const [values, setValues] = useState(() =>
    columns.reduce((acc, column) => {
      acc[column.accessorKey ?? ''] = '';

```

```

        return acc;
      }, {})),
    );

    const handleSubmit = () => {
      //put your validation logic here
      onSubmit(values);
      onClose();
    }

```

Листинг A.5 – EventTable

```

import React, { useCallback, useMemo, useState } from 'react';
import { MaterialReactTable } from 'material-react-table';
import {
  Box,
  Button,
  Dialog,
  DialogActions,
  DialogContent,
  DialogTitle,
  IconButton,
  MenuItem,
  Stack,
  TextField,
  Tooltip,
} from '@mui/material';
import { Delete, Edit } from '@mui/icons-material';
import { data, } from './makeData';
import { MRT_Localization_RU } from "material-react-table/locales/ru";

const Example = () => {
  const [createModalOpen, setCreateModalOpen] = useState(false);
  const [tableData, setTableData] = useState(() => data);
  const [validationErrors, setValidationErrors] = useState({});

  const handleCreateNewRow = (values) => {
    tableData.push(values);
    setTableData([...tableData]);
  };

  const handleSaveRowEdits = async ({ exitEditingMode, row, values })
=> {
    if (!Object.keys(validationErrors).length) {
      tableData[row.index] = values;
      //send/receive api updates here, then refetch or update local
table data for re-render
      setTableData([...tableData]);
      exitEditingMode(); //required to exit editing mode and close
modal
    }
  };
};

```

```

const handleCancelRowEdits = () => {
  setValidationErrors({});
};

const handleDeleteRow = useCallback(
  (row) => {
    if (
      !window.confirm(Are you sure you want to delete
        ${row.getValue('name')}))
    ) {
      return;
    }
    //send api delete request here, then refetch or update local
    table data for re-render
    tableData.splice(row.index, 1);
    setTableData([...tableData]);
  },
  [tableData],
);

const getCommonEditTextFieldProps = useCallback(
  (cell) => {
    return {
      error: !!validationErrors[cell.id],
      helperText: validationErrors[cell.id],
      onBlur: (event) => {
        const isValid = true
        if (!isValid) {
          //set validation error for cell if invalid
          setValidationErrors({
            ...validationErrors,
            [cell.id]: ${cell.column.columnDef.header} is
required,
          });
        } else {
          //remove validation error for cell if valid
          delete validationErrors[cell.id];
          setValidationErrors({
            ...validationErrors,
          });
        }
      },
    };
  },
  [validationErrors],
);

const columns = useMemo(
  () => [
    {
      accessorKey: 'id',
      header: 'ID',
    }
  ]
);

```

```

        enableColumnOrdering: false,
        enableEditing: false, //disable editing on this column
        enableSorting: false,
      },
      {
        accessorKey: 'level',
        header: 'Уровень угрозы',
        muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
          ...getCommonEditTextFieldProps(cell),
        }),
      },
      {
        accessorKey: 'signature',
        header: 'Сигнатура',
        muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
          ...getCommonEditTextFieldProps(cell),
        }),
      },
      {
        accessorKey: 'path',
        header: 'Путь к файлу',
        muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
          ...getCommonEditTextFieldProps(cell),
        }),
      }
    ],
    [getCommonEditTextFieldProps],
  );
const columns = useMemo(
  () => [
    {
      accessorKey: 'id',
      header: 'ID',
      enableColumnOrdering: false,
      enableEditing: false, //disable editing on this column
      enableSorting: false,
    },
    {
      accessorKey: 'level',
      header: 'Уровень угрозы',
      muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
        ...getCommonEditTextFieldProps(cell),
      }),
    },
    {
      accessorKey: 'signature',
      header: 'Сигнатура',
      muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
        ...getCommonEditTextFieldProps(cell),
      }),
    },
    {
      accessorKey: 'path',

```

```

        header: 'Путь к файлу',
        muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
            ...getCommonEditTextFieldProps(cell),
        }),
    },
    {
        accessorKey: 'time',
        header: 'Время срабатывания',
        muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
            ...getCommonEditTextFieldProps(cell),
        }),
    },
    {
        accessorKey: 'station',
        header: 'Станция',
        muiTableBodyCellEditTextFieldProps: ({ cell }) => ({
            ...getCommonEditTextFieldProps(cell),
        }),
    }
],
[getCommonEditTextFieldProps],
);

return (
    <>
        <MaterialReactTable
            columns={columns}
            data={tableData}
            localization={MRT_Localization_RU}
            editingMode="modal" //default
            enableColumnOrdering
            onEditingRowSave={handleSaveRowEdits}
            onEditingRowCancel={handleCancelRowEdits}
        />
    </>
);
};

//example of creating a mui dialog modal for creating new rows
export default Events;

import React, { useCallback, useMemo, useState } from 'react';
import { MaterialReactTable } from 'material-react-table';
import {
    Box,
    Button,
    Dialog,
    DialogActions,
    DialogContent,
    DialogTitle,
    IconButton,
    MenuItem,
    Stack,

```

```

    TextField,
    Tooltip,
  } from '@mui/material';
import { Delete, Edit } from '@mui/icons-material';
import { data, } from './makeData';
import { MRT_Localization_RU } from "material-react-table/locales/ru";

const Events = () => {
  const [createModalOpen, setCreateModalOpen] = useState(false);
  const [tableData, setTableData] = useState(() => data);
  const [validationErrors, setValidationErrors] = useState({});

  const handleCreateNewRow = (values) => {
    tableData.push(values);
    setTableData([...tableData]);
  };

  const handleSaveRowEdits = async ({ exitEditingMode, row, values })
=> {
    if (!Object.keys(validationErrors).length) {
      tableData[row.index] = values;
      //send/receive api updates here, then refetch or update local
table data for re-render
      setTableData([...tableData]);
      exitEditingMode(); //required to exit editing mode and close
modal
    }
  };

  const handleCancelRowEdits = () => {
    setValidationErrors({});
  };

  const handleDeleteRow = useCallback(
    (row) => {
      if (
        !window.confirm(Are you sure you want to delete
${row.getValue('name')}
      ) {
        return;
      }
      //send api delete request here, then refetch or update local
table data for re-render
      tableData.splice(row.index, 1);
      setTableData([...tableData]);
    },
    [tableData],
  );

  const getCommonEditTextFieldProps = useCallback(
    (cell) => {
      return {
        error: !!validationErrors[cell.id],

```

