Санкт-Петербургский государственный университет

Кафедра Системного программирования

Группа 24.М71-мм

Разработка веб-приложения анализа повторов в документациях для проекта DocLine

Макаров Павел Михайлович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к.ф.-м.н., Луцив Д. В.

Санкт-Петербург
2025

Оглавление

Введение **3**

1. Постановка задачи **4**

2. Обзор **5**

2.1. Обзор существующего решения 5

2.2. Обзор используемых технологий 6

3. Реализация **9**

3.1. Серверная часть 9

3.2. Клиентская часть 10

3.3. Асинхронная обработка задач 11

3.4. Структура инструмента 12

4. Тестирование **14**

4.1. Аппаратно-программная конфигурация 14

4.2. Методика тестирования 14

4.3. Результат тестирования 14

Заключение **15**

Список литературы **16**

Введение

Документация, несмотря на кажущуюся второстепенность, является одним из ключевых элементов любого приложения, библиотеки или утилиты. Отсутствие качественной документации значительно усложняет работу с кодом, делая его менее доступным для понимания и использования. В то же время ведение документации позволяет разработчикам упрощать собственную работу, используя схожие пояснения для описания функционала.

Хотя это упрощает процесс создания документации, возникает проблема некорректности, когда часть информации, относящейся к одному понятию, дублируется в другом месте, где такой функционал не реализован. Это может приводить к недопониманиям между разработчиками и пользователями в вопросах реализации, возможностей и областей применения функций.

Для решения подобных проблем существует проект DocLine, направленный на создание инструментария для повторного использования документации [1]. В рамках проекта был разработан инструмент Duplicate Finder, предназначенный для поиска и анализа повторов в документации. Однако из-за сумбурной реализации (использование устаревших библиотек и различных языков программирования) использование Duplicate Finder сопряжено с проблемами совместимости с современными устройствами, и в дальнейшем это будет усугубляться.

Таким образом, возникает необходимость в создании обновлённого подхода к поиску и анализу повторов в документации, который будет более функциональным, современным и удобным для пользователей.

1. Постановка задачи

Целью работы является реинжиниринг инструмента Duplicate Finder с созданием его новой версии, ориентированной на улучшение функциональности, производительности и удобства использования. В рамках выполнения данной задачи были поставлены следующие цели и задачи:

1. Провести анализ механизма работы существующей реализации Duplicate Finder.
2. Разработать предварительный алгоритм работы обновленного Duplicate Finder на основе анализа текущей реализации, с учетом улучшений в плане функциональности и удобства использования.
3. Реализовать прототипы инструмента в виде веб-приложения.
4. Провести тестирование прототипов обновлённого инструмента.
5. На основе результатов тестирования провести анализ выявленных ошибок, доработать инструмент, устранить дефекты и улучшить его функциональность.
6. Подготовить техническую документацию для нового инструмента.

2. Обзор

2.1. Обзор существующего решения

2.1.1. DocLine

В проекте DocLine уже существует Duplicate Finder, который выполняет задачу по поиску повторов в документации.[1] Однако в текущем виде данная утилита имеет ряд серьёзных ограничений:

1. Утилита требует устаревшую версию Python 3.7 из-за проблем совместимости с PyQt, что затрудняет переход на более современные версии Python.

2. В функционировании утилиты участвуют различные языки программирования, такие как Python и C#, что усложняет поддержку и развитие проекта.

3. Поддерживается только платформа Windows. Для запуска на Linux требуется Wine.

4. Сложность в создании и использовании портативной версии, что затрудняет развёртывание утилиты на новых системах.

Эти проблемы создают потребность в разработке альтернативного решения, которое устранит перечисленные недостатки и предоставит пользователям более удобный и функциональный инструмент.

2.1.2. Squirro Cognitive Search

Squirro — это корпоративная система, использующая генеративный искусственный интеллект (GenAI) для преобразования данных организаций в действенные инсайты, способствуя принятию обоснованных решений и оптимизации рабочих процессов [6].

