

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

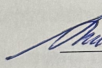
БАКАЛАВРСКАЯ РАБОТА

на тему

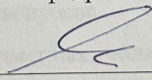
**НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ ЛОГИРОВАНИЯ
ФУНКЦИОНАЛА МОБИЛЬНЫХ КЛИЕНТОВ**

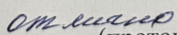
Студент  27.05.24 Мальков Илья Андреевич
(подпись, дата) (ФИО полностью)

Руководитель  27.05.24 А.Н. Токарев
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр  27.05.24 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.24 № 11)

Заведующий кафедрой  М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от 04.06.24 № 9)

Секретарь ГЭК  Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение высшего образования

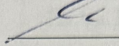
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«19» 04 2024г.

ЗАДАНИЕ НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Мальков Илья Андреевич гр. 20ВВП2 факультета вычислительной техники.

2. Тема работы Нагрузочное тестирование логирования функционала мобильных клиентов.

Тема утверждена приказом ПГУ № 405/6 от "19" 04 2024 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Токарев А. Н.

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

Разработать нагрузочный тест записи лога в базу данных для проверки производительности системы и выявления потенциальных проблемных мест. Использовать такие инструменты, как Supabase, Docker и Apache JMeter.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметных областей и инструментов нагрузочного тестирования систем. Выбрать технологии и определить взаимодействие компонентов.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области тестирования	32	19.04.2024- 22.04.2024	
2	Анализ предметной области нагрузочного тестирования	32	23.04.2024- 27.04.2024	
3	Анализ и выбор инструментов нагрузочного тестирования	65	28.04.2024- 02.05.2024	
4	Разработка системы нагрузочного тестирования на основе проведенного анализа	94	03.05.2024- 23.05.2024	
5	Проверка взаимодействия компонентов, реализация нагрузочного тестирования и анализ результатов	49	24.05.2024- 27.05.2024	
6	Оформление выпускной работы	32	28.05.2024- 30.05.2024	

Дата выдачи задания "19" "04" 2024 г.

Руководитель бакалаврской работы  19.04.2024 А.Н. Токарев
(подпись, дата)

Задание к исполнению принял студент  19.04.2024 И.А. Мальков
(подпись, дата)

Работу к защите допустить
Декан факультета  03.06.2024 Л.Р. Фионова
(подпись, дата)

Реферат

Бакалаврская работа содержит 61 лист, 13 рисунков, 1 таблица, 13 источников, 3 приложения.

ПРОИЗВОДИТЕЛЬНОСТЬ, НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ, ЛОГИРОВАНИЕ, JMeter, ТЕСТИРОВАНИЕ СИСТЕМЫ

Цель работы – изучение систем нагрузочного тестирования и разработка программной системы для реализации проверки производительности системы под нагрузкой.

Объектом разработки является нагрузочный тест, осуществляющий проверку производительности и отказоустойчивости системы логирования. Данная система позволяет выявить возможные узкие места, проблемы производительности или сбои, которые могут возникнуть при реальном использовании.

В результате проведенной работы разработан нагрузочный тест с использованием таких инструментов, как Apache JMeter, Supabase и Docker.

Результаты данной работы применены в научно-исследовательской работе ООО «Ай-Техно ВЦ» для организации рабочего процесса.

					ПГУ 1.090301.32.001 ПЗ				
Изм	Лист	№ докум.	Подпись	Дата	Нагрузочное тестирование логирования функционала мобильных клиентов Пояснительная записка	Литера	Лист	Листов	
Разраб.		Мальков И.А.		27.08.24					
Пров.		Токарев А.Н.		27.08.24			2	74	
Н. Контр.		Бычков А.С.		27.08.24		ФВТ, каф. ВТ гр. 20ВВП2			
Утв.		Митрохин М.А.		27.08.24					

Содержание

Перечень сокращений и обозначений.....	4
Введение.....	5
1 Анализ предметной области.....	7
1.1 Мобильное приложение	7
1.2 Тестирование системы.....	8
1.3 Виды тестирования	9
1.3.1 Функциональное тестирование	9
1.3.2 Модульное тестирование	9
1.3.3 Интеграционное тестирование	11
1.3.4 Регрессионное тестирование	11
1.3.5 Нефункциональное тестирование	11
1.4 Достоинства внедрения нагрузочного тестирования	13
1.5 Использование нагрузочного тестирования.....	14
1.6 Этапы тестирования.....	16
2 Выбор инструментов и решения.....	19
2.1 Инструменты тестирования	19
2.2 Сравнение инструментов тестирования	21
2.2.1 Сходства и различия инструментов тестирования	21
2.2.2 Достоинства и недостатки инструментов тестирования	22
2.3 Использование Apache JMeter	23
2.4 Платформы разработки	24
2.5 Сравнение платформ разработки.....	26
2.5.1 Достоинства и недостатки платформ разработки.....	26
2.5.2 Сходства и различия платформ разработки.	27

2.6	Использование Supabase.....	29
2.7	Docker	30
2.7.1	Преимущества Docker	31
2.7.2	Использование Docker.....	32
2.8	Базы данных.....	34
2.8.1	PostgreSQL	34
2.8.2	Аналоги PostgreSQL	35
2.8.3	Преимущества PostgreSQL.....	36
2.9	Взаимодействие компонентов	36
3	Реализация нагрузочного тестирования	38
3.1	Установка JDK.....	38
3.2	Установка Apache JMeter	39
3.3	Установка Docker	39
3.4	Установка Supabase.....	40
4	Нагрузочное тестирование	42
4.1	Логирование.....	42
4.2	Методика тестирования.....	42
4.2.1	Цель тестирования	44
4.2.2	Профиль нагрузки	44
4.2.3	Пользовательские сценарии.....	44
4.3	Запуск проверки	45
5	Результаты тестирования.....	48
5.1	Вывод.....	52
6	Охрана труда.....	53
	Заключение	56

Список использованных источников	58
Приложение А	60
Приложение А.1 – UML-диаграмма вариантов использования	61
Приложение А.2 – UML-диаграмма вариантов использования краткая	63
Приложение А.3 – UML-диаграмма последовательностей	64
Приложение А.4 – UML-диаграмма деятельности	65
Приложение Б	66
Приложение В.....	68

Перечень сокращений и обозначений

БД – база данных

ПО – программное обеспечение

СУБД – системы управления базами данных

ACID (англ. Atomicity, Consistency, Isolation, Durability) – атомарность, согласованность, изолированность, надежность

API (англ. Application Programming Interface) – передача состояния представления

CLI (англ. Command Line Interface) – интерфейс командной строки

DSL (англ. Domain-Specific Language) – предметно-ориентированный язык

FTP (англ. File Transfer Protocol) – протокол передачи файлов

GUI (англ. Graphical User Interface) – графический интерфейс пользователя

HTTP (англ. Hyper Text Transfer Protocol) – протокол передачи гипертекста

HTTPS (англ. Hyper Text Transfer Protocol Secure) – защищенный протокол передачи гипертекста

IP (англ. Internet Protocol) – интернет-протокол

JDK (англ. Java Development Kit) – набор инструментов разработки на языке Java

QA (англ. Quality Assurance) – обеспечение качества

SQL (англ. Structured Query Language) – язык структурированных запросов

TPS (англ. Ticks per Second) – количество тактов в секунду

UML (англ. Unified Modeling Language) – унифицированный язык моделирования

Введение

В современном мире мобильные технологии занимают одно из ведущих мест в повседневной жизни людей. Развитие мобильных приложений стало неотъемлемой частью информационных технологий, предоставляя пользователям доступ к различным сервисам и информации в любое время и в любом месте. Однако с ростом числа пользователей и увеличением функциональных возможностей приложений возрастает и нагрузка на серверные системы, что требует особого внимания к их производительности и надежности.

Одной из ключевых задач при разработке мобильных приложений является обеспечение стабильной работы системы при высоких нагрузках. Это достигается с помощью нагрузочного тестирования, которое позволяет выявить узкие места и потенциальные проблемы производительности, а также оценить способность системы обрабатывать большие объемы данных и большое количество одновременных запросов. В данной бакалаврской работе рассматривается тема нагрузочного тестирования логирования функционала мобильных клиентов.

Целью данной работы является изучение систем нагрузочного тестирования и разработка программной системы для реализации проверки производительности системы под нагрузкой. Объектом исследования является нагрузочный тест, осуществляющий проверку производительности и отказоустойчивости системы логирования. В рамках работы будут рассмотрены различные инструменты для нагрузочного тестирования, такие как Apache JMeter, Supabase и Docker, а также будут проведены тесты для оценки производительности системы.

Настоящая работа состоит из нескольких разделов, в которых последовательно рассматриваются этапы анализа предметной области, выбор инструментов и решений, реализация нагрузочного тестирования и анализ полученных результатов. В заключении будут подведены итоги проведенного исследования и даны рекомендации по дальнейшему развитию и оптимизации системы.

Таким образом, данная работа направлена на повышение качества и надежности мобильных приложений, что является важным аспектом в условиях постоянного роста числа пользователей и усложнения функционала программных продуктов.

					ПГУ 1.090301.32.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Мобильное приложение

Мобильное приложение представляет собой программный продукт, который создан для какой-либо мобильной платформы, например Android или IOS. Он предназначен для функционирования на мобильных устройствах пользователей и повышения удобства взаимодействия с этим устройством или какими-либо другими сервисами. Процесс разработки мобильного приложения состоит из таких этапов, как анализ продукта, составление документации, планирование, создание и утверждение дизайна, разработка, тестирование и выпуск.

Этап анализа предусматривает сбор требований от стороны заказчика, анализ финансовой составляющей, рисков и вариантов использования продукта. На данном шаге продумывается функционал приложения и стратегии его развития.

Следующий этап заключается в разработке и утверждении технического задания, которое описывает функциональные и нефункциональные требования. Также продумывается архитектура проекта и создается прототип дизайна.

Этап планирования определяет стратегию развития продукта, процесс его создания и какие для этого нужны ресурсы, например, какие требуются специалисты, сколько их должно быть и какие нужны ресурсы. Составляется расписание, взаимодействие и анализируются риски.

На этапе создания дизайна, учитываются прототипы дизайна, окончательно продумывается внешний вид продукта и составляются макеты, по которым в дальнейшем будет строиться процесс разработки. Здесь анализируются аналогичные продукты, продумываются цветовые решения и компоновка элементов, с которыми будет происходить взаимодействие.

Этап разработки заключается непосредственно в написании кода приложения по разработанному техническому заданию и макетам дизайна. Также этот этап предусматривает параллельное тестирование продукта, для

					ПГУ 1.090301.32.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

оптимизации и упрощения процесса разработки, а также ускорения выпуска проекта.

Тестирование необходимо для того, чтобы удостовериться в качестве разработанного приложения и соответствии его техническим требованиям и ожиданиям заказчика. Этот этап может проходить параллельно этапу разработки и ускорять весь процесс запуска программного продукта.

Финальный этап запуска предусматривает то, что программа разработана и функционирует согласно требованиям и ожиданиям. На этом шаге может продумываться рекламная компания для привлечения пользователей, если продукт является коммерческим. В ином случае происходит сдача программы заказчику, демонстрация ее функционирования и плана дальнейшей поддержки.

Работа над проектом может продолжаться и после его запуска, если предусмотрена дальнейшая его поддержка и продуман план развития. В приложения могут добавляться новые функции и совершенствоваться уже существующие.

1.2 Тестирование системы

Проверка программного обеспечения – неотъемлемая часть разработки, которая охватывает различные аспекты функциональности, производительности, безопасности и совместимости продукта. Процесс тестирования выполняется с использованием набора тестов, созданных на основе требований к продукту и ожидаемого поведения. Целью тестирования является не только выявление ошибок и дефектов, но и обеспечение высокого качества и надежности ПО перед его выпуском.

Функциональное тестирование предназначено для проверки соответствия функциональных возможностей программы ее спецификациям. Оно включает в себя тестирование пользовательского интерфейса, основных функций приложения, а также проверку соответствия требованиям безопасности и производительности. Каждый компонент ПО подвергается тщательному анализу, чтобы гарантировать его правильную работу во всех условиях использования.

					ПГУ 1.090301.32.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Безопасность является важным аспектом тестирования ПО, особенно в контексте угроз кибербезопасности. Тестирование безопасности содержит проверку наличия уязвимостей, анализ защиты данных, аутентификацию и авторизацию пользователей, а также другие меры защиты информации.

Тестирование производительности оценивает скорость и эффективность работы программы при различных нагрузках и условиях использования. В него входит измерение времени ответа, загрузки, выхода из строя, а также потребление ресурсов, а именно процессор, память и сеть.

Наконец, тестирование совместимости позволяет убедиться, что ПО работает корректно на различных платформах, устройствах и операционных системах.

Это важно для обеспечения максимальной доступности и удобства использования продукта для всех пользователей.

В целом, тестирование ПО является необходимым этапом разработки, который помогает обеспечить высокое качество, надежность и безопасность программного продукта перед его выходом на рынок [1, 2].

1.3 Виды тестирования

Существует широкий спектр методов тестирования системы.

1.3.1 Функциональное тестирование

Этот метод оценивает соответствие программного обеспечения установленным требованиям функционирования. Он сосредоточен на проверке того, что приложение выполняет ожидаемые функции без необходимости знания его внутренней структуры. В этом случае основное внимание уделяется основным функциям системы и ее доступности для пользователей.

1.3.2 Модульное тестирование

Это тестирование проводится на уровне отдельных компонентов или функций для проверки их соответствия требованиям. Обычно оно выполняется во время разработки приложения и помогает выявить ошибки на ранних этапах.

Модульные тесты как правило проводятся разработчиками перед интеграционным тестированием.

1.3.3 Интеграционное тестирование

Этот метод состоит из тестирования всех компонентов приложения в единой системе. В данном случае основное внимание уделяется оценке соответствия всей системы установленным функциональным требованиям. Оно проверяет взаимодействие различных модулей при их объединении.

1.3.4 Регрессионное тестирование

Этот метод подразумевает повторное выполнение функциональных и нефункциональных тестов после внесения изменений в программное обеспечение, чтобы убедиться в способности ПО продолжать выполнять те же функции, что и ранее. Основная цель метода гарантировать, что изменения не негативно повлияли на существующий функционал.

1.3.5 Нефункциональное тестирование

Этот вид тестирования оценивает различные нефункциональные аспекты приложения, такие как производительность, надежность, удобство использования и другие. Он фокусируется не на том, что делает продукт, а на том, насколько качественно он это делает. Обычно нефункциональное тестирование проводится после функционального тестирования.

Различные типы нефункционального тестирования включают в себя тестирование производительности, надежности, безопасности, масштабируемости, совместимости, нагрузочное тестирование и другие.

Нагрузочное тестирование является одним из важных видов нефункционального тестирования. Оно направлено на оценку способности приложения обрабатывать большие объемы данных и одновременно обслуживать большое количество пользователей. В процессе нагрузочного тестирования создается высокая нагрузка на систему, чтобы оценить ее производительность и стабильность при различных условиях использования. Это помогает выявить узкие места и проблемы производительности, а также оптимизировать работу приложения для обеспечения максимальной эффективности и удовлетворения потребностей пользователей.

Все упомянутые виды тестирования могут быть выполнены как вручную, так и с помощью автоматизированных средств. Это означает, что тестирование может быть как ручным, когда каждый шаг выполняется вручную тестировщиком, так и автоматизированным, когда задачи выполняются с использованием специальных программных средств.

Выбор конкретного типа тестирования зависит от типа компании, который играет ключевую роль в определении методов тестирования. Крупные компании могут предпочитать использовать более масштабные и сложные подходы к тестированию, в то время как для небольших компаний более простые и быстрые методы могут быть более подходящими. Также выбор зависит от времени тестирования, по причине того, что тестировщики могут планировать использовать множество типов тестирования, но ограниченное время зачастую означает, что будут выбраны только наиболее эффективные методы. Временные ограничения могут определять, какие типы тестирования будут наиболее приоритетными для конкретного проекта. Еще на планирование процесса тестирования влияет наличие ресурсов, таких как высокотехнологичное программное обеспечение. При наличии достаточного количества ресурсов можно позволить использовать более сложные методы тестирования. Уровень опыта тестировщика также важен при выборе методов тестирования. Некоторые методы могут требовать специализированных знаний и навыков, которые не каждый тестировщик может обеспечить. Важный фактор выбора методов эффективного тестирования — это наличие достаточного бюджета. Распределение бюджета на тестирование также влияет на выбор методов тестирования. Более высокие бюджеты могут позволить использовать более дорогостоящие программные решения для тестирования, в то время как более ограниченные бюджеты могут требовать более экономичных подходов [3-6].

1.4 Достоинства внедрения нагрузочного тестирования

Внедрение нагрузочного тестирования представляет собой важный этап в разработке программного обеспечения, обладающий рядом значительных достоинств, таких как оптимизация производительности, гарантия масштабируемости, повышение надежности, экономия времени и ресурсов, улучшение пользовательского опыта.