```

        helperText: validationErrors[cell.id],
        onBlur: (event) => {
            const isValid = true
            if (!isValid) {
                //set validation error for cell if invalid
                setValidationErrors({
                    ...validationErrors,
                    [cell.id]: `${cell.column.columnDef.header} is
required,

                });
            } else {
                //remove validation error for cell if valid
                delete validationErrors[cell.id];
                setValidationErrors({
                    ...validationErrors,
                });
            }
        },
    },
    [validationErrors],
);
import React, {useEffect} from "react";
import {Link, useNavigate} from "react-router-dom";
import { ShoppingCart } from "phosphor-react";
import "./navbar.css";
import {useIsAuthenticated, useSignOut} from "react-auth-kit";
import {toast} from "react-toastify";

export const Logout = () => {
    const signOut = useSignOut()
    const notify = () => toast("Выход был выполнен успешно");
    const navigate = useNavigate()
    useEffect(() => {
        signOut()
        notify()
        navigate("/")
    }, [])

    return (
        <>
        </>
    )
};

```

Листинг А.6 – TopBar

```

import React from "react";
import {Link} from "react-router-dom";
import "./navbar.css";
import {useIsAuthenticated, useSignOut} from "react-auth-kit";

```

```

export const Navbar = () => {
  const signOut = useSignOut()
  const isAuthenticated = useIsAuthenticated()
  return (
    <div className="navbar">
      <div className="links">
        <Link to="/reg"> Зарегистрироваться </Link>
        <Link to="/auth"> Войти </Link>
        <Link to="/signout"> Выйти </Link>
        <Link to="/"> Главная страница </Link>
      </div>
    </div>
  );
};

```

Листинг A.7 – Registration

```

import React, {useState} from "react"
import "./Auth.css"
import $ from 'jquery'
import {useSignIn} from "react-auth-kit";
import {useNavigate} from "react-router-dom";
import {toast} from "react-toastify";

export default function (props) {
  const signIn = useSignIn()
  const notify = () => toast("Регистрация была выполнена успешно");

  const [email, setEmail] = useState(0)
  const [name, setName] = useState(0)
  const [password, setPassword] = useState(0)
  const navigate = useNavigate()

  let sendReg = function (event) {
    event.preventDefault()
    var myHeaders = new Headers();
    myHeaders.append("Content-Type", "application/json");

    let data = {
      "email": email,
      "name": name,
      "password": password
    }

    let requestOptions = {
      method: 'POST',
      headers: myHeaders,
      body: JSON.stringify(data),
    };

    fetch("http://localhost:8080/auth/register", requestOptions)
      .then(response => response.json())

```

```

        .then(result => {
            singIn({
                token: result.token,
                expiresIn: 3600,
                tokenType: "Bearer",
                authState: {email: data['email']}
            })
        })
        .catch(error => console.log('error', error));
    notify()
    navigate("/auth")
}

return (
    <div className="Auth-form-container">
        <form id="auth-formid" onSubmit={sendReg} className="Auth-
form">
            <div className="Auth-form-content">
                <h3 className="Auth-form-
title">Зарегистрироваться</h3>
                <div className="form-group mt-3">
                    <label>Имя пользователя</label>
                    <input
                        onChange={(e) => {setName(e.target.value)}}
                        type="text"
                        name="name"
                        className="form-control mt-1"
                        placeholder="e.g Jane Doe"
                    />
                </div>
                <div className="form-group mt-3">
                    <label>Электронная почта</label>
                    <input
                        onChange={(e) => {setEmail(e.target.value)}}
                        type="email"
                        name="email"
                        className="form-control mt-1"
                        placeholder="Email Address"
                    />
                </div>
                <div className="form-group mt-3">
                    <label>Пароль</label>
                    <input
                        onChange={(e) =>
{setPassword(e.target.value)}}
                        type="password"
                        name="password"
                        className="form-control mt-1"
                        placeholder="Password"
                    />
                </div>
                <div className="d-grid gap-2 mt-3">

```

```

        <button type="submit" className="btn btn-
primary">
            Зарегистрироваться
        </button>
    </div>
</div>
</form>
</div>
)
}

```

Листинг A.8 – Login

```

import React, {useState} from "react"
import "./Auth.css"
import {useNavigate} from "react-router-dom";
import {useSignIn} from "react-auth-kit";
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Form, Button, Alert } from 'react-bootstrap';

export default function (props) {
    const notify = () => toast("Вход был выполнен успешно");
    const signIn = useSignIn()
    const navigate = useNavigate()

    const [name, setName] = useState(0)
    const [password, setPassword] = useState(0)
    const [showError, setShowError] = useState(false)

    function toReg() {
        navigate("/reg")
    }

    function signInForm(event) {
        event.preventDefault()
        let myHeaders = new Headers();
        myHeaders.append("Content-Type", "application/json");

        let data = {
            "username": name,
            "password": password
        }

        let requestOptions = {
            method: 'POST',
            headers: myHeaders,
            body: JSON.stringify(data),
        };

        fetch("http://localhost:8080/auth/token", requestOptions)
            .then(response => {

```