Squirro Cognitive Search — это интеллектуальная поисковая система, использующая технологии искусственного интеллекта (AI) и машинного обучения (ML) для улучшения традиционного корпоративного поиска. Её цель — помочь пользователям быстро находить наиболее

релевантную информацию во всех подключенных источниках данных, понимая контекст и намерения запроса.

Squirro Cognitive Search интегрируется в организацию через платформу Squirro, предоставляя интеллектуальные возможности поиска и анализа данных. Для использования Squirro Cognitive Search необходимо установить соответствующее приложение через Squirro Marketplace.

Несмотря на наличие функции поиска повторов, Squirro Cognitive Search не ориентирован исключительно на работу с документацией. Кроме того, этот инструмент является проприетарным программным обеспечением, что ограничивает его использование.

2.1.3. Near Duplicates Finder

Near Duplicates Finder — это программа на Java, предназначенная для нахождения дубликатов и схожих текстовых документов на основе анализа их содержимого. Инструмент предоставляет отчёты, которые могут быть использованы для дальнейших действий.[4]

Хотя Near Duplicates Finder является открытым программным обеспечением (полный код выложен на GitHub), основной акцент сделан на общем анализе текстовых данных, а не на специфических задачах работы с документацией, как и Squirro Cognitive Search.

2.2. Обзор используемых технологий

2.2.1. Go(Golang)

Go (Golang) — это компилируемый язык программирования с открытым исходным кодом, разработанный компанией Google[2]. Его основные преимущества включают:

1. Простоту изучения и использования, что ускоряет процесс разработки.

2. Высокую производительность благодаря компиляции в машинный код.

3. Встроенную поддержку многопоточности, что особенно важно для высоконагруженных приложений.

4. Кроссплатформенность, обеспечивающую работу на Windows, Linux и других операционных системах.

5. Высокую скорость компиляции, что облегчает процесс тестирования и развёртывания.

Для реализации обновлённого проекта Golang был выбран благодаря своей эффективности, простоте и отличной интеграции с веб-технологиями.

2.2.2. Pandoc

Pandoc — это универсальная утилита для конвертации документов между различными форматами. Она поддерживает широкий спектр форматов, включая Markdown, LaTeX, docx, txt и другие [5].

Основные преимущества использования Pandoc в реализации:

1. Возможность конвертации документов в удобный для анализа формат.

2. Поддержка локального использования, что исключает необходимость в сторонних онлайн-сервисах.

3. Надёжность и универсальность при работе с текстовыми файлами различных типов.

В реализации Pandoc используется для преобразования документов, упрощая их дальнейшую обработку и анализ.

2.2.3. Handlebars

Handlebars — это простой и мощный язык шаблонов на JavaScript, являющийся расширением Mustache [3]. Он применяется для:

1. Динамического формирования пользовательского интерфейса.

2. Уменьшения дублирования кода и упрощения поддержки веб-страниц.

3. Обеспечения разделения логики и представления, что упрощает разработку.

Handlebars был выбран для проекта как удобный инструмент для создания и обновления шаблонов пользовательского веб-интерфейса, что способствует улучшению взаимодействия с пользователем.

3. Реализация

В рамках учебной практики был создан прототип системы, включающий серверную часть для обработки данных и веб-интерфейс для взаимодействия с пользователем. Основное внимание уделено инженерным решениям, направленным на обеспечение производительности, надёжности и масштабируемости приложения. Прототип демонстрирует ключевые возможности системы, такие как анализ данных, взаимодействие с пользователем и сохранение результатов работы.

3.1. Серверная часть

Серверная часть системы разработана на языке программирования Golang, который обеспечивает высокую производительность, поддержку многопоточности и простоту интеграции с внешними инструментами. Для обработки входящих HTTP-запросов используется стандартная библиотека `net/http`, а для работы с различными форматами документов применяется инструмент Pandoc, что позволяет унифицировать процесс обработки файлов.