Нагрузочное тестирование не только помогает выявить узкие места в системе, но и предоставляет ценные данные для оптимизации производительности. Путем анализа результатов тестирования команды разработки могут идентифицировать участки кода или запросы к базе данных, которые замедляют работу системы. Дополнительно, используя результаты нагрузочного тестирования, можно определить оптимальные параметры конфигурации серверов или использовать кэширование данных для улучшения отклика системы.

Нагрузочное тестирование предоставляет информацию о том, как система ведет себя при увеличении нагрузки. Это помогает предвидеть, как система будет реагировать на увеличение количества пользователей или объема данных. Зная пределы масштабируемости системы, разработчики могут внести соответствующие изменения в архитектуру или инфраструктуру, чтобы обеспечить ее горизонтальное масштабирование и устойчивость к нагрузкам.

Нагрузочное тестирование помогает выявить скрытые проблемы и уязвимости, которые могут проявиться только при высоких нагрузках. Используя эти данные, разработчики могут устранить проблемы производительности и бутылочные горлышки, что повышает надежность работы системы в реальных условиях эксплуатации. Это также способствует снижению риска сбоев и потери данных, что критически важно для бизнеса.

Предотвращение проблем на ранних этапах разработки с помощью нагрузочного тестирования позволяет избежать необходимости значительных

изменений в коде или архитектуре в будущем. Это сокращает время разработки и улучшает эффективность команды разработчиков. Также, благодаря уменьшению вероятности сбоев в продуктивной среде, сокращаются и затраты на поддержку и обслуживание системы.

Стабильная работа системы при высоких нагрузках не только укрепляет уверенность пользователей в продукте, но и создает положительное впечатление о компании. Улучшение пользовательского опыта способствует удержанию клиентов, повышает лояльность и способствует увеличению прибыли. Надежность и производительность системы становятся ключевыми факторами, определяющими конкурентоспособность компании на рынке.

В целом, внедрение нагрузочного тестирования позволяет повысить качество, производительность и надежность программного обеспечения, что является ключевым для успешного функционирования бизнеса в современном цифровом мире [7, 8].

1.5 Использование нагрузочного тестирования

Нагрузочное тестирование является видом тестирования программного обеспечения, направленным на оценку производительности и устойчивости системы под нагрузкой, близкой к максимальной или предполагаемой реальной. Целью нагрузочного тестирования является определение того, как система будет себя вести при высоких нагрузках, чтобы выявить возможные узкие места, проблемы производительности или сбои, которые могут возникнуть при реальном использовании. В процессе нагрузочного тестирования создается искусственная нагрузка, имитирующая действия реальных пользователей или приложений, чтобы проверить реакцию системы на различные уровни нагрузки. Указанный вид тестирования позволяет выявить проблемы масштабируемости, оптимизировать ресурсы и обеспечить стабильную работу системы даже при значительном увеличении числа пользователей или объема данных.

Нагрузочное тестирование используют разработчики программного обеспечения. Оно помогает разработчикам и инженерам по тестированию оценить производительность и масштабируемость разрабатываемой системы, что позволяет выявить потенциальные проблемы и улучшить архитектуру приложения на ранних этапах разработки.

Также оно может быть полезно бизнес-аналитикам и менеджерам проектов. Результаты нагрузочного тестирования предоставляют важную информацию для принятия решений на уровне бизнеса. Они позволяют оценить, насколько система готова к реальной нагрузке, какие ресурсы необходимы для обеспечения требуемого уровня производительности, и какие изменения могут потребоваться для достижения желаемых целей.

IT-специалисты по обеспечению качества используют нагрузочное тестирование для общей стратегии обеспечения качества (QA), для проверки производительности системы и выявления проблем, которые могут возникнуть при работе приложения в реальных условиях.

Применение нагрузочного тестирования необходимо для оценки производительности. Оно позволяет оценить производительность системы при различных нагрузках. Это позволяет определить максимальное количество пользователей или операций, которые система способна обработать без ухудшения производительности.

Нагрузочное тестирование применяется для выявления потенциальных проблемных мест программного обеспечения. Проведение нагрузочного тестирования помогает выявить узкие места и бутылочные горлышки в системе, такие как медленные запросы к базе данных или неэффективные алгоритмы. Это позволяет разработчикам оптимизировать систему и повысить её производительность.

Применение нагрузочного тестирования — это гарантия стабильной работы. Оно помогает обеспечить стабильную работу приложения при реальных условиях эксплуатации. Это важно для предотвращения сбоев и неполадок, которые могут возникнуть при работе системы под нагрузкой.

					ПГУ 1.090301.32.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

UML диаграмма вариантов использования нагрузочных тестов представлена на рисунке 1 [8, 9].

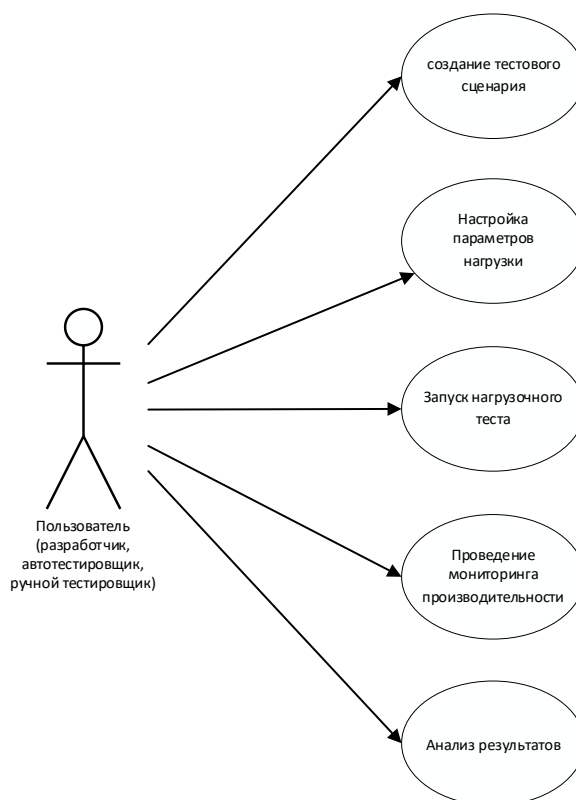


Рисунок 1 - Варианты использования нагрузочного теста

1.6 Этапы тестирования

Нагрузочное тестирование – это методика тестирования программного обеспечения, направленная на оценку производительности и устойчивости системы при реальной нагрузке. Основная цель нагрузочного тестирования – определить границы и возможности системы, выявить узкие места и бутылочные горлышки, а также обеспечить стабильную работу приложения при различных условиях эксплуатации.

Вот основные этапы этого процесса:

– планирование. В этом этапе определяются не только цели тестирования, но и ожидаемые результаты. Также проводится анализ рисков, связанных с нагрузочным тестированием, и разрабатывается стратегия их управления.

Решается, какие метрики производительности будут измеряться и как они будут интерпретироваться. Кроме этого, на этом этапе рассматриваются аспекты масштабируемости системы, такие как возможность ее расширения под высокие нагрузки в будущем;

– подготовка. Этот этап состоит из создания тестовых сценариев, которые моделируют реальные сценарии использования приложения. Важно убедиться, что тестовые данные соответствуют разнообразным ситуациям, с которыми может столкнуться система в реальном мире. Также на этом этапе выполняется настройка тестовой среды, включая конфигурацию серверов, баз данных и других компонентов системы;

– выполнение. Запуск тестовых сценариев и мониторинг производительности системы. Важно иметь возможность масштабировать нагрузку в зависимости от того, как система реагирует на различные уровни нагрузки. Подобные тесты могут быть выполнены как на тестовых средах, так и на производственных системах с использованием специальных инструментов для контроля и изоляции нагрузки;

– анализ. После выполнения тестов анализируются собранные данные для выявления слабых мест системы и идентификации возможных узких мест. Это может включать анализ логов, мониторинга производительности и других данных, собранных во время выполнения тестов. Анализ результатов помогает определить, какие изменения или улучшения могут быть внесены в систему для повышения ее производительности и устойчивости;

– отчетность. Подготовка отчета о результатах тестирования, выявленных проблемах и рекомендациях по их устранению. Отчет представляется заинтересованным сторонам, таким как руководство компании, команда разработки или заказчики, для принятия решений о дальнейших шагах по улучшению производительности и качества системы.

Этапы нагрузочного тестирования представлена на рисунке 2 [7, 10].

					ПГУ 1.090301.32.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

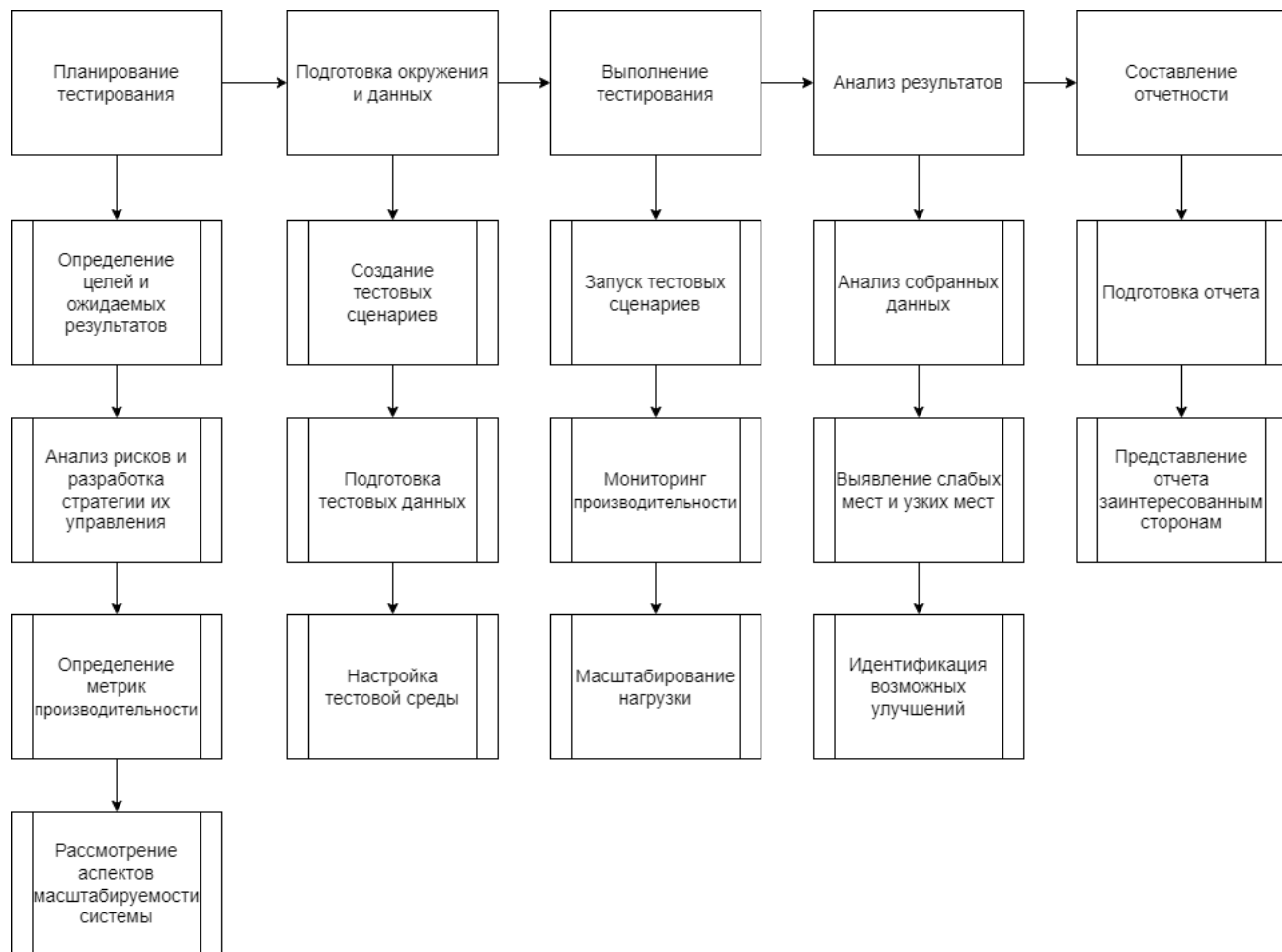


Рисунок 2 - Этапы нагрузочного тестирования

2 Выбор инструментов и решения

Решение любой задачи требует предварительного анализа, в частности технической составляющей. Этот анализ необходим, ввиду большого разнообразия технологий, даже для решения одной и той же задачи. Поэтому для наиболее правильного выбора, требуется изучить возможности каждого инструмента, его стек технологий и условия использования. Это нужно для поддержания актуальности своих разработок и эффективности процесса поддержки продукта.

Немаловажным этапом разработки и обслуживания проекта является процесс его тестирования, в частности нагрузочного. Это довольно ответственный процесс, позволяющий поддерживать приложение в работоспособном состоянии, отвечающем требованиям производительности. Существует множество инструментов, позволяющих провести данное тестирование. Все они по-своему хороши и подходят для осуществления поставленной задачи, но также имеют и недостатки.

2.1 Инструменты тестирования

Инструмент тестирования является основным программным обеспечением для проведения нагрузочного тестирования приложения. Его выбор зависит от многих факторов, таких как возможности и характеристики исследуемого приложения, цели тестирования и условия использования. Первые два фактора оказывают влияние на план тестирования и определяют его процесс. Руководствуясь этими данными, можно принять решение об инструменте, исходя из его возможностей. Третий же фактор влияет на субъект тестирования, так как от него зависит финансовая составляющая проекта. Поэтому на факт необходимости оплаты инструментов следует обращать внимание еще на этапе планирования расходов.

Также к важным особенностям при выборе инструмента тестирования можно отнести его масштабируемость и возможность интеграции с другими

					ПГУ 1.090301.32.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

программами. На это необходимо обращать внимание, так как масштабируемость подразумевает способность адаптировать инструмент и сами тесты под изменение исследуемого приложения, а возможность интеграции позволяет построить взаимодействие с другими компонентами системы.

Далее будут рассмотрены такие инструменты нагрузочного тестирования, как Apache JMeter, Gatling, LoadRunner и BlazeMeter.

JMeter представляет собой ПО, предназначенное для проведения тестов надежности и производительности приложений. В возможности его функционала входит создание тестов, их выполнение и просмотр результатов после окончания процесса тестирования. Также он обладает удобным графическим интерфейсом, что позволяет повысить комфорт взаимодействия тестировщика с программой. Явным преимуществом JMeter является его поддержка разных протоколов данных, например, HTTP, HTTPS и FTP, и то, что он способен масштабироваться под нужды продукта и взаимодействовать с другим ПО. Этот инструмент распространяется бесплатно и имеет открытый исходный код, что тоже позволяет сделать выбор в его пользу.

Следующий рассматриваемый инструмент, Gatling, также является свободно распространяемым ПО с открытым исходным кодом. Он обладает возможностью создания сценариев тестирования с помощью специального языка описания тестов нагрузки DSL. Также у него есть особенность в том, что он способствует эффективности выполнения тестов тем, что у него асинхронная архитектура, что позволяет справляться с высокой нагрузкой.

Третий инструмент, называемый LoadRunner, также предоставляет много возможностей для моделирования нагрузки на испытуемый программный продукт. Он может служить для проверки производительности протоколов передачи данных, клиент-серверных и веб-приложений. LoadRunner является универсальным инструментом для тестирования ПО различного характера, так как обладает большим стеком технологий, решающих разные задачи. Этот инструмент, в отличие от предыдущих коммерческий и распространяется на платных условиях.

Последний рассматриваемый инструмент называется BlazeMeter. Он полностью поддерживает интеграцию с JMeter и доступен как веб-сервис. Его довольно просто освоить, из-за того, что у него удобный и простой графический интерфейс. Этот сервис предоставляет много полезных возможностей, например гибкость настроек тестовых сценариев и генерацию интерактивных и детализированных отчетов. Все перечисленные достоинства позволяют данному инструменту эффективно тестировать ПО, имитируя реальную нагрузку на приложение [11].

2.2 Сравнение инструментов тестирования

2.2.1 Сходства и различия инструментов тестирования

Все перечисленные инструменты обладают хорошим набором функций для решения такой задачи, как нагрузочное тестирование. Однако каждый имеет свои особенности, которые отличают его от остальных. К их сходствам можно отнести то, что все эти инструменты предназначены для выполнения схожих операций, а именно проверки надежности и производительности приложений и последующей оценки результатов тестирования. Также сюда можно отнести то, что эти инструменты предоставляют возможность интеграции с другими программными продуктами для повышения эффективности процесса тестирования приложений. Масштабируемостью также располагает каждый из этих инструментов, позволяя им адаптироваться под изменения конфигурации и функционала исследуемого ПО.