```

        if (!response.ok) {
            throw new Error("Wrong auth")
        }
        return response.text();
    })
    .then(async result => {
        signIn({
            token: result,
            expiresIn: 3600,
            tokenType: "Bearer",
            authState: {email: data['username']}
        })
        notify()
        navigate("/")
    })
    .catch(error => {
        console.log('error', error);
        setShowError(true)
    });
}

return (
    <div className="Auth-form-container">
        <form id="auth-formid" onSubmit={signInForm} className="Auth-
form">
            <div className="Auth-form-content">
                {showError && (
                    <Alert variant="danger">
                        Неправильный логин или пароль. Пожалуйста,
проверьте свои учетные данные и повторите попытку.
                    </Alert>
                )}
                <h3 className="Auth-form-title">Войти в личный
кабинет</h3>
                <div className="form-group mt-3">
                    <label>Логин</label>
                    <input
                        onChange={(e) => {
                            setName(e.target.value)
                        }}
                        type="text"
                        className="form-control mt-1"
                        placeholder="Enter email"
                    />
                </div>
                <div className="form-group mt-3">
                    <label>Пароль</label>
                    <input
                        onChange={(e) => {
                            setPassword(e.target.value)
                        }}
                        type="password"

```

```

        className="form-control mt-1"
        placeholder="Enter password"
      />
    </div>
    <div className="d-grid gap-2 mt-3">
      <button type="submit" className="btn btn-
primary">
        Войти
      </button>
    </div>
    <div className="d-grid gap-2 mt-3">
      <button onClick={toReg} className="btn btn-
primary">
        Зарегистрироваться
      </button>
    </div>
  </div>
</form>
</div>
)
}

```

Приложение Б
ЛИСТИНГ СЕРВЕРНОГО ПРИЛОЖЕНИЯ НА ЯЗЫКЕ
JAVA
(рекомендуемое)

Листинг Б.1 – AuthConfig

```

package com.javatechie.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.AuthenticationProvider
;
import
org.springframework.security.authentication.dao.DaoAuthenticationP
rovider;
import
org.springframework.security.config.annotation.authentication.conf
iguration.AuthenticationConfiguration;
import
org.springframework.security.config.annotation.web.builders.HttpSe
curity;
import
org.springframework.security.config.annotation.web.configuration.E
nableWebSecurity;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity

```

```

public class AuthConfig {

    @Bean
    public UserDetailsService userDetailsService(){
        return new CustomUserDetailsService();
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
        return http.csrf().disable().cors().disable()
            .authorizeHttpRequests()
            .requestMatchers("/auth/login", "/auth/register",
"/auth/token", "/auth/validate").permitAll()
            .requestMatchers("/error").permitAll()
            .and()
            .build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public AuthenticationProvider authenticationProvider(){
        DaoAuthenticationProvider authenticationProvider=new
DaoAuthenticationProvider();

        authenticationProvider.setUserDetailsService(userDetailsService())
;
    }
}

```

```

authenticationProvider.setPasswordEncoder(passwordEncoder());
        return authenticationProvider;
    }

    @Bean
    public AuthenticationManager
authenticationManager(AuthenticationConfiguration config) throws
Exception {
        return config.getAuthenticationManager();
    }
}

```

Листинг Б.2 – CustomUserDetails

```

package com.javatechie.config;

import com.javatechie.entity.UserCredential;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;

public class CustomUserDetails implements UserDetails {

    private final String username;
    private final String password;

    public CustomUserDetails(UserCredential userCredential) {
        this.username = userCredential.getName();
        this.password = userCredential.getPassword();
    }
}

```

```
@Override
public Collection<? extends GrantedAuthority> getAuthorities()
{
    return null;
}

@Override
public String getPassword() {
    return password;
}

@Override
public String getUsername() {
    return username;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
```

```

        return true;
    }
}

```

Листинг Б.3 – CustomUserDetailsService

```

package com.javatechie.config;

import com.javatechie.entity.UserCredential;
import com.javatechie.repository.UserCredentialRepository;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Component;

import java.util.Optional;

@Component
public class CustomUserDetailsService implements
UserDetailsService {

    @Autowired
    private UserCredentialRepository repository;

    @Override

```

```

    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        Optional<UserCredential> credential =
repository.findByName(username);
        return
credential.map(CustomUserDetails::new).orElseThrow(() -> new
UsernameNotFoundException("user not found with name : " +
username));
    }
}

```

Листинг Б.4 – AuthController

```

package com.javatechie.controller;

import com.javatechie.dto.AuthRequest;
import com.javatechie.entity.UserCredential;
import com.javatechie.service.AuthService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthen
ticationToken;
import org.springframework.security.core.Authentication;
import
org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

@RestController

```

```

@RequestMapping("/auth")
@AllArgsConstructor
public class AuthController {
    private final AuthService service;
    private final AuthenticationManager authenticationManager;

    @PostMapping("/register")
    public ResponseEntity<String> addNewUser(@RequestBody
UserCredential user) throws ResponseStatusException{
        if (service.saveUser(user)) {
            return
ResponseEntity.status(HttpStatus.CREATED).build();
        }

        throw new ResponseStatusException(HttpStatus.BAD_REQUEST,
"Not found");
    }

    @PostMapping("/token")
    public ResponseEntity<String> getToken(@RequestBody
AuthRequest authRequest) {
        try {
            Authentication authenticate =
authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(authRequest.getUsername(),
authRequest.getPassword()));
            if (authenticate.isAuthenticated()) {
                String token =
service.generateToken(authRequest.getUsername());
                return
ResponseEntity.status(HttpStatus.OK).body(token);
            }
        } catch (RuntimeException exception) {

```

```

        return
        ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Could not
        authenticate user");
    }

    return
    ResponseEntity.status(HttpStatus.NOT_FOUND).build();
}

@GetMapping("/validate")
public void validateToken(@RequestParam("token") String token)
{
    service.validateToken(token);
}
}

```

Листинг Б.5 – AuthRequest