Основные реализованные функции серверной части включают:

1. Обработку HTTP-запросов от клиентской части с маршрутизацией на соответствующие обработчики.

2. Приём и валидацию пользовательских данных, включая их преобразование из формата JSON.

3. Логирование полученных данных для отладки и мониторинга работы системы.

4. Асинхронную обработку задач с использованием горутин (goroutin) для обеспечения высокой производительности.

5. Формирование и отправку ответов клиентской части в формате JSON.

Серверная часть спроектирована с использованием REST-архитектуры, что упрощает интеграцию между клиентом и сервером и позволяет масштабировать систему за счёт добавления новых эндпоинтов и функций.

Для каждой функции системы предусмотрены отдельные обработчики:

1. Automatic Mode: обработка запросов с параметрами автоматического анализа.

2. Interactive Mode: обработка запросов для интерактивного взаимодействия с пользователем.

3. N-Gram Finder: обработка данных с использованием метода n-грамм.

4. Heuristic Finder: обработка запросов для анализа с применением эвристических методов.

3.2. Клиентская часть

Клиентская часть представляет собой веб-интерфейс, созданный с использованием HTML, CSS и библиотеки шаблонов Handlebars, что обеспечивает удобство создания динамических веб-страниц. Для взаимодействия с серверной частью используется протокол HTTP с отправкой запросов через REST API.

Основные функции клиентской части:

1. Интуитивно понятный интерфейс для выбора режимов работы системы и настройки параметров анализа, таких как длина текста, уровень фильтрации, выбор языка и других параметров.

2. Передача данных пользователя на серверную часть через POST-запросы с использованием формата JSON.

3. Отображение результатов анализа в удобном формате с возможностью визуализации статистики.

4. Адаптивный дизайн, позволяющий использовать интерфейс на устройствах с разными размерами экрана.

Клиентская часть разработана с учётом расширяемости, что позволяет добавлять новые режимы работы и элементы интерфейса в будущем.

Прототип интерфейса представлен на Рисунке 1.