В свою очередь необходимо анализировать также различия инструментов для того, чтобы правильно выбрать требуемый инструмент корректно провести нагрузочное тестирование. Отличительными чертами этих приложений является, например, наличие пользовательского интерфейса, потому что Gatling им не обладает, в отличие от трех других. Они в свою очередь предоставляют GUI, как Apache JMeter, или веб-интерфейс, как LoadRunner и BlazeMeter. Также к отличиям инструментов можно отнести тот факт, что все они используют интерфейс для создания и редактирования тестовых сценариев, кроме Gatling. В

нем используется язык описания сценариев DSL, что позволяет создавать и изменять сценарии при помощи IDE. Другим немаловажным фактором является вариант распространения инструмента, где JMeter и Gatling являются свободно распространяемым ПО, а LoadRunner и BlazeMeter предоставляются на платных условиях.

2.2.2 Достоинства и недостатки инструментов тестирования

Достоинствами JMeter выступают поддержка широкого спектра протоколов, открытый исходный код, гибкость настройки сценариев, а также возможность анализа и формирования отчетов.

К недостаткам можно отнести сложность его графического интерфейса, или выполнение тестов только на уровне протокола.

Положительными чертами Gatling являются возможность интеграции с программами мониторинга, его свободное распространение и использование специального языка описания сценариев.

Недостатками же являются отсутствие у него графического интерфейса, необходимость написания скриптов, а также поддержка относительно небольшого количества протоколов.

У LoadRunner достоинствами выступают поддержка большого количества протоколов, возможность интеграции с другими инструментами и его универсальность в плане объекта тестирования.

К его отрицательным чертам можно отнести то, что это коммерческий, платно распространяемый продукт, а также возможную сложность его установки и настройки.

К положительным чертам BlazeMeter относятся разнообразность тестовых сценариев нагрузки, возможность составления отчетов и масштабируемости к изменениям тестируемого ПО.

Выбор в пользу Apache JMeter обосновывается его универсальностью, удобным графическим интерфейсом (рисунок 3), гибкостью настроек, богатым стеком технологий, а также открытым исходным кодом. Это, и то, что он имеет

большое сообщество пользователей, делает его доступным и привлекательным для широкого круга пользователей [12].

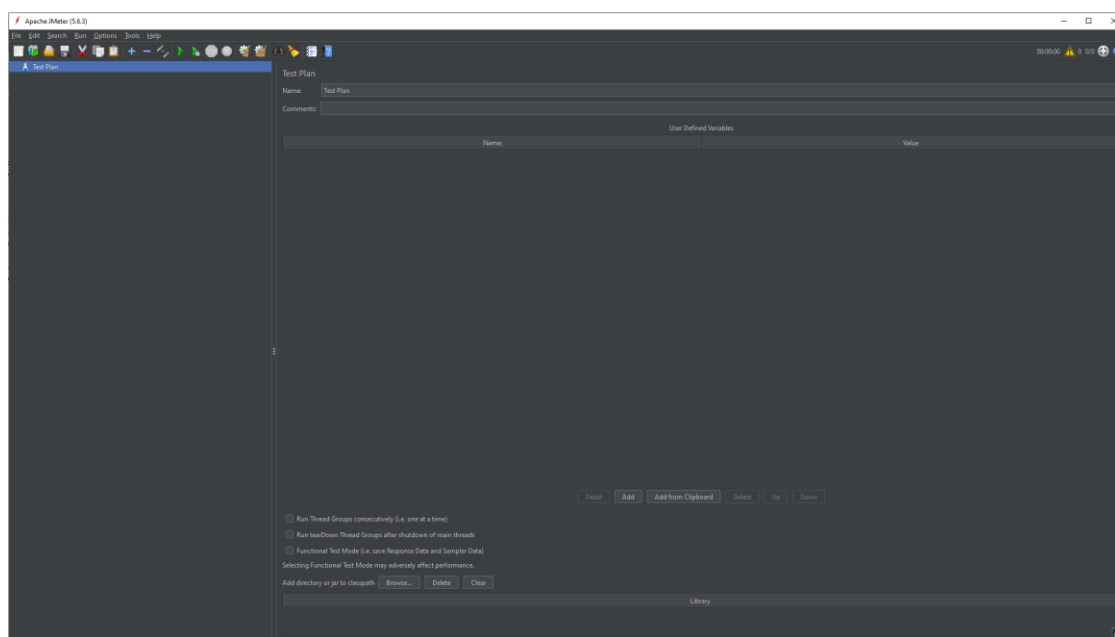


Рисунок 3 - Интерфейс JMeter

2.3 Использование Apache JMeter

Apache JMeter – это инструмент с открытым исходным кодом для проверки производительности и нагрузочного тестирования приложений. Он предоставляет возможность создания и запуска различных видов тестовых сценариев для оценки производительности серверов, сетей и приложений.

Основные возможности использования Apache JMeter включают в себя создание сценарии нагрузки, которые моделируют поведение реальных пользователей при использовании приложения. Они включают в себя отправку запросов на сервер, симуляцию действий пользователей, таких как переходы по ссылкам, заполнение форм и отправка данных. Также, используя этот инструмент, можно анализировать производительность программных продуктов, измерять время отклика сервера, задержки и пропускную способность сети. Это помогает выявить потенциально проблемные места в приложении и оптимизировать его работу. Еще JMeter позволяет проводить тестирование

надежности и безопасности, проверяя, как приложение работает в условиях большой нагрузки или имеет ли оно какие-либо уязвимости.

Для того, чтобы провести нагрузочное тестирование приложения необходимо подготовка тестового плана для определения целей тестирования, выбора метрик и сценариев нагрузки. Затем производится настройка этого плана в JMeter, что включает настройку потоков пользователей, параметров генерации нагрузки и определение метрик производительности для мониторинга.

После составления плана тестирования можно переходить к запуску тестового плана на испытуемом сервере для генерации нагрузки на приложение.

Мониторинг производительности приложения доступен в режиме реального времени, а анализ результатов тестирования для выявления проблем производительности и улучшения работы приложения выполняется в заключении [13].

2.4 Платформы разработки

Платформы разработки – это облачные сервисы, предоставляющие разработчикам инструменты и ресурсы для создания, развертывания и управления приложениями. Они обеспечивают всю инфраструктуру, необходимую для разработки и эксплуатации приложений, включая базы данных, систему аутентификации, хранилище файлов и другие функциональные возможности. Платформы разработки упрощают создание приложений, предоставляя готовые инструменты и сервисы и облегчают процесс поддержки.

Примерами платформ разработки могут выступать Supabase, Firebase, AWS Amplify и Hasura.

Supabase – это открытая, основанная на PostgreSQL платформа, предоставляющая полный стек инструментов для разработки веб-приложений и базу данных. Эта платформа свободно распространяется на бесплатных условиях и предоставляет систему аутентификации, хранения файлов и обновление контента приложения в режиме реального времени. Она предлагает простой интерфейс, приведенный в пример на рисунке 4, и гибкую настройку,

которые

делают

ее привлекательным выбором для разработчиков.

Firebase – это облачный сервис от компании Google, обладающая богатым набором инструментов разработки приложений для разных платформ. Firebase содержит такие системы, как база данных, аутентификация, хранилище файлов, облачные функции, аналитика и управление уведомлениями. Firebase известен своей простотой использования и широким спектром функций.

AWS Amplify – это сервис от Amazon Web Services (AWS), предоставляющий инструменты для создания и развертывания приложений. В Amplify также входят такие же сервисы, как аутентификация, хранение файлов, GraphQL API, управление уведомлениями и многое другое, что обеспечивает его хорошей гибкостью и масштабируемостью.

Hasura – это инструмент для разработки и развертывания GraphQL API, который автоматически создает API на основе существующей схемы базы данных PostgreSQL. Эта платформа обеспечивает быструю и удобную разработку API, из-за чего она является привлекательным выбором для разработки проектов, использующих в качестве базы данных PostgreSQL.

Все эти платформы предоставляют разработчикам разнообразные инструменты и сервисы для создания и развертывания приложений, каждая из которых имеет свои особенности и преимущества.

					ПГУ 1.090301.32.001 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

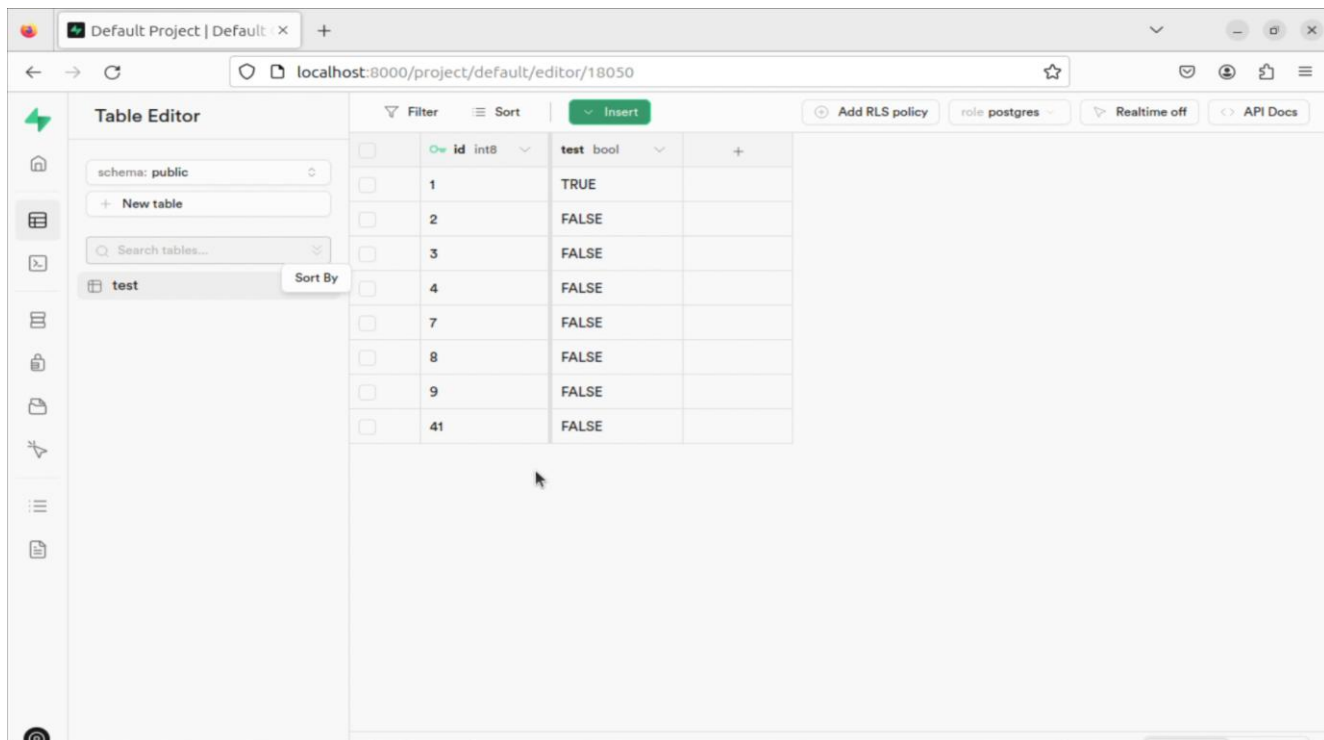


Рисунок 4 – Интерфейс Supabase в режиме редактирования таблицы

2.5 Сравнение платформ разработки

2.5.1 Достоинства и недостатки платформ разработки.

При выборе платформы из Supabase, Firebase, AWS Amplify и Hasura следует учитывать различные аспекты, такие как функциональность, простота использования, гибкость, стоимость и масштабируемость, а значит, нужно рассмотреть достоинства и недостатки каждого сервиса.

К преимуществам Supabase можно отнести то, что она построена поверх базы данных PostgreSQL, что обеспечивает систему удобными возможностями хранения и управления данными. Также она предоставляет широкий набор инструментов для разработки приложений, включая аутентификацию, хранение файлов, веб-хостинг и другие. Быстрому развитию платформы способствует открытый исходный код и активное сообщество разработчиков, которое поддерживает и улучшает свой продукт.

К недостаткам же можно отнести лишь меньшую популярность и менее широкое распространение по сравнению с Firebase и AWS Amplify.

В свою очередь Firebase располагает такими достоинствами, как широкий спектр сервисов и инструментов для разработки и поддержки приложений, включая базу данных, аутентификацию и облачное хранилище. Немаловажно то, что она поддерживает интеграцию с другими сервисами Google и обладает простотой использования, которая делает его привлекательным для быстрой разработки приложений.

Недостатки Firebase включают ограниченные возможности настройки и меньшую гибкость по сравнению с открытыми платформами, такими как Supabase.

AWS Amplify имеет такие преимущества, как поддержка интеграции с экосистемой AWS, предоставляющей доступ к разным облачным сервисам и инструментам разработки и развертывания и обслуживания приложений. Хорошие возможности настройки и масштабируемость, облегчают поддержку больших проектов и помогают командам разработки.

Отрицательными чертами данной платформы являются сложность использования и настройки, и платный характер распространения.

Преимущества Hasura включают в себя автоматическое создание GraphQL API на основе существующей схемы базы данных PostgreSQL, что делает разработку и работу с данными более удобной и эффективной. Также сюда можно отнести и простоту управления информацией.

В качестве недостатка можно назвать ограниченные возможности по сравнению с другими платформами разработки приложений, такими как Firebase и AWS Amplify.

2.5.2 Сходства и различия платформ разработки.

Облачные сервисы таких платформ, как Supabase, Firebase, AWS Amplify и Hasura, предназначены для разработки и развертывания приложений, и каждая из них обладает своими особенностями, преимуществами и недостатками.

Сходствами может выступать то, что все четыре платформы предназначены для разработки приложений и предоставляют инструменты для работы с данными, аутентификации, хранения файлов и другими аспектами разработки.

Они все предлагают облачные сервисы, которые обеспечивают инфраструктуру для развертывания приложений в облаке и управления ими, а также обладают необходимой степенью гибкости и расширяемости, позволяющей разработчикам настраивать и расширять функциональность своих приложений. Возможности управления пользователями, включая аутентификацию, авторизацию и управление доступом также предоставляют все четыре сервиса.

Отличительными чертами является то, что Supabase основан на PostgreSQL и предоставляет инструменты на основе SQL, Firebase основан на NoSQL-технологиях и использует свои уникальные сервисы, такие как Realtime Database и Firestore. В свою очередь AWS Amplify предоставляет доступ к широкому спектру облачных сервисов Amazon, а Hasura автоматически создает GraphQL API на основе существующей схемы PostgreSQL. Также можно сказать, что Firebase и AWS Amplify известны своей простотой использования и широким выбором инструментов для разработки приложений, в свою очередь Supabase и Hasura также довольно просты, но могут потребовать больше настройки и конфигурации в сравнении с первыми двумя платформами. Еще следует упомянуть то, что по сравнению с остальными сервисами, Firebase может показаться менее масштабируемой из-за ограничений других своих продуктов. Важно также то, что стоимость использования каждой платформы может существенно различаться. Firebase и Supabase предлагают бесплатные тарифные планы с возможностью перехода на платные для расширения своего функционала, а AWS Amplify и Hasura могут иметь более сложные ценовые модели, основанные на использовании облачных ресурсов.

В целом, каждая из этих платформ имеет свои сильные и слабые стороны, и выбор конкретной платформы зависит от требований проекта, уровня опыта разработчиков и других факторов. Выбор в пользу Supabase вместо других платформ обосновывается ее достоинствами в виде использования базы данных PostgreSQL, большого выбора инструментов разработки приложений, активного сообщества разработчиков, а также открытого исходного кода и масштабируемости данной платформы.

2.6 Использование Supabase

Supabase представляет собой облачную платформу, которая предоставляет инструменты для разработки и управления приложениями и базами данных. Хотя Supabase в первую очередь ориентирован на создание приложений и хранение данных, его также можно использовать для некоторых вариантов нагрузочного тестирования.

Один из способов использования Supabase для нагрузочного тестирования заключается в проверке производительности приложения при обращении к базе данных. Можно создать тестовые сценарии, которые имитируют большое количество запросов к базе данных и оценить, как разрабатываемое приложение обрабатывает эту нагрузку.

Для этого можно использовать инструменты тестирования производительности, такие как, например, уже выбранный Apache JMeter, чтобы создать тестовые сценарии. Они будут генерировать большое количество запросов к базе данных Supabase, после чего можно проанализировать результаты тестирования для оценки производительности программы и определения, сможет ли она эффективно обрабатывать ожидаемую нагрузку.

Однако стоит отметить, что Supabase в первую очередь предназначен для хранения и обработки данных, а не для тестирования производительности. Поэтому его использование для нагрузочного тестирования может быть ограничено, и для более полного и точного тестирования производительности приложения могут потребоваться специализированные инструменты и техники.