```

package com.javatechie.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class AuthRequest {

    private String username;
    private String password;
}

```

```
}
```

Листинг Б.6 – UserCredential

```
package com.javatechie.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class UserCredential {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    @Column(unique = true)
    private String email;
    private String password;
}
```

Листинг Б.7 – CustomForbiddenException

```
package com.javatechie.exceptions;

import org.springframework.http.HttpStatus;
```

```
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.server.ResponseStatusException;

@ResponseStatus(HttpStatus.FORBIDDEN)
public class CustomForbiddenException extends
ResponseStatusException {
    public CustomForbiddenException(String message) {
        super(HttpStatus.FORBIDDEN, message);
    }
}
```

Листинг Б.8 – UserCredentialRepository

```
package com.javatechie.repository;

import com.javatechie.entity.UserCredential;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface UserCredentialRepository extends
JpaRepository<UserCredential, Integer> {
    Optional<UserCredential> findByName(String username);

    Optional<UserCredential> findByEmail(String username);
}
```

Листинг Б.9 – AuthService

```
package com.javatechie.service;

import com.javatechie.entity.UserCredential;
import com.javatechie.repository.UserCredentialRepository;
```

```

import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

@Service
@AllArgsConstructor
public class AuthService {
    private final UserCredentialRepository repository;
    private final PasswordEncoder passwordEncoder;
    private final JwtService jwtService =
JwtService.getInstance();

    public boolean saveUser(UserCredential credential) {
        if
(repository.findByEmail(credential.getEmail()).isPresent()) {
            return false;
        }

credential.setPassword(passwordEncoder.encode(credential.getPasswo
rd()));

        repository.save(credential);
        return true;
    }

    public String generateToken(String username) {
        return jwtService.generateToken(username);
    }

    public void validateToken(String token) {
        jwtService.validateToken(token);
    }
}

```

```
    }
}
```

Листинг Б.10 – CurrentUser

```
package com.javatechie.service;

import com.javatechie.entity.UserCredential;

public final class CurrentUser {
    private static CurrentUser INSTANCE;
    private UserCredential userCredential;

    public UserCredential getUserCredential() {
        return userCredential;
    }

    public void setUserCredential(UserCredential userCredential) {
        this.userCredential = userCredential;
    }

    private CurrentUser() {
    }

    public static CurrentUser getInstance() {
        if (INSTANCE == null) {
            INSTANCE = new CurrentUser();
        }

        return INSTANCE;
    }
}
```

Листинг Б.11 – JwtService

```
package com.javatechie.service;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.io.Decoders;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import java.security.Key;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class JwtService {

    private static final String SECRET =
"5367566B59703373367639792F423F4528482B4D6251655468576D5A71347437"
;

    private static JwtService INSTANCE;
    private JwtService() {}

    public static JwtService getInstance() {
        if (INSTANCE == null) {
            INSTANCE = new JwtService();
        }

        return INSTANCE;
    }
}
```

```

    public void validateToken(final String token) {

        Jwts.parserBuilder().setSigningKey(getSignKey()).build().parseClaimsJws(token);

    }

    public String generateToken(String userName) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userName);
    }

    private String createToken(Map<String, Object> claims, String
userName) {
        return Jwts.builder()
            .setClaims(claims)
            .setSubject(userName)
            .setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis()
+ 1000 * 60 * 30 * 60))
            .signWith(getSignKey(),
SignatureAlgorithm.HS256).compact();
    }

    private Key getSignKey() {
        byte[] keyBytes = Decoders.BASE64.decode(SECRET);
        return Keys.hmacShaKeyFor(keyBytes);
    }
}

```

Листинг Б.12 – IdentityServiceApplication

```

package com.javatechie;

```

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class IdentityServiceApplication {

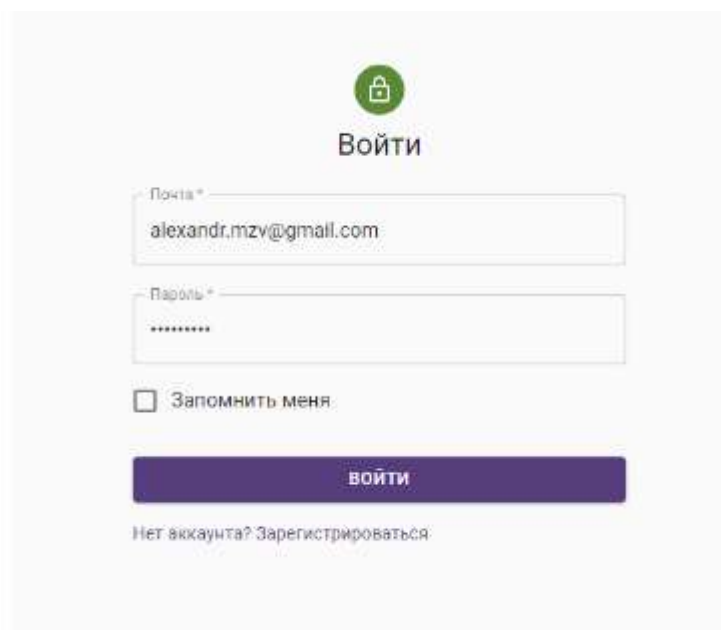
    public static void main(String[] args) {
        SpringApplication.run(IdentityServiceApplication.class,
args);
    }

}
```

Приложение В

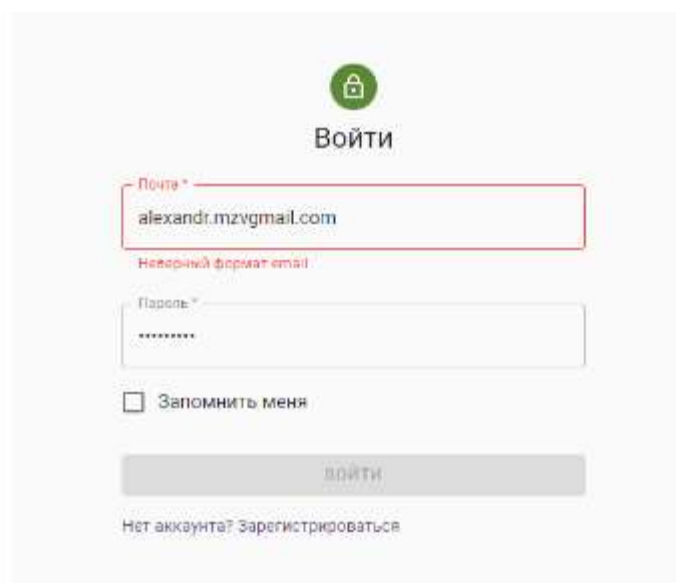
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