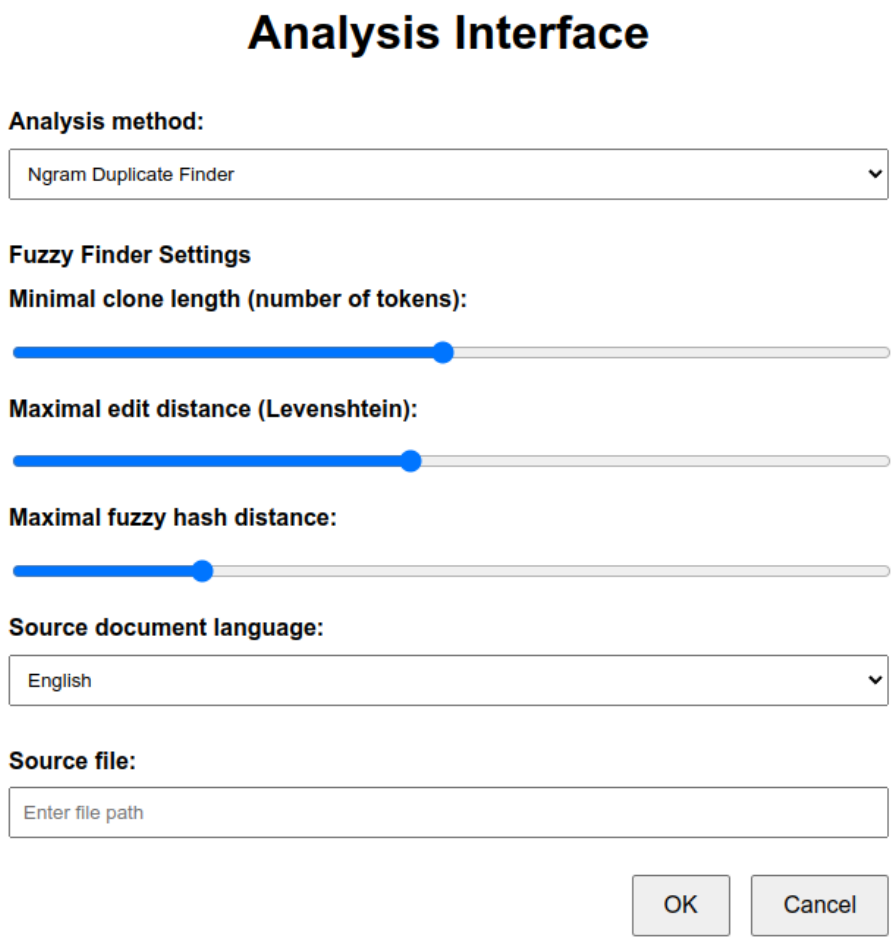

Рис.1 Прототип пользовательского веб-интерфейса

3.3. Асинхронная обработка задач

В серверной части предусмотрена асинхронная обработка задач, связанных с выполнением внешних команд, таких как вызов инструментов анализа данных. Для этого используется механизм горутин (goroutine), который обеспечивает параллельное выполнение задач без блокировки

основного процесса. Это повышает производительность системы и позволяет обрабатывать несколько запросов одновременно.

3.4. Структура инструмента

Для обеспечения наглядности проектных решений и упрощения понимания структуры системы были разработаны диаграммы:

1. Диаграмма вариантов использования (Use Case) представлена на Рисунке 2. Она отражает ключевые сценарии взаимодействия пользователей с системой.

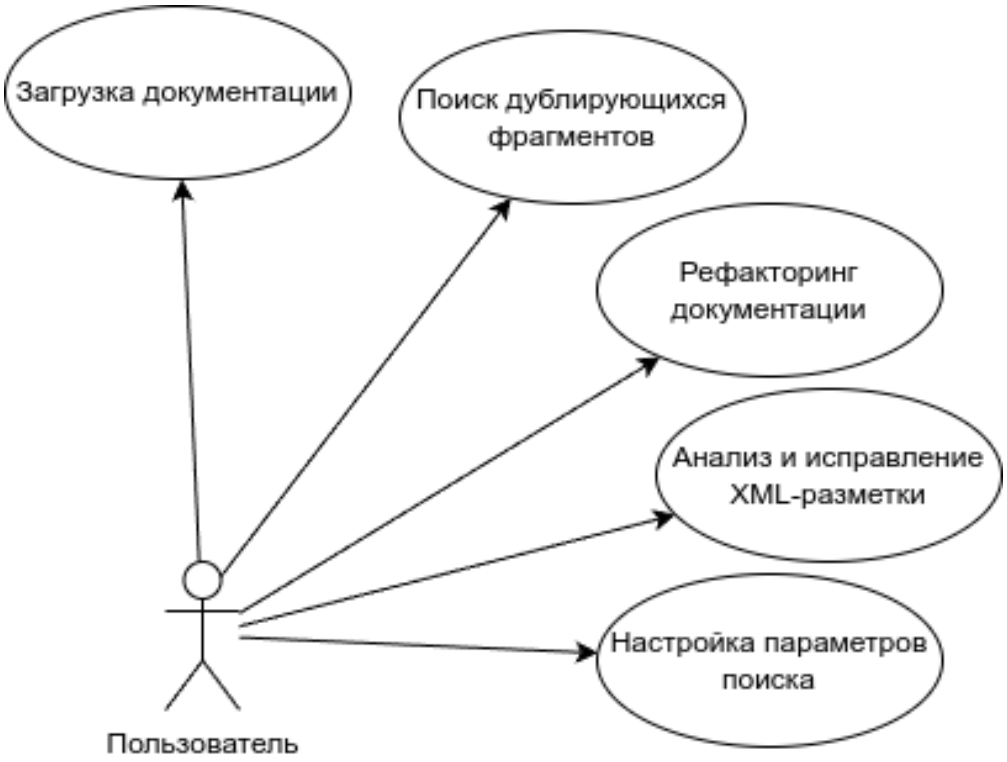

Рис.2 Use Case Diagram

2. Диаграмма компонентов представлена на Рисунке 3. Она демонстрирует основные модули системы и их взаимодействие.