В контексте нагрузочного тестирования, Supabase может использоваться для создания тестовых сред и управления инфраструктурой приложений. Эта платформа также может интегрироваться с Docker для развертывания приложений внутри изолированных контейнеров. Разработчики могут использовать Docker для локального развертывания и тестирования приложений, а затем легко масштабировать и развертывать их на производственных серверах с помощью Docker-образов.

Таким образом, использование Supabase в сочетании с Docker обеспечивает гибкость, управляемость и изолированность окружения как для нагрузочного тестирования, так и для развертывания приложений, что упрощает процесс разработки и тестирования исследуемого ПО.

2.7 Docker

Supabase рассчитана на работу с системой Docker, так как он может выступать в качестве одного из рекомендуемых способов установки. Использование Docker входит в требования установки Supabase, предоставляя более гибкий и удобный способ развертывания и управления сервисами приложения. Docker обеспечивает изолированную среду из необходимых зависимостей и конфигураций для работы платформы. Он делает процесс установки и настройки более простым и эффективным, поэтому выбор Docker обусловлен как техническими требованиями самого Supabase, так и удобством его использования для развертывания и управления приложениями внутри контейнеров.

Docker – это платформа для разработки, доставки и запуска приложений в контейнерах. Эти контейнеры обеспечивают стандартизированную среду для выполнения приложений, включая все необходимые зависимости, библиотеки и конфигурации, что делает их переносимыми и воспроизводимыми на различных средах.

Принцип работы Docker основан на контейнеризации, которая позволяет изолировать приложения и их зависимости от основной операционной системы и других приложений. Каждый контейнер работает в своем собственном изолированном пространстве, где у него есть своя собственная файловая система, сеть и процессы, что обеспечивает высокий уровень безопасности и управляемости.

Основные компоненты Docker включают Docker Engine, Docker CLI, Docker Hub и Docker Compose. Docker Engine – это основной компонент, который управляет контейнерами и обеспечивает их выполнение на хост-системе. Docker

					ПГУ 1.090301.32.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

CLI предоставляет пользовательский интерфейс для взаимодействия с Docker Engine через командную строку. Docker Hub является облачным репозиторием образов контейнеров, где можно хранить и загружать образы, а Docker Compose определяет и управляет многоконтейнерными приложениями в Docker с помощью файла конфигурации.

2.7.1 Преимущества Docker

Использование Docker обосновывается рядом его преимуществ, так как он один из самых популярных инструментов для контейнеризации, и имеет огромное сообщество разработчиков и пользователей. Также он предоставляет различные интеграции с другими инструментами и сервисами и известен своей простотой в использовании и интуитивно понятным интерфейсом, показанном на рисунке 5. Он предоставляет простые команды командной строки для создания, запуска, остановки и удаления контейнеров, что делает его удобным в использовании. Еще стоит отметить, что Docker поддерживает различные операционные системы, включая Linux, MacOS и Windows для обеспечения универсальности и возможности развертывания контейнеров на различных платформах без изменения кода или конфигурации системы. Он имеет богатую экосистему инструментов и сервисов, таких как Docker Compose для управления многоконтейнерными приложениями, Docker Swarm для оркестрации контейнеров и Docker Hub для управления образами контейнеров. Это делает его практически незаменимым инструментом для разработки, тестирования и развертывания приложений.

Хотя аналоги Docker, такие как Podman, Kubernetes и rkt, также предлагают определенные преимущества, например, бездемонную архитектуру и более высокий уровень автоматизации, он все еще остается предпочтительным выбором благодаря своей простоте, распространению и обширной экосистеме.

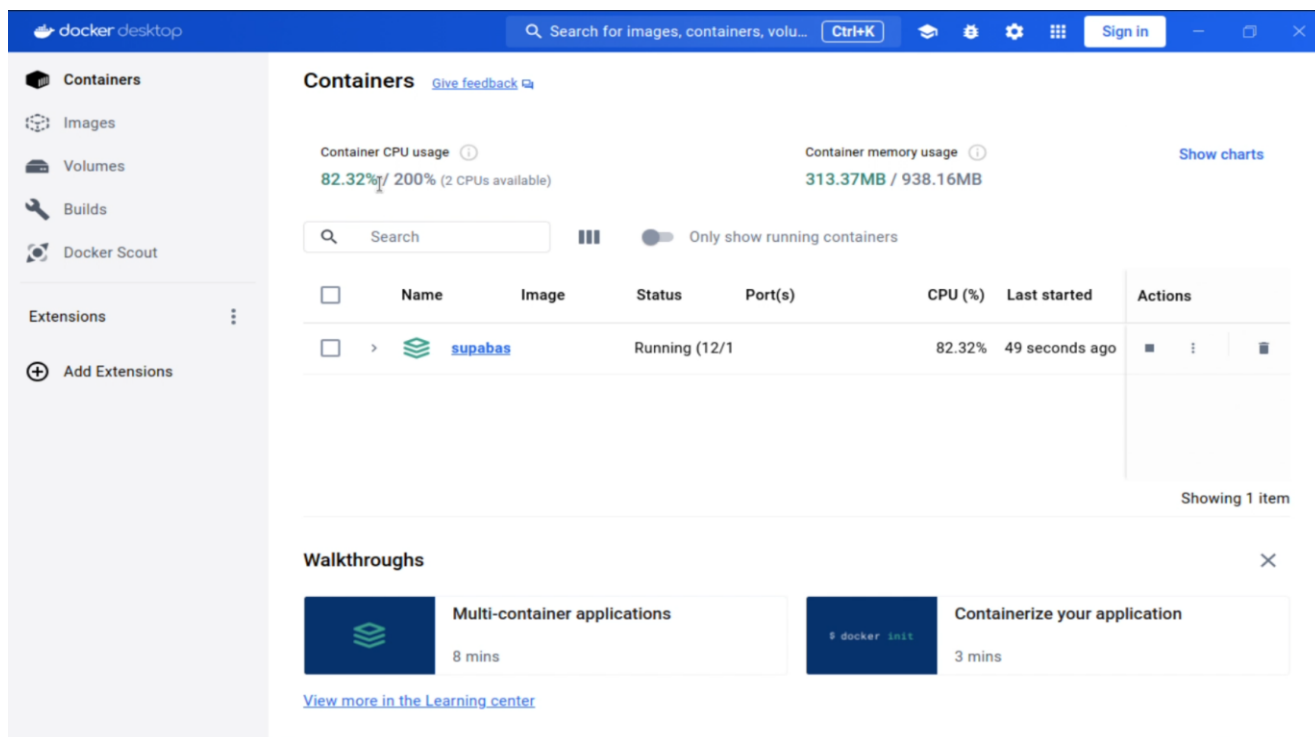


Рисунок 5 – интерфейс Docker

2.7.2 Использование Docker

Использование Docker предоставляет множество вариантов для упрощения и оптимизации процесса разработки, развертывания и управления приложениями и сервисами. Он позволяет создавать локальные контейнеры для разработки и тестирования приложений, изолируя их зависимости от операционной системы и других приложений на хосте. Это упрощает перенос приложений между окружениями и предотвращает конфликты зависимостей. Также Docker идеально подходит для построения микросервисной архитектуры, в которой каждый сервис упакован в отдельный контейнер. Такая организация обеспечивает легкость масштабирования, управления и обновления сервисов независимо друг от друга. Docker также может взаимодействовать с инструментами непрерывной интеграции и доставки (CI/CD), упрощая процесс сборки, тестирования и развертывания приложений, позволяя поддерживать продукт в актуальном состоянии. Он также облегчает развертывание приложений в различных окружениях, включая локальные серверы, облачные платформы и собственные серверы из-за упаковки каждого сервиса в отдельный контейнер. С помощью

Docker также можно создавать воспроизводимые тестовые среды для автоматизации или упрощения тестирования приложений.

Применение Docker для нагрузочного тестирования и работы с Supabase может значительно облегчить процесс разработки и проверки приложения, потому что он позволяет создавать контейнеры, в которых могут быть развернуты инструменты для нагрузочного тестирования, такие как Apache JMeter. Использование контейнеров обеспечивает изоляцию тестового окружения от основной системы и обеспечивает возможность воспроизводимости тестовых сценариев, не оказывая на нее влияния. Кроме этого, Docker может быть использован для развертывания и управления инфраструктурой Supabase. Он предоставляет возможность создать контейнеры с функционалом Supabase и зависимостями приложения на этой платформе, такими как PostgreSQL, и легко управлять ими с помощью Docker Compose и других инструментов работы с контейнерами. Это обеспечивает изолированное и воспроизводимое окружение для разработки, тестирования и развертывания приложения на базе Supabase.

После создания контейнеров с тестовыми инструментами можно легко настроить тестовые сценарии и запустить нагрузочное тестирование функционала приложения. Облегчает масштабирование и настройку тестового окружения для приложений также то, что Docker обеспечивает гибкость в управлении конфигурациями тестовых сред.

В конечном итоге, Docker предоставляет широкие возможности для улучшения процесса разработки, развертывания и управления приложениями и сервисами. Он облегчает работу разработчиков, повышая эффективность и обеспечивая надежность работы приложений в различных окружениях. Использование Docker для нагрузочного тестирования и взаимодействия с Supabase обеспечивает гибкость, изолированность и воспроизводимость окружения, которая упрощает процесс разработки, тестирования и развертывания приложения.

2.8 Базы данных

2.8.1 PostgreSQL

Использование PostgreSQL, основанной на объектно-реляционной базе данных, оправдано, поскольку Supabase, платформа для разработки приложений, основана на этой системе управления базами данных. PostgreSQL известна своей надежностью, производительностью и масштабируемостью, из-за которых она является привлекательным выбором для хранения данных в приложениях Supabase. Благодаря PostgreSQL, Supabase предоставляет разработчикам удобные инструменты для работы с данными, обеспечивая при этом высокий уровень безопасности и производительности.

PostgreSQL – это самая популярная объектно-реляционная система управления базами данных, которая обладает большим набором функций и хорошо подходит для хранения данных различных типов приложений. Она распространяется по лицензии, которая делает ее свободно доступной для использования, модификации и распространения. Это является важным фактором при выборе как системы управления базы данных, так и платформы разработки приложений. PostgreSQL предоставляет множество встроенных типов данных, функций и операторов, а также поддерживает пользовательские типы данных и расширения, что позволяет разработчикам создавать собственные функции для удовлетворения специфических потребностей разрабатываемых продуктов. PostgreSQL полностью совместим с языком структурированных запросов SQL, обеспечивая легкость и удобство в работе с данными. Эта СУБД поддерживает множество расширений SQL, таких как подзапросы, объединения, агрегатные функции и триггеры и обеспечивает высокий уровень транзакционной безопасности благодаря поддержке транзакций ACID, означающих атомарность, согласованность, изоляция и устойчивость. Они предоставляют целостность данных и надежность при работе с транзакциями. PostgreSQL обладает хорошей масштабируемостью и производительностью, поддерживает различные методы репликации и кластеризации для обеспечения высокой доступности

приложений. Также эта система предоставляет различные механизмы аутентификации и авторизации, включая роли пользователей, привилегии и аудит действий и есть поддержка шифрования данных и SSL-соединения для обеспечения безопасности данных в пути.

Эти особенности делают PostgreSQL самой популярной системой управления базами данных в мире, и ее использование в качестве базы данных для Supabase обеспечивает надежное и производительное хранилище данных для разрабатываемых приложений.

2.8.2 Аналоги PostgreSQL

Другой популярной открытой реляционной системой управления базами данных является MySQL, которая, как и PostgreSQL, широко используется в различных приложениях. Она также предоставляет разнообразные функции и хорошо подходит для приложений разного плана. Однако, по сравнению с PostgreSQL, эта СУБД имеет некоторые различия в функционале и подходе к реализации некоторых функций.

Улучшенная версия MySQL под названием MariaDB, разработана для обеспечения бесплатной и открытой альтернативы MySQL. Она сохраняет с ней совместимость и расширяет ее функциональность некоторыми дополнительными возможностями. MariaDB предлагает ряд новых функций и улучшений по сравнению с MySQL и поддерживает большое количество способов хранения данных.

SQLite является встраиваемой СУБД, которая обладает некоторыми уникальными особенностями по сравнению с PostgreSQL. Она проста в использовании, легко встраивается в приложения и не требует отдельного сервера базы данных для работы, обеспечивая хранение данных в одном файле базы данных, что удобно для простых приложений, но может быть менее подходящим для крупных и высоконагруженных систем

Реляционная СУБД, разработанная компанией Microsoft для работы под управлением операционных систем Windows, называется Microsoft SQL Server. Она обладает богатым набором функций, включая поддержку транзакций,

хранимых процедур, триггеров и многих других. Эта система хорошо интегрируется с другими продуктами Microsoft, делая ее популярным выбором для организаций, использующих технологии Microsoft.

Эти аналоги PostgreSQL предоставляют большой выбор, в зависимости от конкретных потребностей и требований проекта. Каждая из них имеет свои сильные и слабые стороны, и решение о выборе между ними зависит от конкретной ситуации и предпочтений разработчика.

2.8.3 Преимущества PostgreSQL

PostgreSQL представляет собой удобную объектно-реляционную систему управления базами данных, обладающую широким спектром функций и хорошо подходящую для различных типов приложений. Доступным для использования его делает то, что это свободно распространяемое открытое решение. PostgreSQL обладает обширным набором функций, поддерживает множество типов данных, имеет продвинутый SQL-движок и расширяема под нужды программных продуктов. Благодаря полной совместимости с языком SQL, Postgres обеспечивает легкость и удобство работы с данными, а также поддерживает транзакции ACID для высокого уровня транзакционной безопасности. Она также хорошо масштабируется и обеспечивает высокую производительность, делая себя подходящим выбором для приложений различного масштаба.

2.9 Взаимодействие компонентов

При запуске тестов Apache JMeter генерирует запросы к API Supabase с заданной нагрузкой. Эти запросы направляются на сервер Supabase, после чего они принимаются и обрабатываются с использованием этого API. На этом этапе выполняются необходимые операции взаимодействия с базой данных PostgreSQL. Если запросы предназначены для записи данных, то они сохраняются или обновляются в базе. В случае запросов на чтение возвращаются соответствующие данные.

Ввиду того, что при нагрузочном тестировании количество схожих запросов довольно велико, в диаграмме отражена общая структура последовательности, где пропущено некоторое количество одинаковых шагов. Диаграмма представлена на рисунке 6.

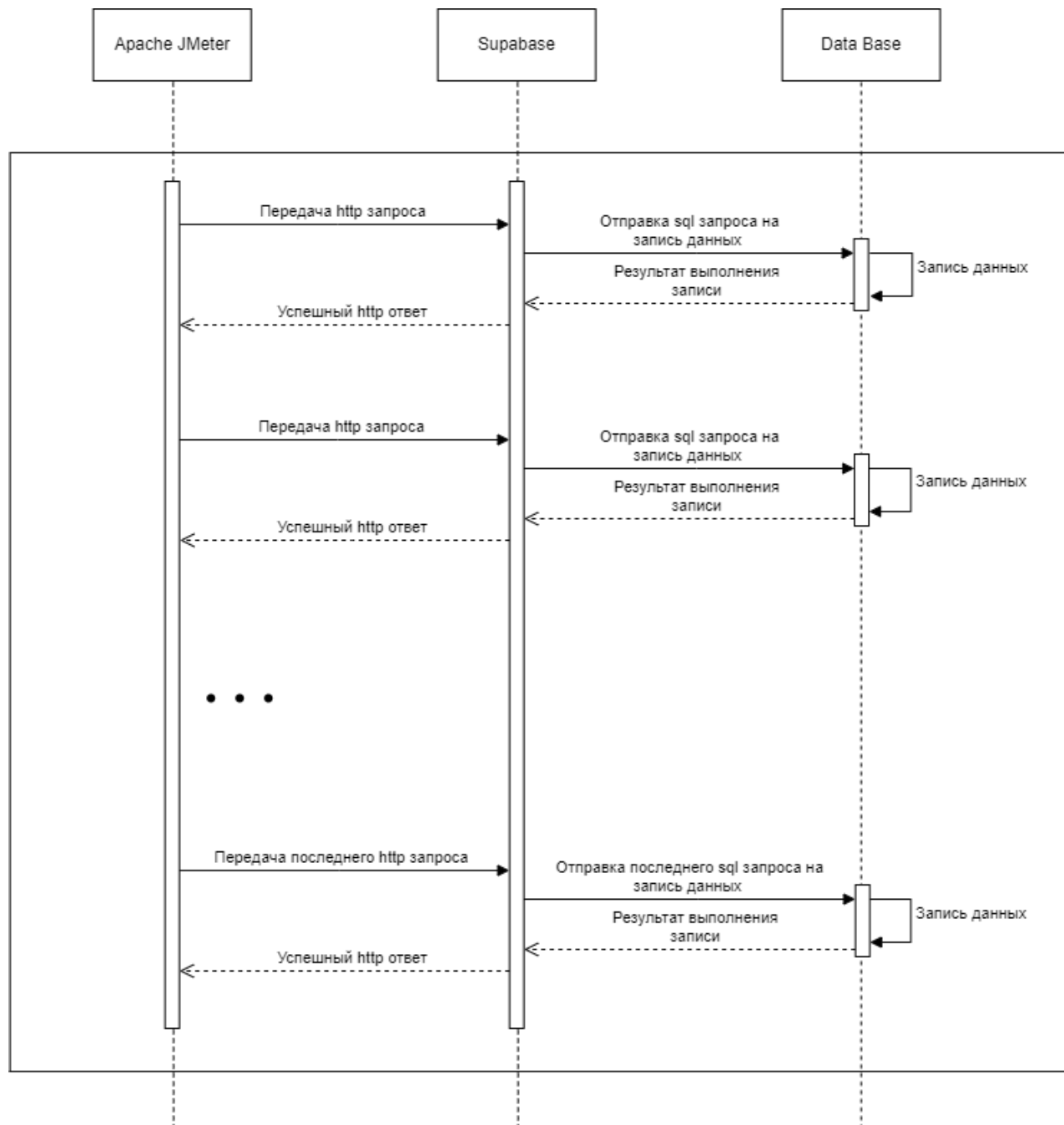


Рисунок 6 – UML-диаграмма последовательности

3 Реализация нагрузочного тестирования

3.1 Установка JDK

Так как JMeter написан на языке Java, для его корректной работы необходимо наличие установленного JDK. Он нужен для выполнения Java-приложений, так как содержит необходимые инструменты, библиотеки и исполнительную систему Java. Также в него входят примеры, документация и другие утилиты, необходимые для разработки на этом языке.

