**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования «Кабардино-Балкарский государственный университет**

**им. Х.М. Бербекова» (КБГУ)**

**ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И ЦИФРОВЫХ ТЕХНОЛОГИЙ**

**КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**КУРСОВАЯ РАБОТА**

по дисциплине

Разработка программных приложений:

**«Разработка приложения для анализа log-файлов»**

|  |
| --- |
| Выполнил:  студент 3 курса направления  09.03.03 Прикладная информатика  (профиль «Корпоративные информационные системы») ОФО |
| Каншау Джанти |
|  |
| Руководитель:  доцент кафедры КТиИБ  Елеев И.З. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Нальчик 2023**

**Содержание**

[**Введение** 3](#_Toc154739266)

[**1.** **Аналитическая часть** 3](#_Toc154739267)

[1.1 Концепция и требования приложения 3](#_Toc154739268)

[1.2 Системный проект. 3](#_Toc154739269)

[1.3 Выбор инструментария разработки 3](#_Toc154739270)

[1.4 Управление проектом. Выбор жизненного цикла проекта 7](#_Toc154739275)

**Введение**

В современном информационном обществе огромное количество данных непрерывно генерируется и накапливается. Этот поток информации охватывает различные аспекты нашей жизни, включая взаимодействие с компьютерными системами и приложениями. Одним из наиболее значимых источников данных, особенно в области информационных технологий, являются лог-файлы.

Лог-файлы представляют собой записи событий, происходящих в системе или приложении, и предоставляют важную информацию о её работе. Однако, объем и сложность этих данных часто делает их анализ нетривиальной задачей. В связи с этим возникает необходимость в разработке специализированных инструментов и приложений, направленных на обработку и анализ лог-файлов.

Цель данной курсовой работы - разработать приложение для анализа лог-файлов, которое позволит эффективно извлекать ценную информацию из больших объемов данных, представленных в формате логов. В рамках работы будут рассмотрены основные принципы анализа логов, методы обработки и визуализации данных, а также разработка и реализация программного решения для автоматизации процесса анализа.

Данная работа представляет актуальный научно-исследовательский интерес, так как обеспечивает практическое применение теоретических знаний в области анализа данных и разработки программного обеспечения.

Кроме того, разработанное приложение может быть полезным инструментом для администраторов систем, разработчиков программного обеспечения и специалистов по информационной безопасности для повышения эффективности и оперативности работы с лог-данными.

# **Аналитическая часть**

* 1. Концепция и требования приложения

**Описание первоначальной концепции цифрового сервиса**

Целью разработки данного приложения является создание инструмента для анализа лог-файлов, обеспечивающего оперативное выявление аномалий, мониторинг работы системы и обнаружение потенциальных проблем. Приложение будет предоставлять пользователям возможность эффективно обрабатывать большие объемы лог-данных, проводить аналитику событий и получать визуализированные отчеты для принятия обоснованных решений.

**Модель бизнес-требований**

* Эффективность анализа: Приложение должно обеспечивать быстрый и точный анализ лог-файлов, позволяя пользователям оперативно реагировать на события и проблемы.
* Гибкость и масштабируемость: Приложение должно поддерживать работу с различными форматами и источниками лог-данных, а также масштабироваться для работы с большими объемами данных.
* Безопасность: Обеспечение конфиденциальности и целостности данных, а также защита от несанкционированного доступа к приложению и его функционалу.

**Модель требований высокого уровня к приложению**

* Интерфейс пользователя: Интуитивно понятный и удобный интерфейс для загрузки лог-файлов, настройки параметров анализа и просмотра результатов.
* Система управления данными: Механизмы для хранения, поиска и фильтрации лог-данных, обеспечивающие высокую производительность и эффективность работы.
* Модуль анализа и визуализации: Реализация алгоритмов анализа данных, включая поиск аномалий, генерацию отчетов и визуализацию результатов.
* Интеграция с другими системами: Возможность интеграции с существующими информационными системами и сервисами для обмена данными и автоматизации процессов.