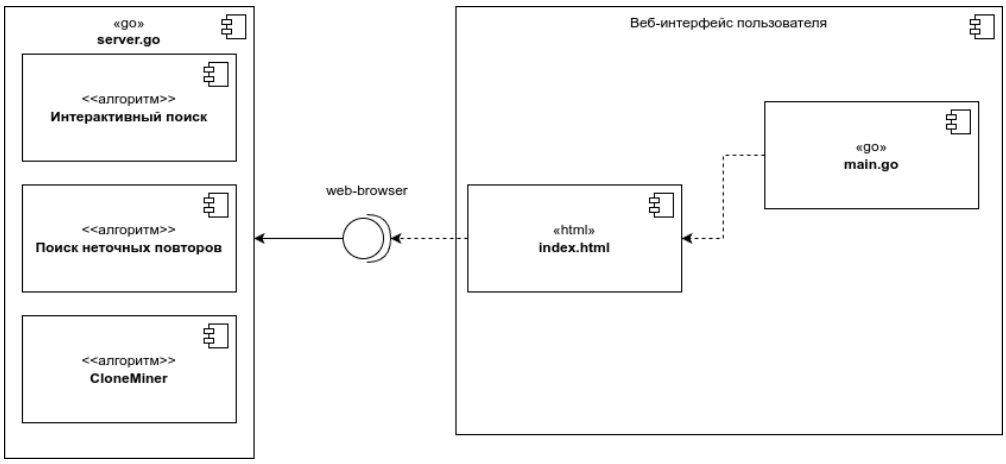

Рис.3 Component Diagram

4. Тестирование

Основная цель – выявить стабильность прототипа, а именно совместимость между текущими используемыми программными инструментами.

4.1. Аппаратно-программная конфигурация

Тестирование прототипа проводилось на устройстве, имеющем следующие характеристики:

- процессор: AMD Ryzen 5 5500U;
- ОЗУ: 16 Гб;
- версия Go: 1.23.3.

Тестирование проходило на 2 ОС:

- Ubuntu 24.10;
- Windows 11.

4.2. Методика тестирования

Тестирование проводилось вручную. Нужно проверить корректность отображения пользовательского веб-интерфейса и функционирование серверной части.

Был запущен основной файл, который перенаправил пользователя на основную веб-страницу интерфейса. После нужно было определить, что серверная часть проекта для анализа данных стабильно запустилась вместе с веб-интерфейсом.

4.3. Результат тестирования

Пользовательский веб-интерфейс и сервер для анализа данных показали стабильность во взаимодействии инструментов.

Заключение

Результатом работы стал прототип обновленного инструмента Duplicate Finder в котором реализованы:

- веб-интерфейс пользователя;
- сервер для обработки данных.

Направления для дальнейшей работы:

1. Перенос функционала со старой версии Duplicate Finder на обновлённую.
2. Улучшение UI и UX.
3. Изучение возможности реализации интерфейса командной оболочки.
4. Корректировка технологического стека.

Исходный код можно найти в репозитории PavelMkr/docline-new.

Список литературы

[1] DocLine. — URL: https://docline.github.io/docline/index.ru (дата обращения: 1 декабря 2024 г.).

[2] GoLang. — URL: https://handlebarsjs.com/guide/ (дата обращения: 15 декабря 2024 г.).

[3] Handlebars. — URL: https://handlebarsjs.com/guide/ (дата обращения: 15 декабря 2024 г.).

[4] Near Duplicates Finder. — URL: https://github.com/e-orlov/neardup (дата обращения: 18 декабря 2024 г.).

[5] Pandoc. — URL: https://go.dev/ (дата обращения: 1 декабря 2024 г.).

[6] Squirro. — URL: https://docline.github.io/docline/index.ru (дата обращения: 18 декабря 2024 г.).