(рекомендуемое)



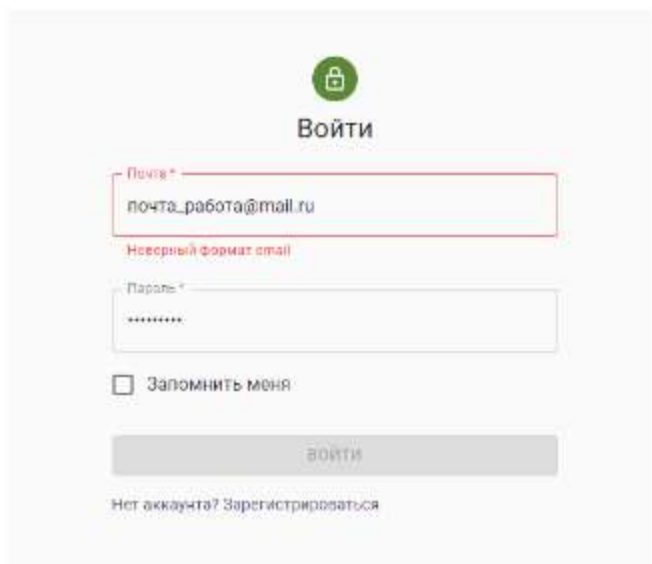
The image shows a login form titled "Войти" (Login) with a green lock icon. It contains two input fields: "Почта*" (Email) with the value "alexandr.mzv@gmail.com" and "Пароль*" (Password) with masked characters. Below the password field is a checkbox labeled "Запомнить меня" (Remember me). A purple button labeled "ВОЙТИ" (Login) is positioned below the checkbox. At the bottom, there is a link "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.1 – Функциональное тестирование методом определения классов эквивалентности, успешный



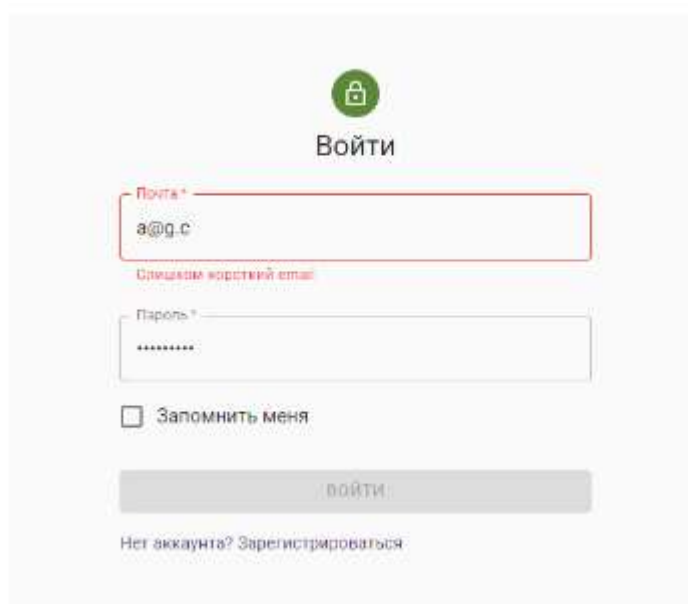
The image shows the same login form as in Figure B.1, but with validation errors. The "Почта*" field is outlined in red and has the error message "Неверный формат email" (Invalid email format) below it. The "Пароль*" field is also outlined in red. The "Запомнить меня" checkbox is present. The "ВОЙТИ" button is now greyed out. The link "Нет аккаунта? Зарегистрироваться" remains at the bottom.

Рисунок В.2 – Функциональное тестирование методом определения классов эквивалентности, нет символа



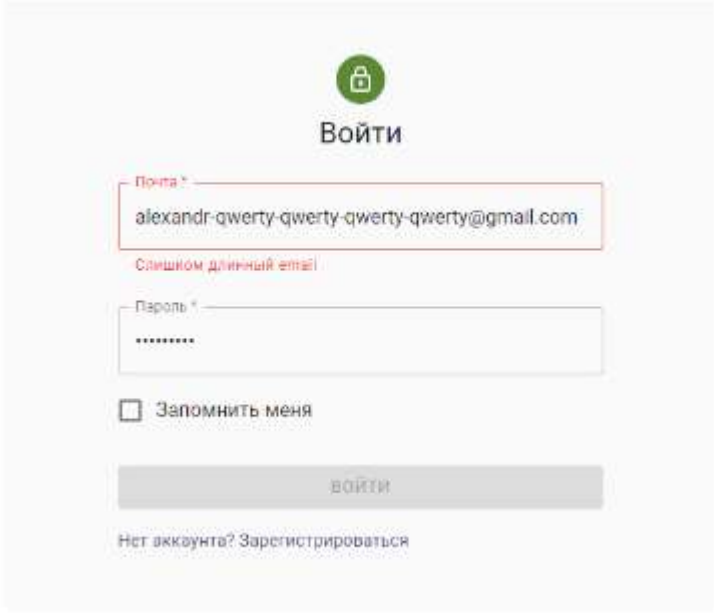
The screenshot shows a login form titled "Войти" (Login) with a green lock icon. The "Почта*" (Email) field contains "почта_работа@mail.ru" and has a red border with a red error message "Неверный формат email" (Invalid email format). The "Пароль*" (Password) field contains "*****". There is a checkbox for "Запомнить меня" (Remember me) and a "ВОЙТИ" (Login) button. At the bottom, it says "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.3 – Функциональное тестирование методом определения классов эквивалентности, русские символы