**Модель требований пользователей к приложению**

* Администраторы систем: Необходимость мониторинга работы системы, обнаружения аномалий и выявления проблем для обеспечения ее стабильной работы.
* Разработчики приложений: Потребность в анализе лог-данных для отладки и оптимизации работы приложений, а также для улучшения пользовательского опыта.
* Специалисты по безопасности: Важность обнаружения атак и аномалий в системе, анализа уязвимостей и разработки мер по их предотвращению.

**Модель функциональных требований к приложению**

* Загрузка лог-файлов: Возможность загрузки лог-файлов из различных источников и форматов.
* Поиск и фильтрация данных: Функционал для поиска и фильтрации лог-событий по заданным критериям.
* Анализ данных: Реализация алгоритмов анализа для выявления аномалий, трендов и закономерностей в данных.
* Визуализация результатов: Предоставление пользователю удобных инструментов для визуализации результатов анализа в виде графиков, диаграмм и отчетов.
* Уведомления и оповещения: Возможность настройки оповещений и уведомлений о выявленных аномалиях и важных событиях.
* Экспорт данных: Возможность экспорта результатов анализа в различные форматы для дальнейшего использования или обмена с другими системами.

## **1.2 Системный проект.**

**Компоненты системы:**

**Интерфейс пользователя (UI):**

* Отвечает за отображение пользовательского интерфейса и взаимодействие с пользователем.
* Реализует функции загрузки лог-файлов, настройки параметров анализа и отображения результатов.
* Может быть разработан с использованием стандартных библиотек Go для GUI (например, Walk, Fyne) или веб-интерфейса с помощью библиотеки Go-HTML-Template.

**Модуль анализа данных:**

Отвечает за обработку и анализ лог-данных.

* Реализует алгоритмы анализа, поиск аномалий, корреляцию событий и генерацию отчетов.
* Использует библиотеки Go для обработки данных и статистического анализа (например, gonum, Gota).

**Модуль управления данными:**

* Обеспечивает сохранение и доступ к лог-файлам.
* Может включать в себя функции загрузки, хранения и поиска лог-данных.
* Использует стандартные библиотеки Go для работы с файловой системой и управления данными.

**Модуль уведомлений:**

* Отвечает за отправку уведомлений и оповещений пользователей о важных событиях.
* Реализует механизмы для настройки и отправки уведомлений (например, по электронной почте, через API сторонних сервисов).
* Может использовать стандартные пакеты Go для работы с сетью и отправки почты.

**Архитектурные решения:**

Модульная архитектура: Разделение функциональности на независимые модули для обеспечения удобства сопровождения и расширения приложения.

Многопоточная обработка данных: Использование горутин для параллельной обработки данных и повышения производительности приложения.

Использование стандартных библиотек Go: Максимальное использование стандартных библиотек Go для обеспечения эффективности и надежности приложения.

Тестирование и обработка ошибок: Реализация тестов для каждого компонента приложения и обработка ошибок для обеспечения надежности работы приложения.

Управление зависимостями: Использование модульной системы управления зависимостями Go (например, Go Modules) для эффективного управления зависимостями и версиями библиотек.

Такая архитектура позволит разрабатывать и поддерживать десктопное приложение для анализа логов на Go с высокой производительностью и удобством использования.

**1.3 Выбор инструментария разработки.**

При разработке десктопного приложения на Go с использованием Fyne в качестве фреймворка для создания пользовательского интерфейса, необходимо учитывать следующий инструментарий:

**Язык программирования:**

Go: Язык программирования, на котором будет написано основное приложение.

**Фреймворк для пользовательского интерфейса:**

Fyne: Фреймворк для создания кроссплатформенных приложений с графическим интерфейсом на Go.

**Инструменты разработки:**

Visual Studio Code с Go Extension: Интегрированная среды разработки (IDE), предоставляющая удобные инструменты для написания кода на Go, отладки и управления зависимостями проекта.

GitHub: Для управления исходным кодом, совместной работы над проектом и контроля версий.

**Управление зависимостями:**

Go Modules: Система управления зависимостями Go, позволяющая управлять зависимостями проекта и обеспечивать их совместимость и актуальность.

**Тестирование и отладка:**

Стандартные инструменты Go: Встроенные инструменты для тестирования и отладки Go-приложений, такие как go test и go debug.

**Дизайн и прототипирование:**

Figma: Инструмент для создания дизайна пользовательского интерфейса и прототипирования.