Чтобы избежать потенциальных конфликтов версий, необходимо проверить совместимость версий JDK и JMeter. В данной работе был использован Apache JMeter версии 5.6.3. Для этой версии на официальном сайте написано, что минимальным требованием для работы является наличие установленной Java не старше чем Java 8, но рекомендуется использовать не старше Java 17.

Перед началом установки, во избежание проблем, нужно проверить версию Java на компьютере, выполнив в терминале команду `java -version`. Если JDK установлен, то в терминал будет выведена информация об имеющейся версии. Иначе покажется сообщение об ошибке, в этом случае нужно его установить.

В случае отсутствия JDK, нужно его загрузить, выполнив в терминале команду `sudo apt install default-jdk`. В этом случае не нужно выбирать версию, потому что по умолчанию будет установлена последняя. Также не нужно беспокоиться о совместимости с архитектурой процессора или операционной системой, так как при установке таким методом, выбор происходит автоматически, в соответствии с конфигурацией системы.

После того, как установка будет завершена, чтобы убедиться в корректности выполнения проделанной операции, следует снова выполнить команду `java -version`. В случае, если все операции были проделаны правильно, как упоминалось выше, в терминале будет отображена информация о версии установленного JDK.

3.2 Установка Apache JMeter

После установки JDK, можно переходить к установке Apache JMeter. Одним из его преимуществ является то, что он не требует самой установки, а может выполняться из распакованного архива, содержащего исполняемый файл.

Для начала нужно перейти на официальный сайт Apache JMeter, на страницу загрузки. На этой странице нужно выбрать файл для скачивания, здесь представлена последняя версия программы в виде архива или файла исходного кода. Чтобы решить поставленную задачу, нужно скачать архив и распаковать его в удобную директорию.

После того, как архив распакован, для правильной работы необходимо добавить программе JMeter Plugins Manager. Он нужен для того, чтобы управлять установленными плагинами и позволяет устанавливать или удалять конкретные расширения. Это решение нужно для облегчения программы и ускорения работы, так как существует большое количество плагинов для решения различных задач, но не все они нужны для каждого запуска.

Следующим шагом может выступать сам запуск Apache JMeter, потому что после проделанных действий он готов к работе и можно переходить к его использованию. Для того, чтобы его запустить, необходимо в терминале перейти в каталог, содержащий исполняемый файл, обычно это папка bin в каталоге программы. Далее можно произвести запуск программы с графическим интерфейсом с помощью команды `./jmeter`. В результате проделанных шагов Jmeter будет запущен и можно переходить к этапу создания плана тестирования.

3.3 Установка Docker

Для правильного функционирования Supabase, требуется его развертывание в контейнере Docker, о чем написано перед инструкцией по установке self-hosting Supabase. Это позволяет изолировать контейнеры и обеспечить масштабируемость проекта.

Для удобства развертывания системы ссылка на инструкцию по установке Docker на различные платформы. Сначала, если среда рабочего стола отличается от `gnome-terminal`, то ее следует установить командой `sudo apt install gnome-terminal`. Это позволит в дальнейшем использовать Docker Desktop, который предоставляет графический интерфейс для удобства управления контейнерами и содержит Docker Engine, обеспечивающий работу контейнеров.

Далее, после установки среды рабочего стола, необходимо скачать deb-пакет Docker Desktop, ссылка для скачивания которого представлена также на странице инструкции. Следующим шагом следует установить скаченный пакет с помощью команды `sudo apt-get install ./docker-desktop-<version>-<arch>.deb`

Для того, чтобы убедиться в корректности выполнения операций, можно выполнить команду `docker --version`, которая покажет версию установленного Docker Desktop. Если все выше представленные шаги были выполнены корректно, то Docker Desktop готов к использованию.

3.4 Установка Supabase

После установки описанных инструментов, можно переходить к развертыванию self-hosting Supabase, где будет база данных, в которую будут записываться логи.

Сперва необходимо получить все необходимые файлы Supabase, с помощью клонирования репозитория Git, выполнив команду `git clone --depth 1 https://github.com/supabase/supabase`. После выполнения данного шага в текущей директории создастся папка с проектом. Для дальнейшей работы необходимо перейти в папку `docker` внутри проекта командой `cd supabase/docker`.

Следующим шагом идет замена файла с расширением `.env.example` на `.env`, так как необходимо сделать этот файл файлом конфигурации, но в первом варианте он является только информационным. Выполнить это можно с помощью команды `cp .env.example .env`.

После проделанной операции self-hosting Supabase готов к использованию.

4 Нагрузочное тестирование

4.1 Логирование

Логи – это файлы, содержащие информацию о работе какой-либо программы, в которые вносятся определённые действия пользователя или самой программы.

Логирование нужно для анализа использования функционала ПО и отслеживания показателей и действий во время неполадок. Это помогает в кратчайшие сроки разобраться с причинами ошибок и устранить их.

Так как в система используется большим количеством пользователей, могут случаться внештатные ситуации, влекущие за собой повышение нагрузки на определенные функции. Так, во время сбоя, все клиенты, которых он затронул, отправляют логи на сервер, создавая большое количество одинаковых запросов.

Для того, чтобы проверить, сможет ли система безотказно работать в условиях повышенной нагрузки существует тестирование производительности. В данной работе будет выполнено нагрузочное тестирование функции логирования, для проверки соответствия системы этим требованиям.

Весь процесс будет проходить на тестовом стенде с такими характеристиками:

- Процессоры: 4 CPU
- Оперативная память: 4 Гб
- Дисковое пространство: 50 Гб

4.2 Методика тестирования

Для проведения нагрузочного тестирования был использован специально подобранный тестовый сценарий, который имитировал реальные условия использования системы. Тестирование ограничено в течение определенного временного интервала с постоянной нагрузкой в 125 транзакций в секунду (TPS) для оценки производительности системы при пиковой нагрузке, исходя из данных в таблице 1.

Таблица 1 – вероятностно-временные характеристики.

Характеристика	Значение
Общее количество пользователей в субъекте	1500000
Доля участников, которых затронул сбой	0,1
Длительность временного промежутка при возникновении сбоев, мин	20
Требуемый пиковый TPS	125
Средний размер лога, МБ	0,06
Срок хранения лога, дни	7
Объем БД, МБ	63000
Объем БД, ГБ	61,5234375

В качестве сценария нагрузочного тестирования используется POST запрос для добавления записи лога в БД. Лог содержит следующие поля:

- продукт (Журнал, Дневник, Спутник, Библиотека);
- версия приложения;
- ОС (Android/iOS);
- устройство;
- версия ОС;
- прошивка;
- объем оперативной памяти;
- объем свободной памяти;
- объем хранилища;
- объем свободного пространства в хранилище;
- токен АУПД;
- profile-id;
- ФИО;
- СНИЛС;
- организация;

- дампы с логом;
- статус (новое, в обработке, решено);
- время создания записи.

Эти поля позволяют осуществлять поиск в БД и анализ данных.

4.2.1 Цель тестирования

Цель тестирования – проверить стабильность системы при пиковой нагрузке в 125 запросов в секунду, где тестовым запросом будет отправка лога через POST-запрос.

4.2.2 Профиль нагрузки

Для реализации нагрузки использовался профиль, где было 50 одновременно работающих потоков пользователей, ограниченные 125 TPS. Средний вес лога 62,7 кб, а длительность нагрузки – 10 минут.

4.2.3 Пользовательские сценарии

POST-запрос на отправку данных лога выглядит следующим образом:

```
POST /rest/v1/log_storage?select=*
{
  "full_name": "Петров Петр Петрович",
  "snils": "843 607 021 93",
  "organization": "iteco",
  "dump_with_log": "${logData}",
  "status": "Новое",
  "product": "Журнал",
  "application_version": "1.3.0-beta-2455 Thu Nov 09 21:54:04
GMT+03:00 2023",
  "OS": "Android",
  "OS_version": "Android 10",
  "firmware": "29",
  "device": "Xiaomi Redmi Note 7",
  "RAM_capacity": "2734 MB",
  "amount_of_free_memory": "2055 MB",
  "storage_capacity": "21789 MB",
  "free_space_storage": "8535 MB",
  "AUPD_token": "13456543213245432"
}
```

В соответствии с этой структурой был сформирован curl, выглядящий следующим образом:

					ПГУ 1.090301.32.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

```
curl 'http://localhost:8000/api/pg-meta/default/query?key=' -X POST
-H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0' -H 'Accept: application/json' -H
'Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3' -H 'Accept-
Encoding: gzip, deflate, br' -H 'Referer:
http://localhost:8000/project/default/editor/25264' -H 'content-
type: application/json' -H 'x-connection-encrypted: ' -H 'x-request-
id: 12f5fd34-df64-422d-a71e-879bbb36b36f' -H 'Origin:
http://localhost:8000' -H 'Authorization: Basic
c3VwYWJhc2U6dGhpc19wYXNzd29yZF9pc19pbmNlY3VyZV9hbmRfc2hvdWxkX2JlX3Vw
ZGF0ZWQ=' -H 'Connection: keep-alive' -H 'Sec-Fetch-Dest: empty' -H
'Sec-Fetch-Mode: cors' -H 'Sec-Fetch-Site: same-origin' --data-raw
$ '{"query": "insert into public.\\\"Log\\\" (\\\"full
name\\\",snils,organization,dump_with_log,status,product,application_
version,\\\"OS\\\",\\\"OS_version\\\",firmware,device,\\\"RAM_capacity\\\"
,amount_of_free_memory,storage_capacity,free_space_storage,\\\"AUPD_t
oken\\\")
select \\\"full
name\\\",snils,organization,dump_with_log,status,product,application_
version,\\\"OS\\\",\\\"OS_version\\\",firmware,device,\\\"RAM_capacity\\\"
,amount_of_free_memory,storage_capacity,free_space_storage,\\\"AUPD_t
oken\\\" from jsonb_populate_recordset(null::public.\\\"Log\\\",
' [{\\\"full
name\\\":\\\"\\u041f\\u0435\\u0442\\u0440\\u043e\\u0432
\\u041f\\u0435\\u0442\\u0440\\u043e\\u0432\\u0438\\u0447\\\",\\\"snils\\\":\\\"8
43
607
021
93\\\",\\\"organization\\\":\\\"iteco\\\",\\\"dump_with_log\\\":\\\"https://
teamtest.r7-
office.ru:3443\\\",\\\"status\\\":\\\"\\u041d\\u043e\\u0432\\u043e\\u0435\\\",
\\\"product\\\":\\\"\\u0416\\u0443\\u0440\\u043d\\u0430\\u043b\\\",\\\"applicat
ion_version\\\":\\\"1.3.0-beta-2455 Thu Nov 09 21:54:04 GMT+03:00
2023\\\",\\\"OS\\\":\\\"Android\\\",\\\"OS_version\\\":\\\"Android
10\\\",\\\"firmware\\\":\\\"29\\\",\\\"device\\\":\\\"Xiaomi Redmi Note
7\\\",\\\"RAM_capacity\\\":\\\"2734
MB\\\",\\\"amount_of_free_memory\\\":\\\"2055
MB\\\",\\\"storage_capacity\\\":\\\"21789
MB\\\",\\\"free_space_storage\\\":\\\"8535
MB\\\",\\\"AUPD_token\\\":\\\"13456543213245432\\\"}]}\\\"';" }
```

4.3 Запуск проверки

Перед началом тестирования, требуется запустить Docker Desktop, активировав Docker Engine, управляющий контейнером с установленным Supabase, для его дальнейшего функционирования.

					ПГУ 1.090301.32.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

Далее для запуска самого Supabase необходимо в терминале командой `cd supabase/docker` перейти в каталог с файлом `docker-compose.yml`, содержащим конфигурацию Docker Compose для Supabase. Docker Compose — это инструмент, предназначенный для определения и управления многоконтейнерными Docker-приложениями. С его помощью можно легко управлять запуском нескольких контейнеров, которые образуют одно приложение, используя простой файл конфигурации YAML. Docker Compose позволяет описать всю необходимую инфраструктуру приложения, включая сети, тома и переменные окружения, в одном месте и управлять ею с помощью нескольких команд.

Находясь в этом каталоге, нужно выполнить команду `docker compose pull` для загрузки (или обновления) образов контейнеров, определённых в файле `docker-compose.yml`. Эта команда полезна, когда нужно убедиться, что используются самые последние версии образов перед запуском или перезапуском сервисов.

После этого следует выполнить команду `docker compose up -d`. Эта команда запускает все сервисы для Supabase, определённые в файле `docker-compose.yml`. Ими являются контейнеры для различных компонентов Supabase, таких как база данных PostgreSQL, API (PostgREST), сервер аутентификации, и другие сервисы, необходимые для полноценной работы. Опция `-d` означает `detached mode`, запускает контейнеры в фоновом режиме, позволяя терминалу оставаться доступным для выполнения других команд и не блокироваться выводом логов контейнеров.

Далее, когда Supabase запущен, можно проверить базу данных. Для этого, используя шлюз API на порту 8000 можно получить доступ к панели управления Supabase. В описываемом случае Docker используется локально, поэтому обращение будет через `localhost:8000`.

На следующем шаге будет предложено ввести имя пользователя и пароль. По умолчанию в качестве учетных данных используются:

- имя пользователя: `supabase`;
- пароль: `this_password_is_insecure_and_should_be_updated`.

Редактор таблицы находится по адресу `project/default/editor`. Здесь находится таблица, содержащая логи, запись в которую и будет тестироваться.

Для того, чтобы провести тестирование, нужно настроить план тестирования в JMeter. Для его запуска нужно с помощью команды `cd apache-jmeter-5.5/bin` перейти в директорию `bin` в папке с JMeter, после чего запустить графический интерфейс инструмента командой `./jmeter`.

После этого необходимо настроить план тестирования, который будет сохранен в файл с расширением `jmx`. Здесь нужно указать количество пользователей (поток), которые будут выполнять сценарий тестирования. После этого нужно добавить `Sampler`. `Sampler` определяет действия, которые пользователи будут выполнять, такие как HTTP-запросы. В нем нужно указать:

- Name: имя запроса;
- Server Name or IP: Домен или IP-адрес целевого сервера;
- Port Number: Номер порта (обычно 80 для HTTP и 443 для HTTPS);
- HTTP Request: метод запроса (GET, POST и т.д.);
- Path: Путь к ресурсу на сервере;
- Parameters: параметры запроса, если есть.

Далее следует добавить `Listener`. Он позволяет собирать и отображать результаты тестирования.

Когда план тестирования настроен, можно преступать непосредственно к тестированию. Для этого есть кнопка запуска в панели инструментов. Здесь же есть кнопка остановки процесса.

После выполнения тестирования можно посмотреть его результаты, перейдя в соответствующее меню `Listener`.

5 Результаты тестирования

Общее количество запросов за 10 минут составило 79 866 запросов, что означает способность системы обработать большой объем запросов за сравнительно короткий промежуток времени. Среднее время запроса является достаточно низким показателем для такого объема данных и составляет 11,76 мс. Медианное время запроса равно 10 мс, что указывает на быстроту обработки большинства запросов.