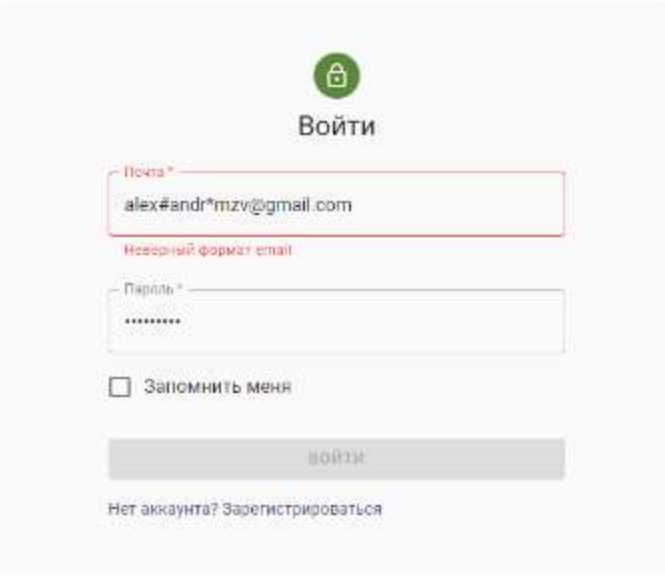
The screenshot shows the same login form as Figure В.3, but the "Почта*" field contains "a@q.c" and has a red border with a red error message "Слишком короткий email" (Email too short). The "Пароль*" field contains "*****". There is a checkbox for "Запомнить меня" (Remember me) and a "ВОЙТИ" (Login) button. At the bottom, it says "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.4 – Функциональное тестирование методом определения классов эквивалентности, слишком короткая почта



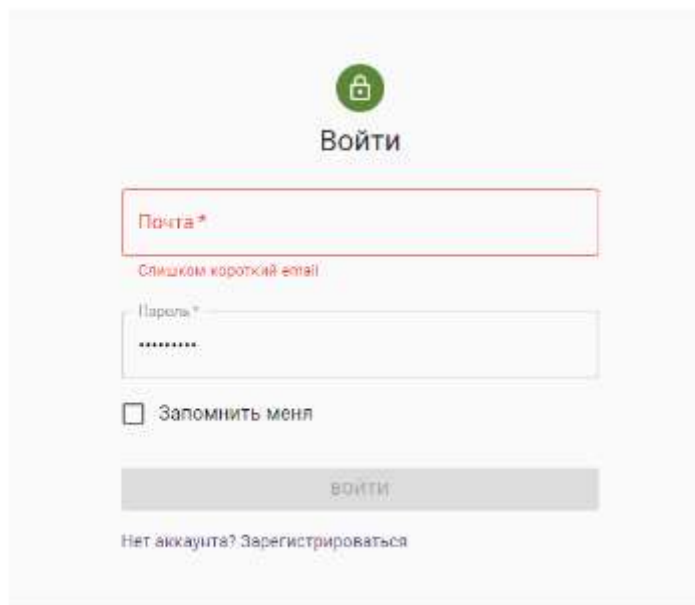
The screenshot shows a login page titled "Войти" (Login) with a green padlock icon. The "Почта" (Email) field contains the long address "alexandr-qwerty-qwerty-qwerty-qwerty@gmail.com". A red border highlights the field, and a red error message "Слишком длинный email" (Email is too long) is displayed below it. The "Пароль" (Password) field is filled with asterisks. There is a checkbox for "Запомнить меня" (Remember me) and a "Войти" (Login) button. At the bottom, there is a link "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.5 – Функциональное тестирование методом определения классов эквивалентности, слишком длинная почта



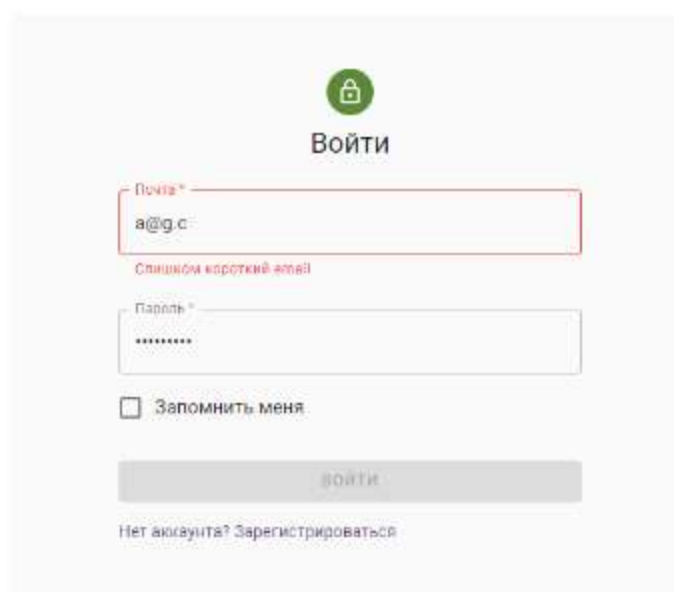
The screenshot shows the same login page. The "Почта" field now contains "alex#andr*mzv@gmail.com". A red border highlights the field, and a red error message "Неверный формат email" (Invalid email format) is displayed below it. The "Пароль" field is filled with asterisks. The "Запомнить меня" checkbox and "Войти" button are present. The link "Нет аккаунта? Зарегистрироваться" is at the bottom.