**Документация:**

Markdown: Для написания документации и README-файлов проекта.

**Тестирование на различных платформах:**

Windows: Для проверки совместимости приложения.

## **1.4 Управление проектом. Выбор жизненного цикла проекта**

Выбор жизненного цикла веб-сайта является важным этапом, так как именно от него будет зависеть подход, который мы будем применять для управления проектом. Необходимо понимать, что универсального цикла не существует. Однако представляется возможным выбор подходящего ИС после сравнения различных моделей. Так как веб-сайт будет усовершенствоваться со течением времени, необходимо выбрать гибкий режим реализации проекта.

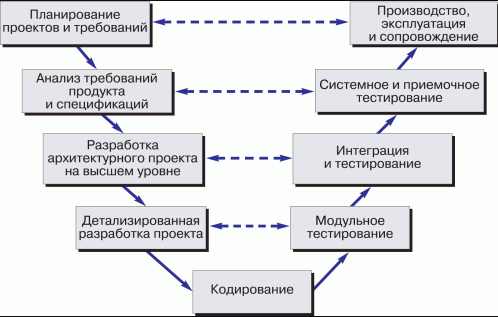
Главной отличительной чертой каскадной модели является то, не представляется возможным переход на следующий этап разработки до завершения предыдущего. Из этого следует, в нашем случае выбор данного жизненного цикла будет неразумным, из-за принимается решение отказаться от этой модели. Она не предоставляет нам необходимую гибкость.



*Рисунок 1.5.1 – Каскадная модель жизненного цикла*

Следующей моделью становится V-образная. V-модель или разработка через тестирование.

Данная модель имеет более приближенный к современным методам алгоритм, однако все еще имеет ряд недостатков. Является одной из основных практик экстремального программирования. Уже на стадии написания требований начинается процесс тестирования. Поэтому данная модель нам не подходит, поскольку нет возможности перехода на следующий этап, не убедившись в работоспособности предыдущего этапа.

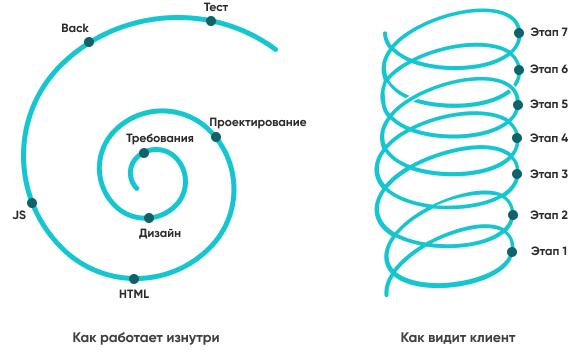


*Рисунок 1.5.2 – V-образная модель жизненного цикла*

В спиральной модели жизненный путь разрабатываемого продукта изображается в виде спирали, которая, начавшись на этапе планирования, раскручивается с прохождением каждого следующего шага.

Преимуществами являются быстрое получение результата, повышение конкурентоспособность, а также если в какой-то момент нужны будут изменения, для спиральной модели это не проблема.

В нашем проекте спиральная модель будет намного эффективна, так как у нас всегда будет возможность внести изменения в проект, больше возможности планирования различных методов на каждом из этапов разработки.



*Рисунок 1.5.3 – Спиральная модель жизненного цикла*

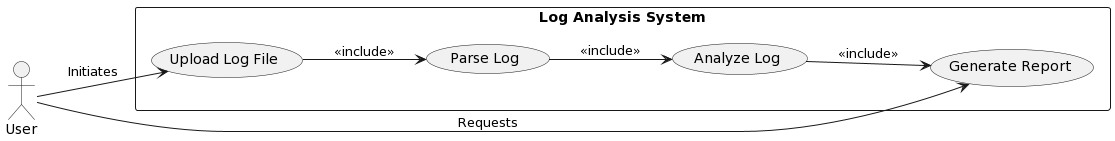
# **Практическая часть**

* 1. Разработка UML модели проекта

Для моделирования процессов в системе была использована диаграмма прецедентов в формате UML. На основе анализа требований выделены следующие прецеденты:

* Загрузка файла журнала (UC1)
* Разбор файла журнала (UC2)
* Анализ содержимого журнала (UC3)
* Генерация отчета (UC4)

Диаграмма прецедентов выглядит следующим образом:



*Рисунок 2.1.1 – Диаграмма прецедентов*

При проведении обзора было выявлено, что для хранения и обработки системных логов часто применяют реляционные базы данных, такие как MySQL, PostgreSQL и другие. Также было обнаружено, что структура базы данных должна быть гибкой и удобной для анализа данных, а также обеспечивать защиту информации о пользователях.