Показатели процентилей также свидетельствуют о хорошей производительности системы, где 90-й процентиль — 19 мс и 95-й процентиль — 24 мс. Согласно этим данным, в 95% случаев запросы обрабатывались быстрее, чем за 24 мс. Также стабильность системы подтверждает факт отсутствия ошибочных запросов (рисунок 7).

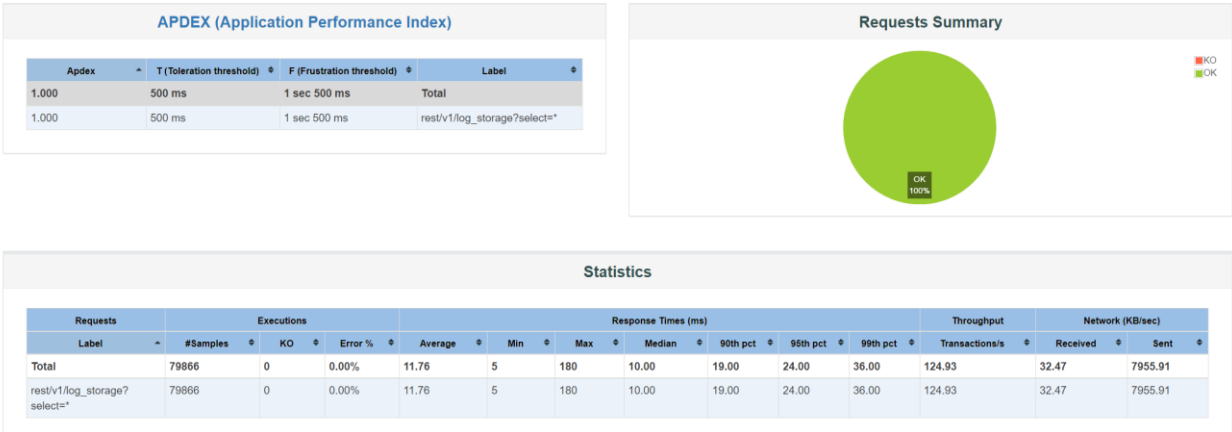


Рисунок 7 - Общая статистика

На графике на рисунке 8 представлено количество запросов в секунду в течение всего тестирования. Согласно графику, количество запросов оставалось стабильным, без значительных просадок или скачков нагрузки. Это означает, что система способна поддерживать нагрузку равномерно, без потерь в производительности.

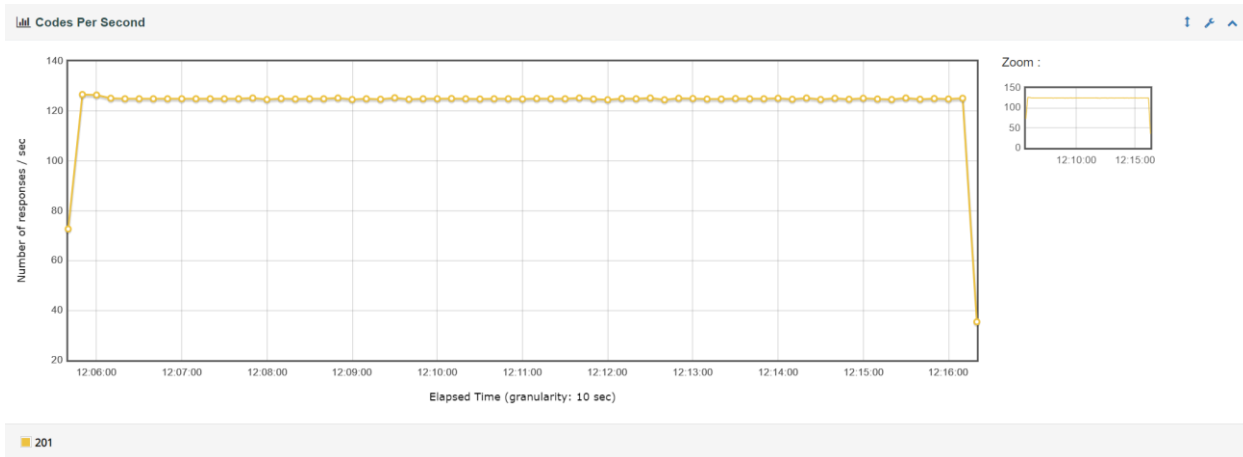


Рисунок 8 - График количества запросов в секунду

На графике на рисунке 9 времени отклика видно, что среднее время на выполнение запроса оставалось приблизительно на одном уровне, не увеличиваясь со временем.

По этому графику можно судить о надежности системы, так как на нем отражено то, что она обрабатывала запросы с одинаковой скоростью и не перегружалась при этом.

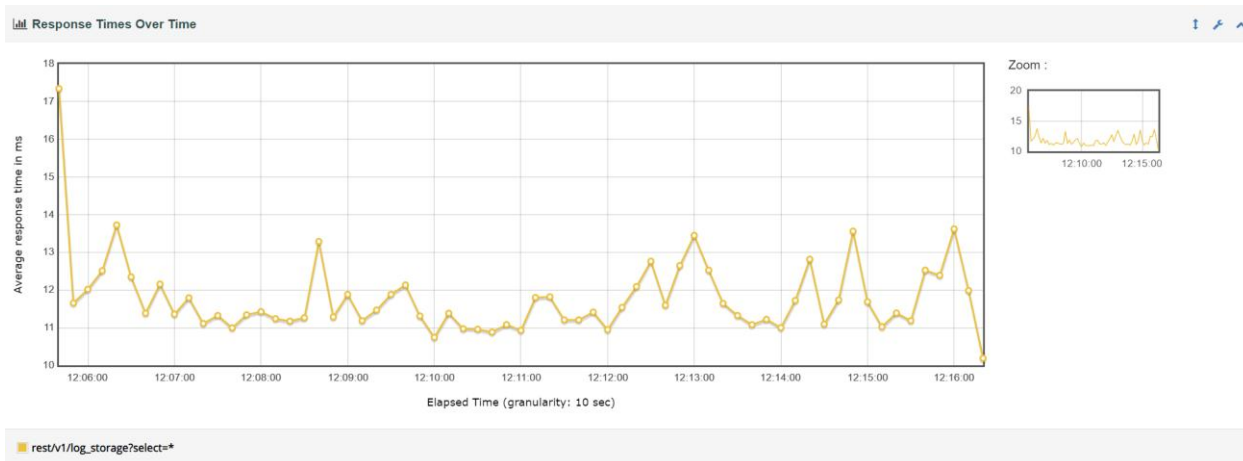


Рисунок 9 - График откликов

На рисунке 10 изображен график пропускной способности сети, где показано, что скорость обмена данными все время стабильно сохранялась на уровне 8 000 000 бит в секунду. Этот мера также подтверждает факт надежности и производительности системы, потому что показатель достаточно высок.

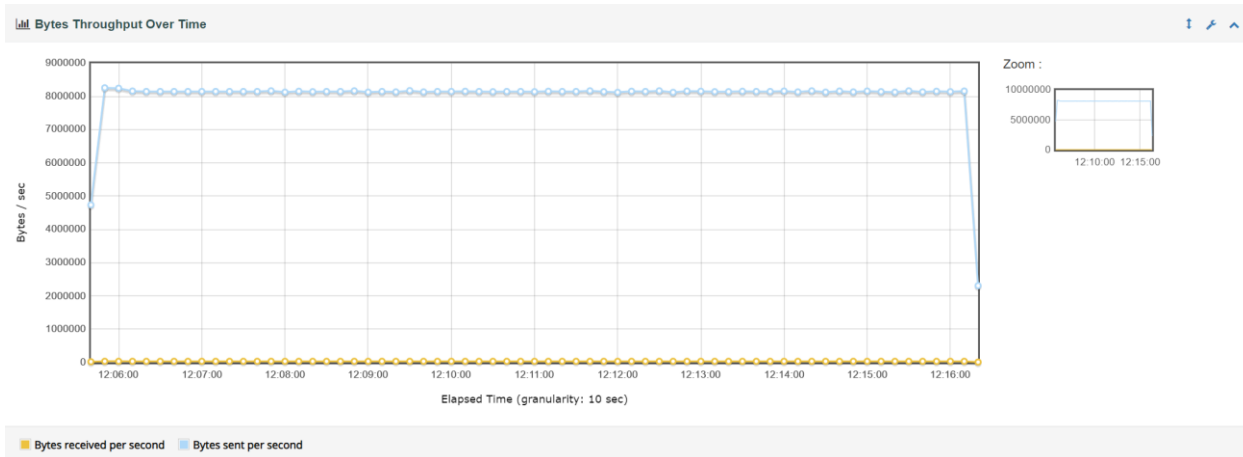


Рисунок 10 - График пропускной способности

На графике количества устройств видно, что тест задействовал 50 активных потоков, которые обеспечивали необходимую нагрузку на систему, имитируя условия, максимально приближенные к реальным (рисунок 11).

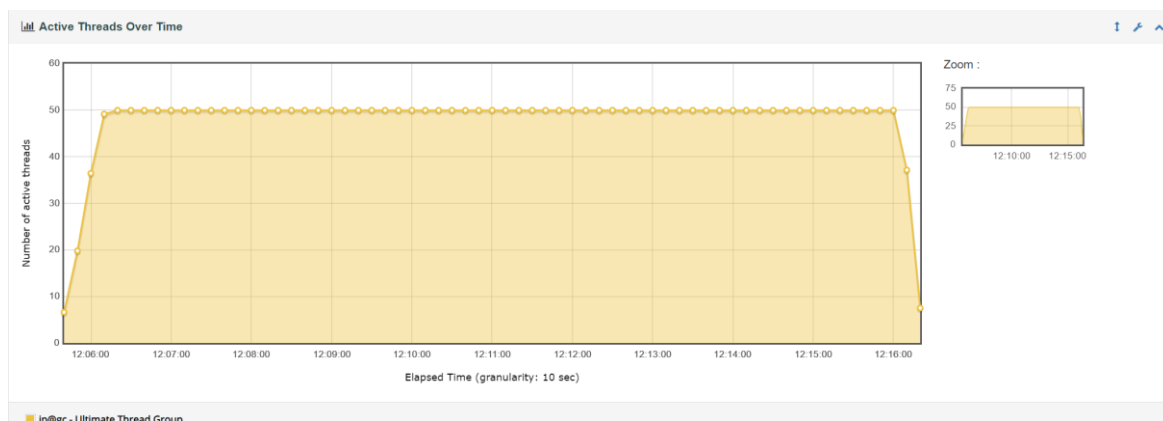


Рисунок 11 - График активных потоков

Во время нагрузочного тестирования процессор был задействован примерно на 100% из возможных 400%, что означает полное использование системой только одного процессора из четырех доступных. Интенсивное использование ресурсов процессора также приводит к повышению местами нагрузки на него до 125%. Это отражает график, изображенный на рисунке 12, где также отражено использование оперативной памяти во время процесса тестирования. Она, в свою очередь, была использована в среднем в количестве 1,86 Гб, также с небольшими повышениями потребления.

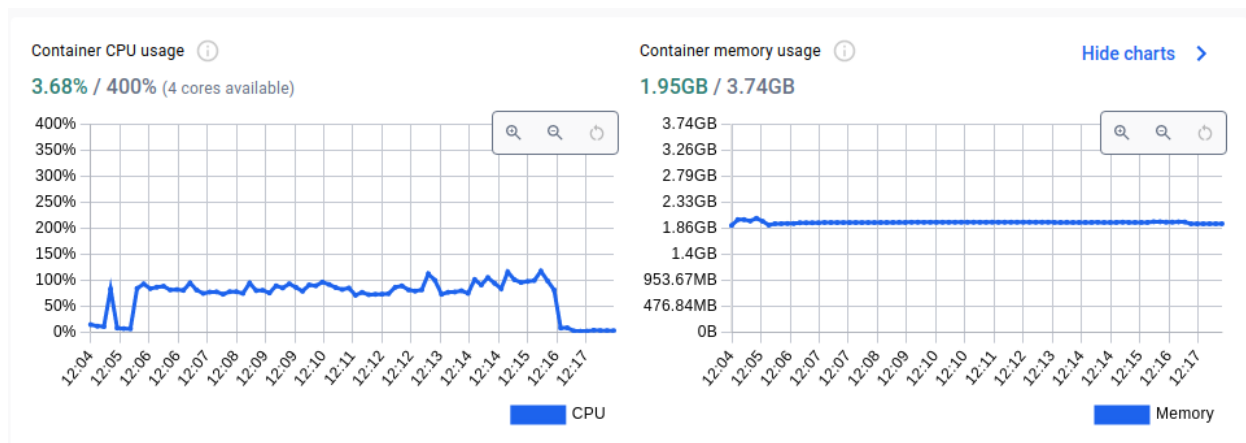


Рисунок 12 - График использования ресурсов в Docker

Ранее упоминалось, что было выполнено 79 866 запросов. Проверить корректность работы системы можно, зайдя в интерфейс Supabase для проверки количества записей. На рисунке 13 показано, что в таблицу было добавлено 79 866 строк, подтверждая безошибочное выполнение и обработку запросов, а значит настройка теста была сделана правильно и рассматриваемая система работает корректно.

id	project	app	os	device	os_version	firm	ram	amount	storage	free	aup	full
74378	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74379	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74380	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74381	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74382	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74383	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74384	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74385	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74386	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74387	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74388	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74389	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74390	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74391	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74392	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74393	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74394	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74395	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе
74396	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе

Рисунок 13 - Добавленные данные в таблице

5.1 Вывод

Рассматриваемый стенд показал способность безошибочно обработать требуемую нагрузку, в размере 125 запросов в секунду, доказывая свою надежность и производительность, а также корректность настройки и выполнения нагрузочного тестирования.

Тест показал, что система логирования, развернутая на стенде, оснащенный 4 CPU и 4 Гб RAM, способна справиться с интенсивной нагрузкой и стабильно работать под расчетной нагрузкой без увеличения времени отклика. В связи с этим можно сделать вывод о том, что нагрузочный тест сконфигурирован правильно и корректно выполнен, а система логирования отвечает требованиям производительности.

					ПГУ 1.090301.32.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

6 Охрана труда

Создание правильного и безопасного рабочего места является важным аспектом при выполнении нагрузочного тестирования. Оптимальная организация рабочего пространства способствует не только повышению производительности, но и снижает риск возникновения профессиональных заболеваний и травм. Основные рекомендации по организации рабочего места включают следующие положения.

Рабочий стол должен быть достаточно просторным для размещения всех необходимых инструментов и оборудования. Поверхность стола должна быть не слишком высокой и не слишком низкой, чтобы руки и запястья могли находиться в естественном положении. Рабочее кресло должно иметь регулировки высоты и наклона спинки, а также подлокотники для обеспечения поддержки рук. Кресло должно поддерживать нижнюю часть спины (поясничный отдел), чтобы предотвратить боли в спине.

Монитор должен быть установлен на уровне глаз на расстоянии 50-70 см от лица. Верхняя часть экрана должна находиться на уровне глаз или чуть ниже, чтобы предотвратить напряжение шеи. Экран должен быть расположен так, чтобы минимизировать отражения и блики от источников света. Клавиатура должна располагаться так, чтобы запястья могли оставаться прямыми, а локти — под углом 90 градусов или немного больше. Мышь должна находиться на одном уровне с клавиатурой и быть в пределах легкой досягаемости, чтобы минимизировать движение руки.

Эргономика рабочего места имеет важное значение для сохранения здоровья и повышения эффективности труда. Правильно организованное рабочее место помогает предотвратить множество профессиональных заболеваний, таких как боли в спине и шее, синдром запястного канала, проблемы со зрением и общую усталость. Неправильная поза или длительное пребывание в неудобной позе могут привести к мышечным напряжениям, нарушению кровообращения и другим негативным последствиям.

Эргономичные принципы включают использование оборудования, адаптированного под физиологические и анатомические особенности человека. Например, регулируемые кресла и столы позволяют каждому работнику настроить рабочее место в соответствии с его индивидуальными потребностями. Положительное влияние на здоровье работников также оказывает правильное освещение, которое должно быть достаточным, но не слишком ярким, чтобы не напрягать глаза.

Эргономичное использование оборудования подразумевает регулировку яркости и контрастности монитора в соответствии с условиями освещения для снижения нагрузки на глаза. Также рекомендуется использование подставки для ног при необходимости, чтобы обеспечить комфортное положение ног и снизить нагрузку на позвоночник.

Безопасное обращение с компьютером включает в себя несколько аспектов, направленных на предотвращение как физических, так и психических нагрузок. Основные правила включают профилактику переутомления, эргономичное использование оборудования, электробезопасность и правильное использование программного обеспечения.

Для профилактики переутомления рекомендуется делать 5-10 минутные перерывы каждые 1-2 часа работы за компьютером, чтобы снизить нагрузку на глаза и избежать утомления. Периодически следует выполнять простые упражнения для глаз, такие как фокусировка на дальних и ближних объектах, чтобы предотвратить зрительное напряжение. Во время работы необходимо периодически менять позу, вставать и разминаться, чтобы избежать затекания мышц и улучшить кровообращение.