Рисунок В.6 – Функциональное тестирование методом определения классов эквивалентности, недопустимые символы



The screenshot shows a login form titled "Войти" (Login) with a green lock icon. The form contains two input fields: "Почта*" (Email) and "Пароль*" (Password). The "Почта*" field has a red border and a red error message "Слишком короткий email" (Email is too short) below it. The "Пароль*" field has a red border and a red error message "Пароль слишком короткий" (Password is too short) below it. There is a checkbox labeled "Запомнить меня" (Remember me) and a "ВОЙТИ" (Login) button. At the bottom, there is a link "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.7 – Функциональное тестирование методом анализа граничных условий, нет почты



The screenshot shows a login form titled "Войти" (Login) with a green lock icon. The form contains two input fields: "Почта*" (Email) and "Пароль*" (Password). The "Почта*" field has a red border and a red error message "Слишком короткий email" (Email is too short) below it. The "Пароль*" field has a red border and a red error message "Пароль слишком короткий" (Password is too short) below it. There is a checkbox labeled "Запомнить меня" (Remember me) and a "ВОЙТИ" (Login) button. At the bottom, there is a link "Нет аккаунта? Зарегистрироваться" (No account? Register).

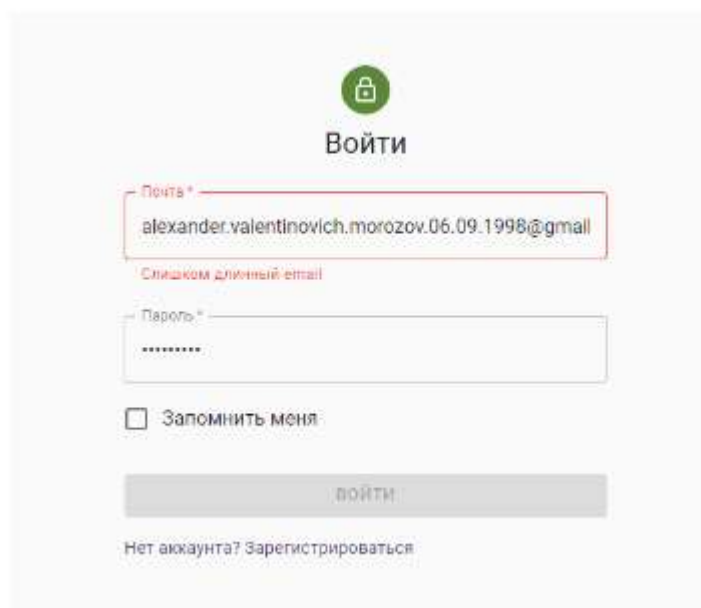
Рисунок В.8 – Функциональное тестирование методом анализа граничных условий, короткая почта

The screenshot shows a login form with the title "Войти" (Login) and a green lock icon. The form contains two input fields: "Почта *" (Email) and "Пароль *" (Password). The email field contains the text "a@g.ru". The password field contains a series of dots. Below the password field is a checkbox labeled "Запомнить меня" (Remember me). A purple button labeled "ВОЙТИ" (Login) is positioned below the checkbox. At the bottom of the form, there is a link that reads "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.9 – Функциональное тестирование методом анализа граничных условий, успешно

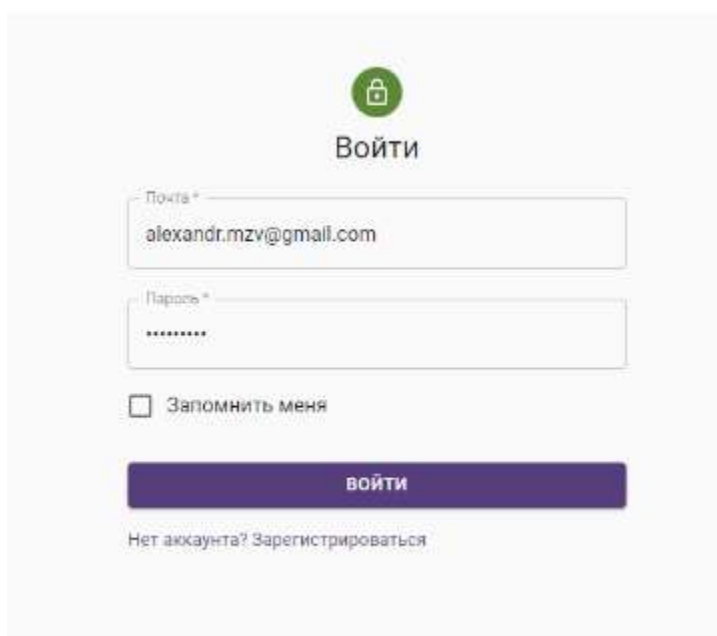
The screenshot shows a login form with the title "Войти" (Login) and a green lock icon. The form contains two input fields: "Почта *" (Email) and "Пароль *" (Password). The email field contains the text "alexander.v.morozov.06.09.1998@gmail.com". The password field contains a series of dots. Below the password field is a checkbox labeled "Запомнить меня" (Remember me). A purple button labeled "ВОЙТИ" (Login) is positioned below the checkbox. At the bottom of the form, there is a link that reads "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.10 – Функциональное тестирование методом анализа граничных условий, длинная почта успешно