Для разработки базы данных был использован метод анализа требований и метод проектирования баз данных. Этапы проектирования включают в себя выявление сущностей, их атрибутов и связей, а также определение первичных и внешних ключей.

Проектирование базы данных:

Сущность Log (Журнал):

* log\_id: Идентификатор записи (первичный ключ).
* timestamp: Временная метка события.
* message: Сообщение о событии.
* severity: Уровень серьезности события.

Сущность User (Пользователь):

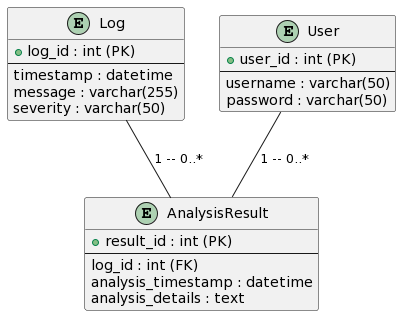
* user\_id: Идентификатор пользователя (первичный ключ).
* username: Имя пользователя.
* password: Хэшированный пароль пользователя.

Сущность AnalysisResult (Результат анализа):

* result\_id: Идентификатор результата анализа (первичный ключ).
* log\_id: Идентификатор записи журнала (внешний ключ, связывает сущность AnalysisResult с Log).
* analysis\_timestamp: Временная метка анализа.
* analysis\_details: Подробности результатов анализа.

Связи между сущностями:

* Каждая запись в таблице AnalysisResult связана с одной записью в таблице Log (связь "один ко многим").
* Каждая запись в таблице AnalysisResult связана с одним пользователем в таблице User (связь "один ко многим").



*Рисунок 2.1.2 – Диаграмма базы данных*

Был проведен анализ существующих систем анализа логов и их взаимодействия с пользователями. Было выявлено, что использование диаграммы последовательности позволяет наглядно представить последовательность действий и коммуникаций между участниками системы.

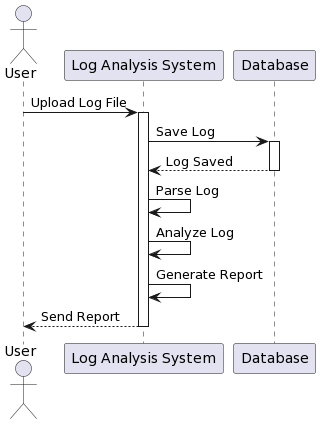
Для анализа взаимодействия между пользователями и системой анализа логов была использована методика моделирования диаграммы последовательности. Этапы методики включают в себя идентификацию участников системы, определение последовательности их действий и взаимодействий, а также анализ этой последовательности с целью оптимизации процессов.

Участники:

* Пользователь: инициирует анализ логов, загружая файлы журналов в систему.
* Система анализа логов: осуществляет обработку загруженных файлов, проводит анализ и генерирует отчеты.

Последовательность действий:

* Пользователь загружает файл журнала в систему анализа логов.
* Система анализа логов сохраняет файл в базе данных.
* Система разбирает содержимое файла журнала.
* Система проводит анализ содержимого журнала.
* Система генерирует отчет на основе результатов анализа.
* Система отправляет сгенерированный отчет пользователю.



*Рисунок 2.1.3 – Диаграмма последовательности*

При проведении обзора были рассмотрены существующие методики проектирования баз данных для систем анализа логов. Было обнаружено, что использование диаграмм классов позволяет наглядно представить структуру данных и их взаимосвязи.

Для разработки структуры базы данных была использована методика построения диаграмм классов. Этапы методики включают в себя идентификацию основных сущностей, определение их атрибутов и взаимосвязей, а также построение самой диаграммы классов.