Электробезопасность включает регулярную проверку состояния электропроводки и всех подключенных устройств. Необходимо убедиться, что все кабели и розетки находятся в исправном состоянии. Использование устройств с заземлением и стабилизаторов напряжения позволяет защитить оборудование от перепадов напряжения.

Правильное использование программного обеспечения предполагает установление и обновление антивирусного программного обеспечения для защиты данных и системы от вредоносных программ. Регулярное резервное копирование данных помогает предотвратить их потерю в случае сбоев системы или аппаратных неисправностей.

					ПГУ 1.090301.32.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения данной бакалаврской работы была достигнута основная цель — изучение систем нагрузочного тестирования и разработка программной системы для проверки производительности системы логирования мобильных клиентов под нагрузкой. Были рассмотрены и проанализированы современные инструменты для нагрузочного тестирования, такие как Apache JMeter, Supabase и Docker. Эти инструменты были выбраны не случайно: их использование позволяет эффективно моделировать реальные условия эксплуатации систем и выявлять возможные узкие места в их производительности.

На этапе планирования были определены цели тестирования, ожидаемые результаты, а также разработана стратегия управления рисками. В ходе подготовки были созданы тестовые сценарии, имитирующие реальные сценарии использования приложения, и настроена тестовая среда. Этап выполнения включал запуск тестовых сценариев и мониторинг производительности системы. Анализ собранных данных позволил выявить слабые места системы и предложить пути их устранения. В завершении работы был подготовлен отчет о результатах тестирования, выявленных проблемах и рекомендациях по их устранению.

Результаты нагрузочного тестирования показали, что система логирования способна выдерживать высокие нагрузки, однако существуют определенные аспекты, которые требуют оптимизации для повышения производительности и устойчивости. В частности, было выявлено несколько узких мест, связанных с обработкой запросов и взаимодействием с базой данных, которые могут быть устранены путем оптимизации кода и настройки серверного окружения.

Использование Docker и Supabase для развертывания тестовой среды показало свою эффективность и удобство, так как эти инструменты обеспечивают изоляцию и консистентность среды, что крайне важно для корректного проведения тестов. Apache JMeter продемонстрировал свои возможности в моделировании нагрузочных сценариев и сборе подробных данных о производительности системы.

В заключение, данная работа подтвердила важность и необходимость проведения нагрузочного тестирования при разработке мобильных приложений. Предложенные методы и инструменты могут быть использованы для дальнейших исследований и оптимизации систем, обеспечивая высокое качество и надежность программных продуктов в условиях роста нагрузки и усложнения их функционала. Надеемся, что результаты и выводы данной работы будут полезны как для академического сообщества, так и для практикующих разработчиков и инженеров, занимающихся вопросами производительности и масштабируемости информационных систем.

					ПГУ 1.090301.32.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Тестирование ПО (QA testing) // Блог программиста [САЙТ] - URL: <https://blog-programmista.ru/post/24-testirovanie-po-qa-testing.html> (дата обращения: 19.04.2024).
2. Тестирование и обеспечение качества // mellarius [САЙТ] - URL: <https://mellarius.ru/testing> (дата обращения: 20.04.2024).
3. Типы и виды тестирования. Уровни тестирования. методы тестирования // intellect [САЙТ] - URL: <https://intellect.icu/typy-i-vidytestirovaniya-urovni-testirovaniya-metody-testirovaniya-5187> (дата обращения: 20.04.2024).
4. Рыбалко М. А., Иванова Е. А. Тестирование программного обеспечения, методы тестирования // Информационное общество: современное состояние и перспективы развития. – 2017. – С. 320-322.
5. Уровни Тестирования // Тестинг.ру [САЙТ] - URL: <http://www.protesting.ru/testing/testlevels.html> (дата обращения: 23.04.2024).
6. Этапы проведения нагрузочного тестирования // Тестинг.ру [САЙТ] - URL: <http://www.protesting.ru/automation/load/loadteststages.html> (дата обращения: 24.04.2024).
7. Цели нагрузочного тестирования // Тестинг.ру [САЙТ] - URL: <http://www.protesting.ru/automation/load/loadtestgoals.html> (дата обращения: 24.04.2024).
8. Нагрузочное тестирование // PerformanceLab [САЙТ]. - URL: <https://www.performance-lab.ru/blog/load-testing/testirovanieproizvoditelnosti> (дата обращения: 30.04.2024).
9. Использование диаграммы вариантов использования UML при проектировании программного обеспечения. // habr.com [САЙТ] - URL: <https://habr.com/ru/post/566218/> (дата обращения: 05.05.2024).
10. Искусство тестирования программ // Гленфорд Майерс. - Москва: Вильямс, 2005. - 736 с.

11. Sarojadevi H. Performance Testing: Methodologies and Tools. Journal of information engineering and applications, 2011, 5 - pp. 5-13.
12. Мясников С. О., Намиот Д. Е. Инструменты нагрузочного тестирования // Прикладная информатика. — 2018. — Т. 13, № 1. — С. 92–102.
13. User's Manual // jmeter.apache.org [САЙТ]. - URL: <https://jmeter.apache.org/usermanual/index.html> (дата обращения: 14.05.2024).

Приложение А

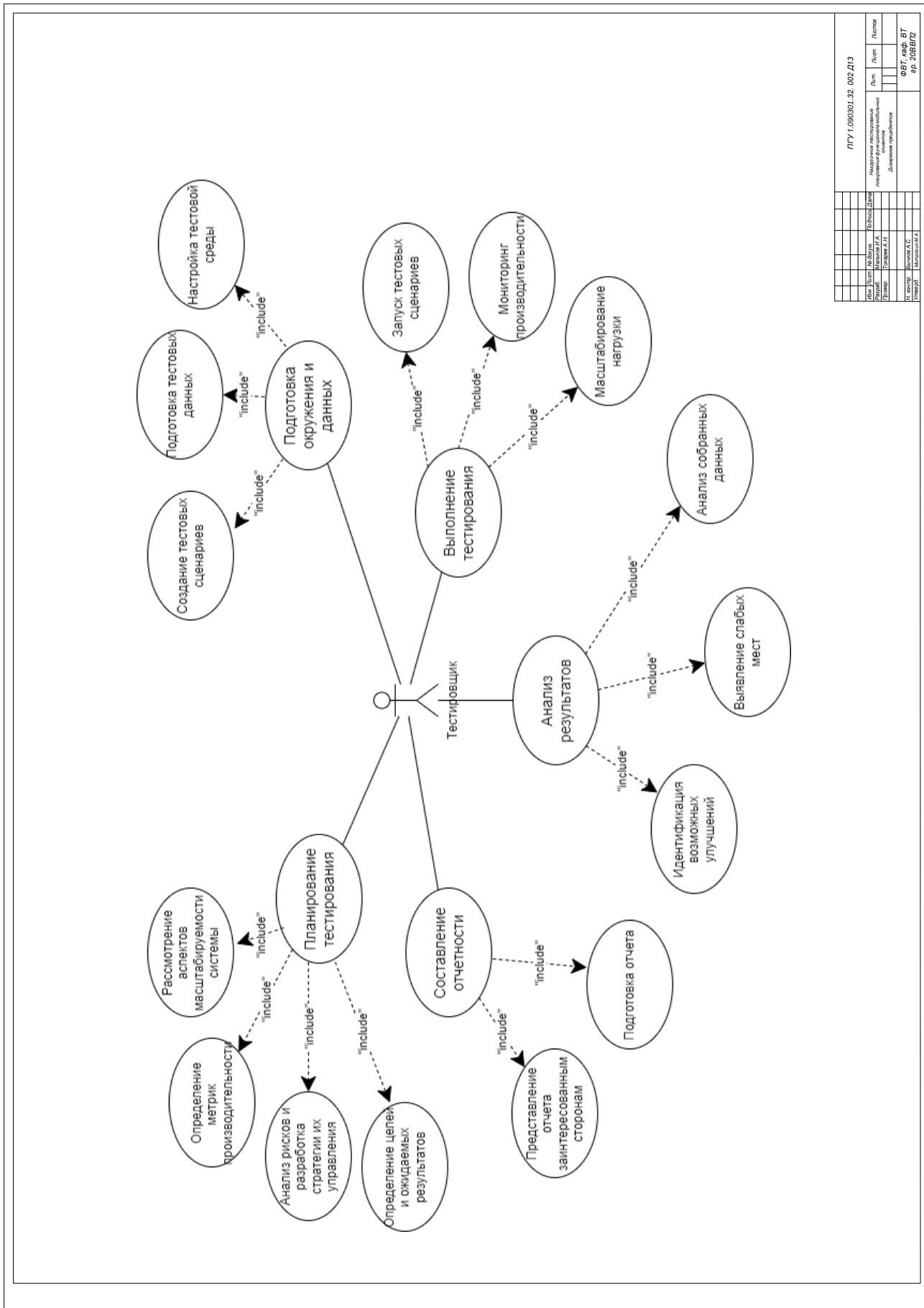
(Обязательное)

Нагрузочное тестирование логирования функционала мобильных клиентов

UML-диаграммы

					ПГУ 1.090301.32.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – UML-диаграмма вариантов использования

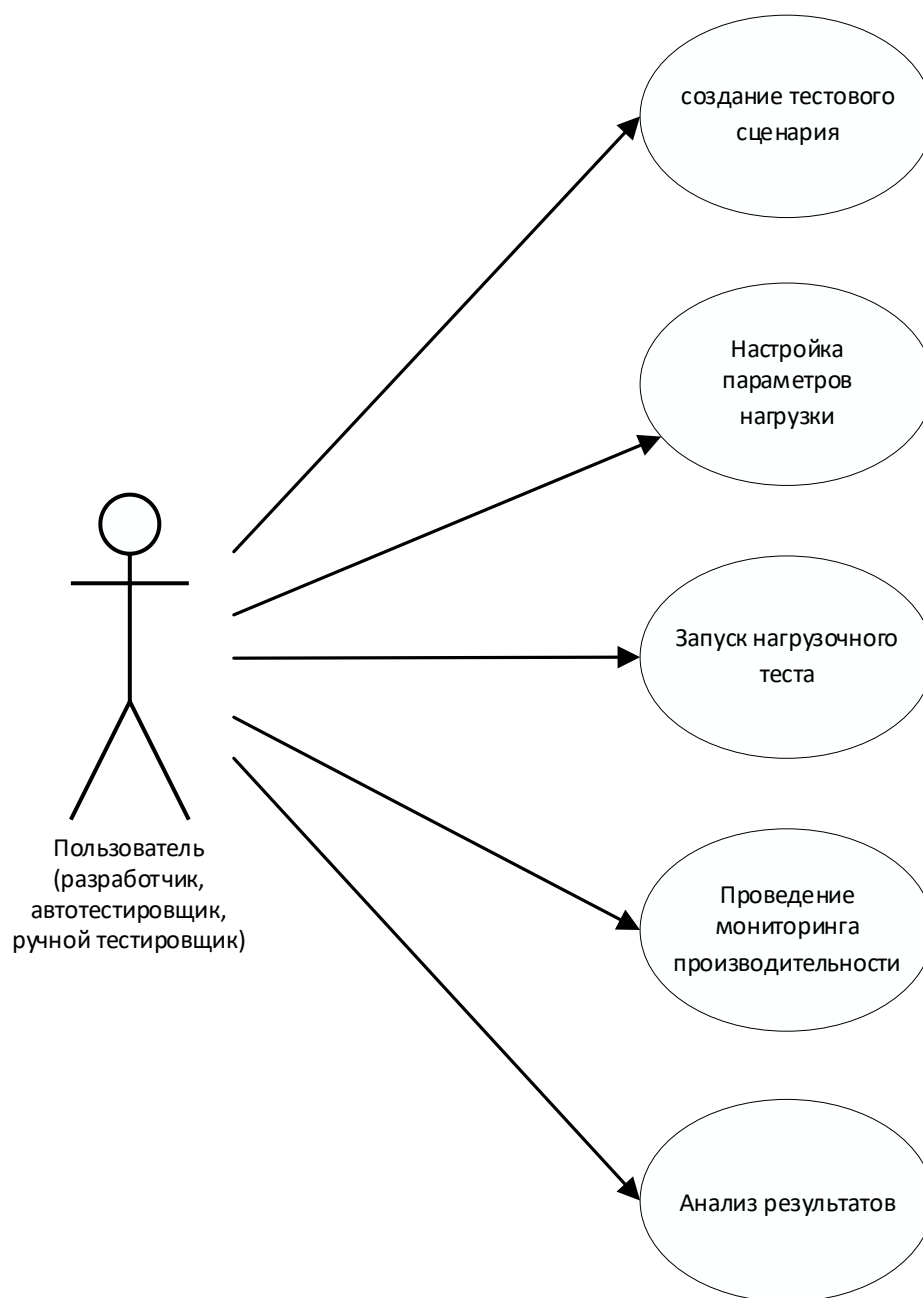


ПГУ 1.090301.32.002 Д13									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов
Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов	Рязань	Михайлов

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.32.001 ПЗ

Приложение А.2 – UML-диаграмма вариантов использования краткая



ПГУ 1.090301.32.003.Д13					
Изм.	Лист	№ докум.	Подпись	Дата	
Разработ.	Матвеев И.А.				
Провер.	Гонимов А.Н.				
И. контр.	Белков А.С.				
Утвержд.	Митрошкин И.А.				

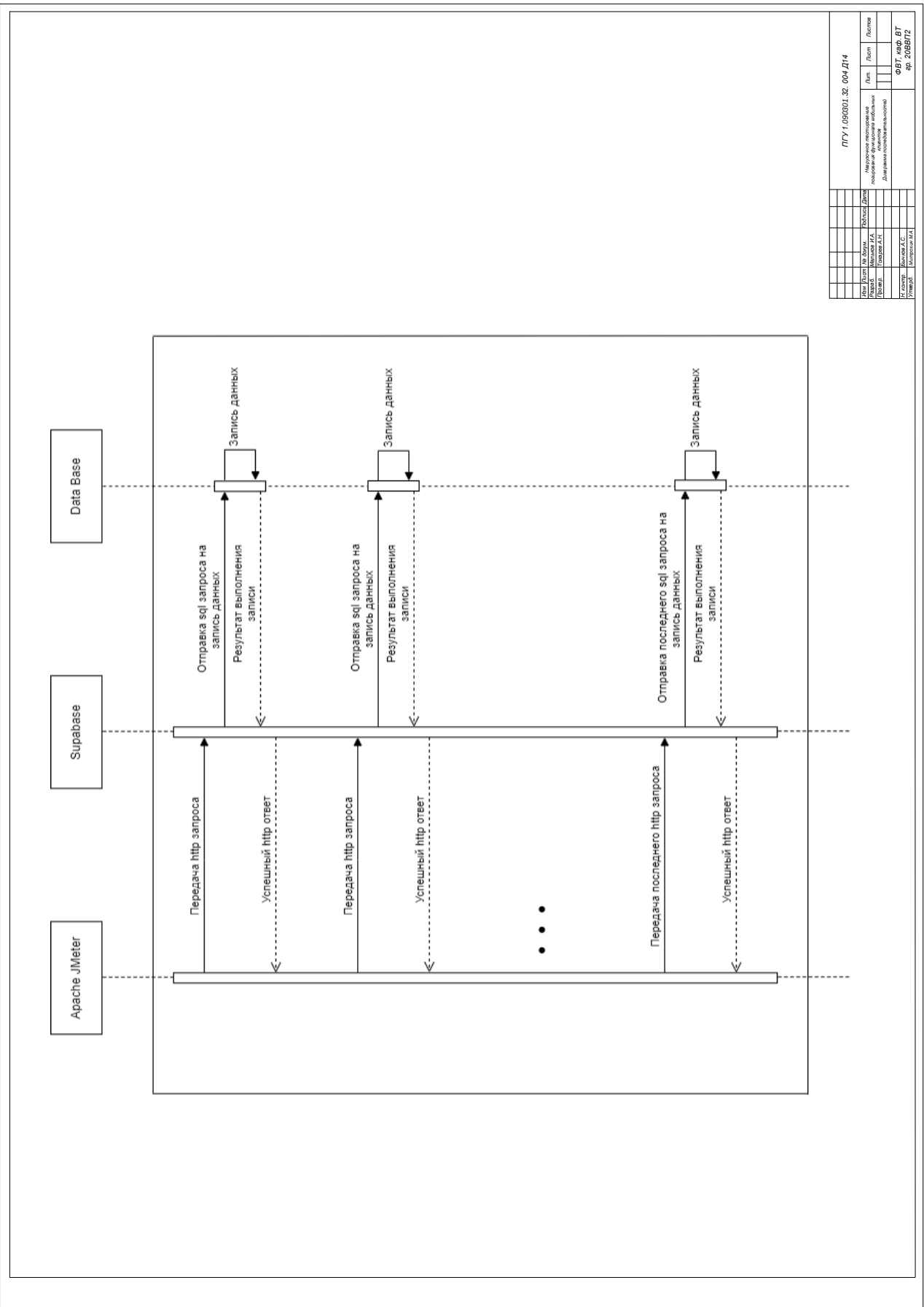
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.32.001 ПЗ