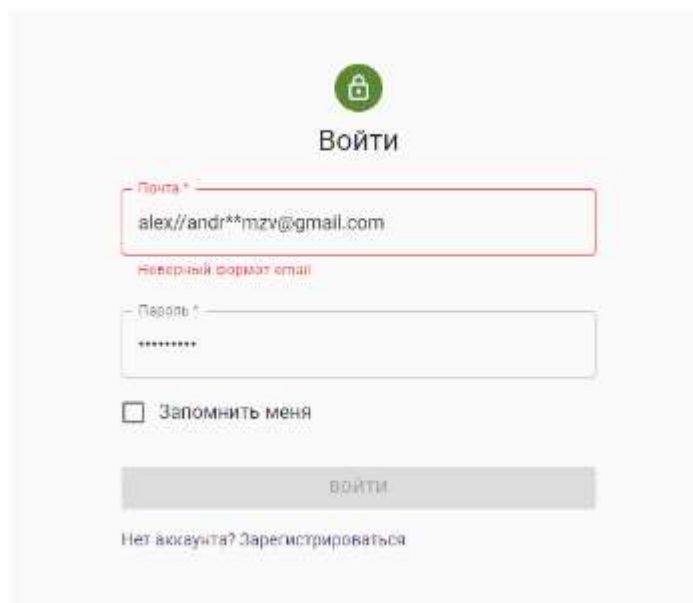
The screenshot shows a login page titled "Войти" (Login) with a green lock icon. The "Почта" (Email) field contains the long email address "alexander.valentinovich.morozov.06.09.1998@gmail". A red border highlights the email field, and a red error message "Слишком длинный email" (Email too long) is displayed below it. The "Пароль" (Password) field is filled with asterisks. There is an unchecked checkbox for "Запомнить меня" (Remember me) and a greyed-out "ВОЙТИ" (Login) button. At the bottom, there is a link "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.11 – Функциональное тестирование методом анализа граничных условий, слишком длинная почта



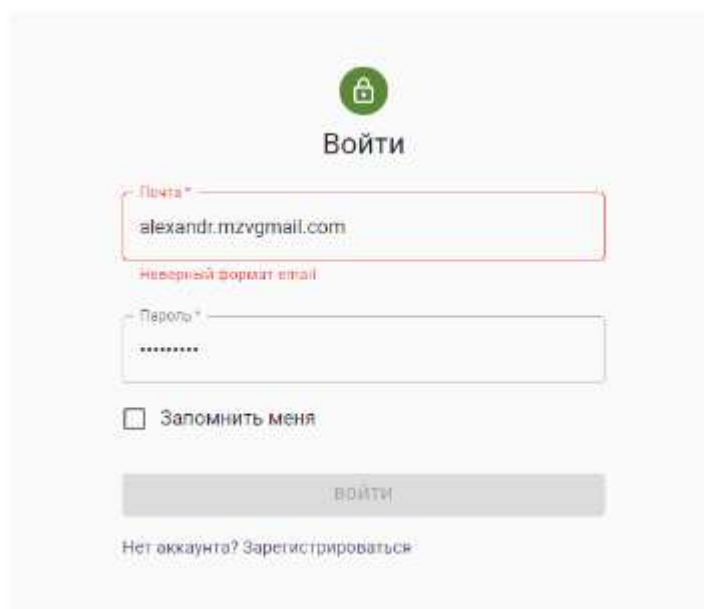
The screenshot shows the same login page as Figure В.11, but with a valid email address "alexandr.mzv@gmail.com" in the "Почта" field. The "Пароль" field is still masked with asterisks. The "Запомнить меня" checkbox remains unchecked. The "ВОЙТИ" button is now a solid purple color, indicating it is active. The "Нет аккаунта? Зарегистрироваться" link is still present at the bottom.

Рисунок В.12 – Функциональное тестирование методом анализа граничных условий, успешно



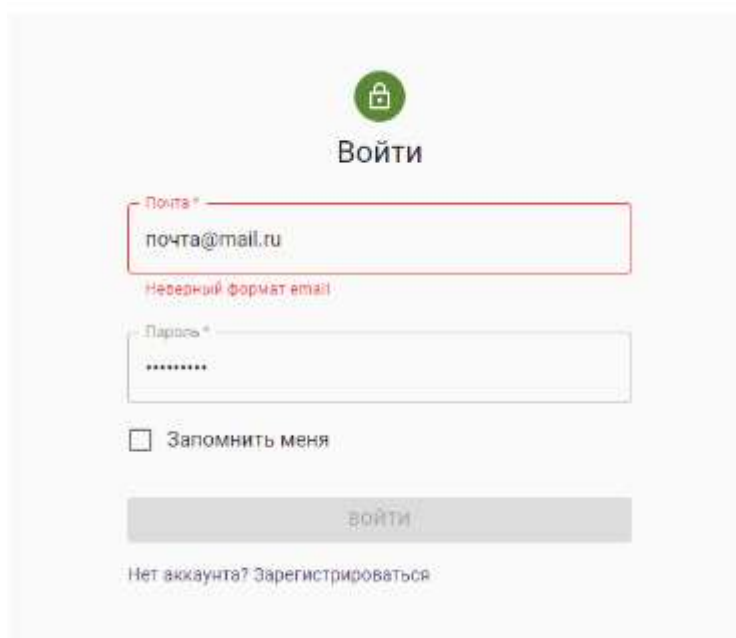
The screenshot shows a login page titled "Войти" (Login) with a green lock icon. The email field contains "alex//andr**mzv@gmail.com" and is highlighted with a red border. Below it, a red error message reads "Неверный формат email". The password field is masked with asterisks. There is a checkbox for "Запомнить меня" (Remember me) and a "ВОЙТИ" (Login) button. At the bottom, a link says "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.13 – Функциональное тестирование методом анализа граничных условий, недопустимые символы



The screenshot shows the same login page as Figure В.13, but with a different email address: "alexandr.mzv@gmail.com". The email field is still highlighted with a red border, and the red error message "Неверный формат email" is still present. The rest of the form, including the password field, checkboxes, and buttons, remains the same.

Рисунок В.14 – Функциональное тестирование методом анализа граничных условий, неверный формат



The image shows a login interface with the title "Войти" (Login) and a lock icon. It contains two input fields: "Почта*" (Email) and "Пароль*" (Password). The "Почта*" field has a red border and contains the text "почта@mail.ru". Below it is a red error message: "Неверный формат email". The "Пароль*" field contains masked characters "*****". There is a checkbox labeled "Запомнить меня" (Remember me) and a "ВОЙТИ" (Login) button. At the bottom, there is a link: "Нет аккаунта? Зарегистрироваться" (No account? Register).

Рисунок В.15 – Функциональное тестирование методом анализа граничных условий, русские символы