Класс Log (Журнал):

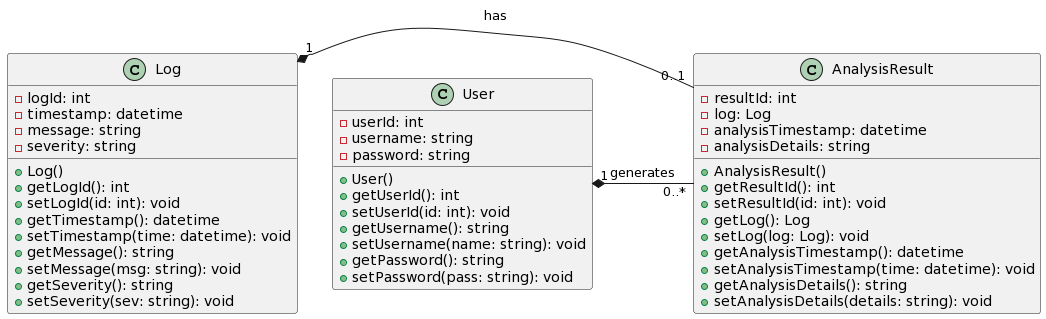
* Атрибуты: logId, timestamp, message, severity.
* Методы: конструктор, геттеры и сеттеры для каждого атрибута.

Класс User (Пользователь):

* Атрибуты: userId, username, password.
* Методы: конструктор, геттеры и сеттеры для каждого атрибута.

Класс AnalysisResult (Результат анализа):

* Атрибуты: resultId, log (ссылка на объект Log), analysisTimestamp, analysisDetails.
* Методы: конструктор, геттеры и сеттеры для каждого атрибута.



*Рисунок 2.1.4 – Диаграмма классов*

Проведенный обзор показал, что диаграммы деятельности широко используются для моделирования процессов в различных областях, включая анализ данных и обработку информации. Они позволяют наглядно представить последовательность шагов и принимаемые решения в процессе выполнения задачи.

Для моделирования процесса анализа системных логов была использована методика построения диаграммы деятельности. Этапы методики включают в себя определение начального и конечного состояний процесса, выделение основных шагов и принимаемых решений, а также построение самой диаграммы.

Начальное состояние (Start):

* Начало процесса анализа системных логов.

Шаги процесса:

Проверка загрузки пользователем файла журнала.

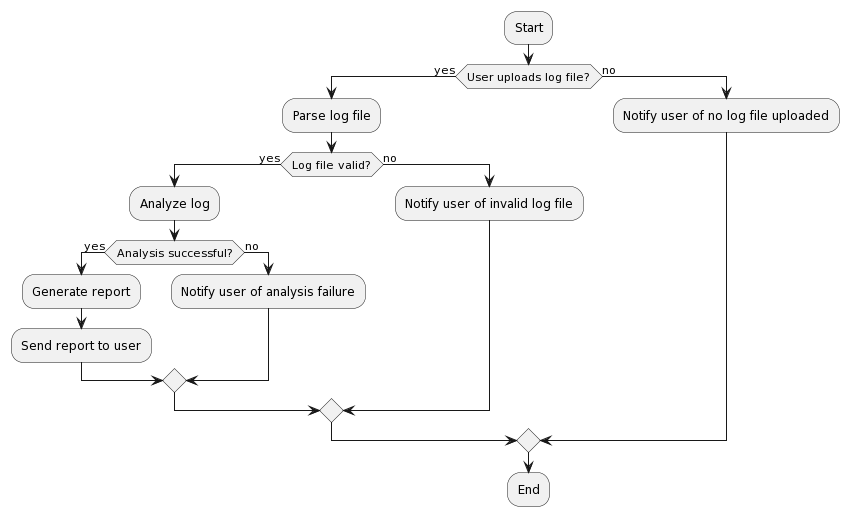
* Разбор загруженного файла журнала.
* Проверка валидности файла журнала.
* Анализ содержимого файла журнала.
* Генерация отчета на основе результатов анализа.

Исходы процесса:

* Успешное завершение анализа с генерацией отчета.
* Ошибка в процессе анализа с уведомлением пользователя.
* Ошибка при загрузке файла журнала с соответствующим уведомлением.

Конечное состояние (End):

* Завершение процесса анализа системных логов.



*Рисунок 2.1.5 – Диаграмма деятельности*