Лист

63

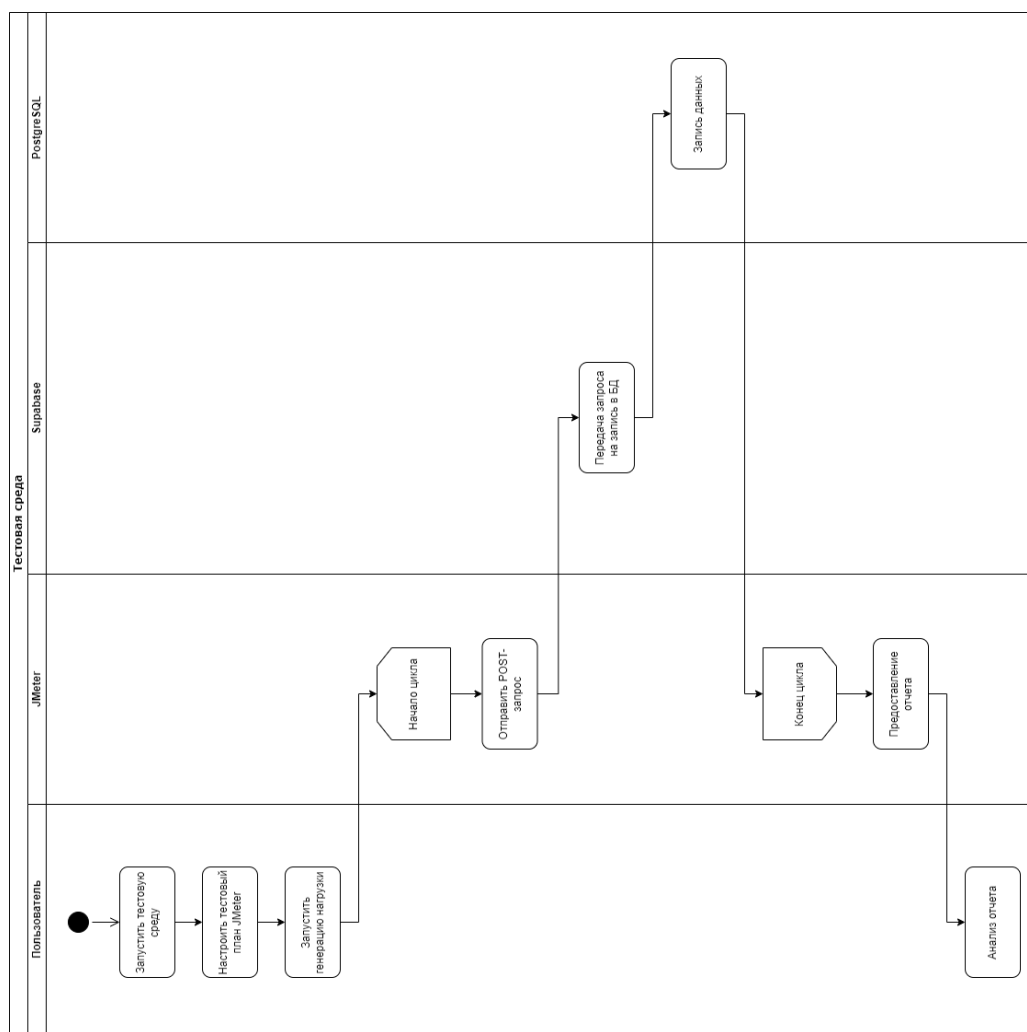
Приложение А.3 – UML-диаграмма последовательностей



ПГУ 1.090301.32.004 Д14			
Имя	Имя	Имя	Имя
Фамилия	Фамилия	Фамилия	Фамилия
Должность	Должность	Должность	Должность
Подпись	Подпись	Подпись	Подпись
Дата	Дата	Дата	Дата
Лист	Лист	Лист	Лист
ФВТ, кэф. ВТ	ФВТ, кэф. ВТ	ФВТ, кэф. ВТ	ФВТ, кэф. ВТ
Фр. 20ВВ/12	Фр. 20ВВ/12	Фр. 20ВВ/12	Фр. 20ВВ/12

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4 – UML-диаграмма деятельности



ПГУ 1.090301.32.005 Д17									

Приложение Б

(Обязательное)

Нагрузочное тестирование логирования функционала мобильных клиентов

Curl-запрос на запись лога в базу данных

					ПГУ 1.090301.32.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66

```

curl 'http://localhost:8000/api/pg-meta/default/query?key=' -X POST
-H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/118.0' -H 'Accept: application/json' -H
'Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3' -H 'Accept-
Encoding: gzip, deflate, br' -H 'Referer:
http://localhost:8000/project/default/editor/25264' -H 'content-
type: application/json' -H 'x-connection-encrypted: ' -H 'x-request-
id: 12f5fd34-df64-422d-a71e-879bbb36b36f' -H 'Origin:
http://localhost:8000' -H 'Authorization: Basic
c3VwYWJhc2U6dGhpc19wYXNzd29yZF9pc19pbmNlY3VyZV9hbmRfc2hvdWxkX2JlX3Vw
ZGF0ZWQ=' -H 'Connection: keep-alive' -H 'Sec-Fetch-Dest: empty' -H
'Sec-Fetch-Mode: cors' -H 'Sec-Fetch-Site: same-origin' --data-raw
$ '{"query": "insert into public.\\\"Log\\\" (\\\"full
name\\\",snils,organization,dump_with_log,status,product,application_
version,\\\"OS\\\",\\\"OS_version\\\",firmware,device,\\\"RAM_capacity\\\"
,amount_of_free_memory,storage_capacity,free_space_storage,\\\"AUPD_t
oken\\\")
select \\\"full
name\\\",snils,organization,dump_with_log,status,product,application_
version,\\\"OS\\\",\\\"OS_version\\\",firmware,device,\\\"RAM_capacity\\\"
,amount_of_free_memory,storage_capacity,free_space_storage,\\\"AUPD_t
oken\\\" from jsonb_populate_recordset(null::public.\\\"Log\\\",
' [{\\\"full
name\\\":\\\"\\u041f\\u0435\\u0442\\u0440\\u043e\\u0432
\\u041f\\u0435\\u0442\\u0440\\u043e\\u0432\\u0438\\u0447\\\",\\\"snils\\\":\\\"8
43
607
021
93\\\",\\\"organization\\\":\\\"iteco\\\",\\\"dump_with_log\\\":\\\"https://
teamtest.r7-
office.ru:3443\\\",\\\"status\\\":\\\"\\u041d\\u043e\\u0432\\u043e\\u0435\\\",
\\\"product\\\":\\\"\\u0416\\u0443\\u0440\\u043d\\u0430\\u043b\\\",\\\"applicat
ion_version\\\":\\\"1.3.0-beta-2455 Thu Nov 09 21:54:04 GMT+03:00
2023\\\",\\\"OS\\\":\\\"Android\\\",\\\"OS_version\\\":\\\"Android
10\\\",\\\"firmware\\\":\\\"29\\\",\\\"device\\\":\\\"Xiaomi Redmi Note
7\\\",\\\"RAM_capacity\\\":\\\"2734
MB\\\",\\\"amount_of_free_memory\\\":\\\"2055
MB\\\",\\\"storage_capacity\\\":\\\"21789
MB\\\",\\\"free_space_storage\\\":\\\"8535
MB\\\",\\\"AUPD_token\\\":\\\"13456543213245432\\\"}]]\\\";"}'

```

Приложение В

(Обязательное)

Нагрузочное тестирование логирования функционала мобильных клиентов

Презентация

					ПГУ 1.090301.32.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

Нагрузочное тестирование логирования функционала мобильных клиентов

- Автор работы: студент группы 20ВВП2 Мальков И.А.
- Научный руководитель: к.т.н., доцент кафедры ВТ Токарев А.Н.

Цели работы

- изучение систем нагрузочного тестирования
- создание программной системы взаимодействия компонентов тестирования
- реализации проверки производительности системы под нагрузкой

					ПГУ 1.090301.32.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

Актуальность

Логи — это файлы, содержащие информацию о работе какой-либо программы, в которые вносятся определённые действия пользователя или самой программы.

Логирование нужно для анализа использования функционала ПО и отслеживания показателей и действий во время неполадок. Это помогает в кратчайшие сроки разобраться с причинами ошибок и устранить их.

Так как в система используется большим количеством пользователей, могут случаться внештатные ситуации, влекущие за собой повышение нагрузки на определенные функции. Так, во время сбоя, все клиенты, которых он затронул, отправляют логи на сервер, создавая большое количество одинаковых запросов.

Для того, чтобы проверить, сможет ли система безотказно работать в условиях повышенной нагрузки существует тестирование производительности. В данной работе будет выполнено нагрузочное тестирование функции логирования, для проверки соответствия системы этим требованиям.

3

Аналоги

- Firebase
- Яндекс Метрика



Ключевые недостатки:

- Невозможность развернуть локально
- Зависимость

4

					ПГУ 1.090301.32.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

Выбранный инструмент

Supabase



Ключевые достоинства:

- Возможность развернуть локально
- Встроенная система безопасности
- Отличная документация и активное сообщество

5

Инструмент тестирования

Apache JMeter



Достоинства:

- Лицензия с открытым исходным кодом.
- Удобный графический интерфейс
- Гибкая настройка тестовых сценариев

Аналоги:

- Gatling
- LoadRunner



6

					ПГУ 1.090301.32.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

Характеристики стенда

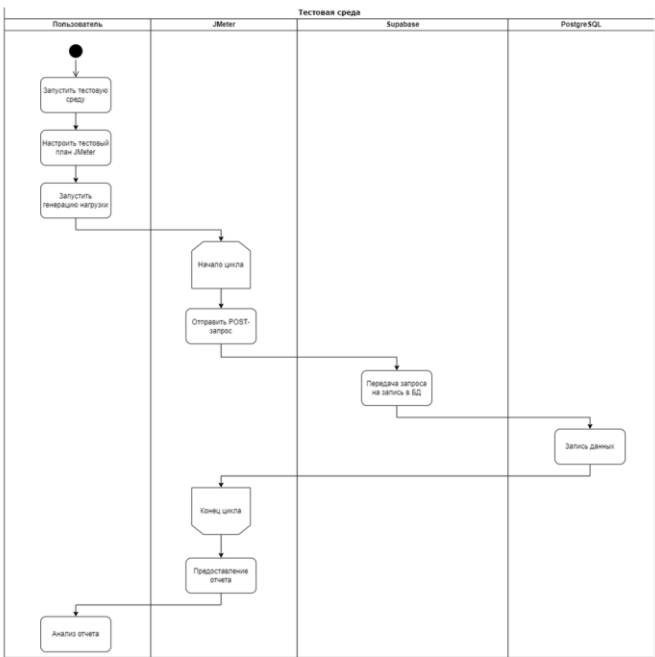
- Процессоры: 4 CPU
- Оперативная память: 4 Гб
- Дисковое пространство: 50 Гб

Цель тестирования – проверить стабильность системы при пиковой нагрузке в 125 запросов в секунду.

Логирование – запись информации в базу данных.
Сценарий: отправка данных лога через POST-запрос

7

UML-диаграмма деятельности



8

Разработка скрипта

Профиль нагрузки:

- Параллельных потоков пользователей: 50 потоков
- Средний вес лога: 62,7 кб
- Длительность нагрузки: 10 минут

Лог содержит следующие поля: продукт, версия приложения, ОС , устройство, версия ОС, прошивка, объем оперативной памяти, объем свободной памяти, объем хранилища, объем свободного пространства хранилища, ID профиля, Организация, дамп с логом, статус, время создания записи.

9

Определение требуемой нагрузки

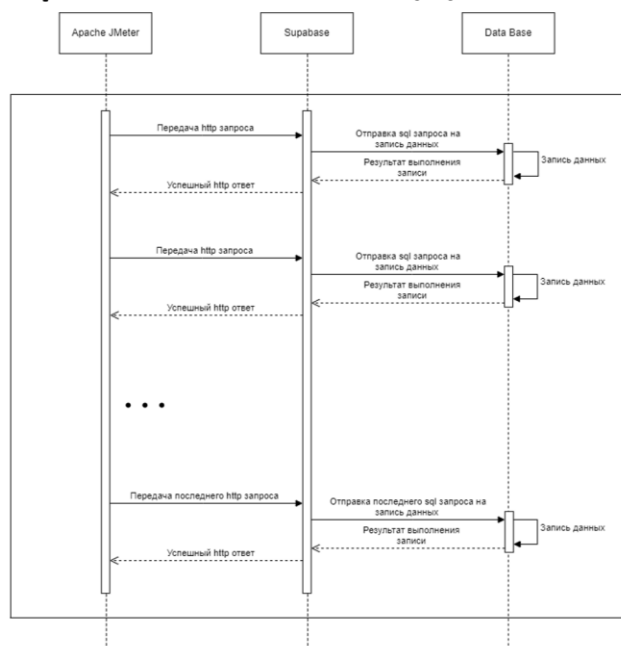
- Общее количество пользователей в субъекте: 1 500 000 человек
- Процент участников которых, затронут сбой: 10%
- Длительность сбоя: 20 минут

Исходя из этих данных требуемый пиковый TPS должен быть равен 125 транзакций в секунду.

10

					ПГУ 1.090301.32.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

UML-диаграмма последовательности



11

Результаты тестирования

Table Editor														
localhost:8000/project/default/editor/698762														
WARNING: You are allowing anonymous access to your table. Enable Row Level Security														
	id	pro...	appl...	OS	dev...	OS	firm...	RAM	mem...	stora...	free...	AUP...	full...	tot...
	74378	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74379	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74380	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74381	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74382	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74383	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74384	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74385	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74386	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74387	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74388	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74389	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74390	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74391	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74392	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74393	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74394	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74395	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
	74396	Журнал	1.3.0-beta-2455	Android	Xiaomi Redmi N	Android 10	29	2734 MB	2055 MB	21789 MB	8535 MB	1345654321324	Петров Пе	
Page 1 of 799 100 rows 79 866 records														

12

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.32.001 ПЗ

Лист

74

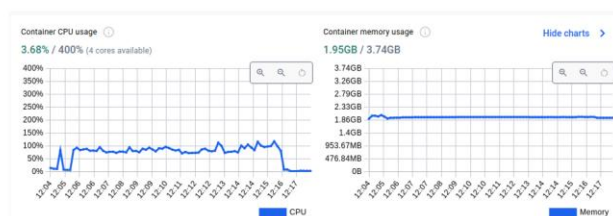


График использования ресурсов

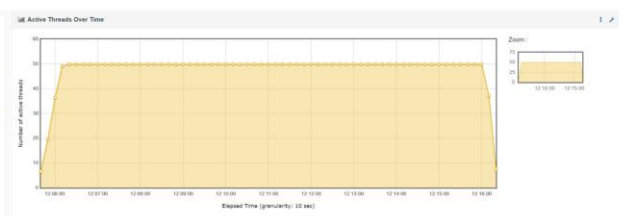


График активности потоков

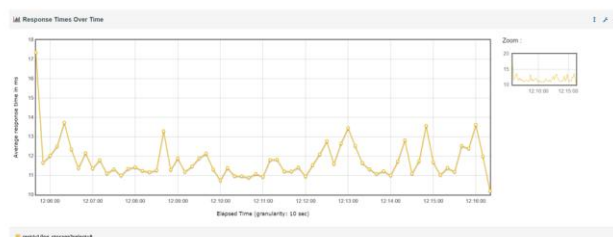
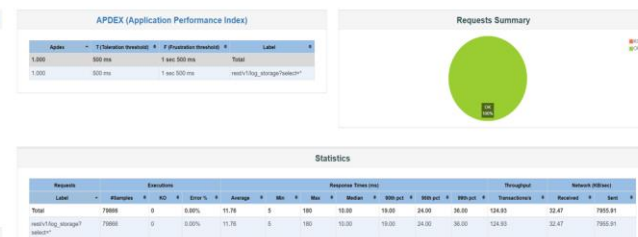


График времени отклика



Общая статистика

13

Заключение

В ходе выполнения данной бакалаврской работы была достигнута основная цель — изучение систем нагрузочного тестирования и разработка программной системы для проверки производительности системы логирования мобильных клиентов под нагрузкой.

Были рассмотрены и проанализированы современные инструменты для нагрузочного тестирования, такие как Apache JMeter, Supabase.

Результатом стало доказательство того, что Supabase отвечает требованиям производительности и подходит для использования в качестве системы хранения логов.

14

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.32.001 ПЗ

Лист

75