МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка системы поиска шаблонов в данных портала GitHub

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-339.ВКР

Научный руководитель,
профессор кафедры СП, д.ф.-м.н.,
доцент
_____ М.Л. Цымблер

Автор работы,
студент группы КЭ-403
_____ Е.В. Елисеев

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-403
Елисееву Егору Вадимовичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 №764-13/12)

Разработка системы поиска шаблонов в данных портала GitHub.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. – Morgan Kaufmann Publishers, 2012. – 703 p.

3.2. McKinney W. Python for Data Analysis. – O'Reilly Media, 2018. – 540 p.

3.3. GitHub. [Электронный ресурс] URL: https://github.com (дата обращения: 02.02.2024 г.).

3.4. Github Archive. [Электронный ресурс] URL: https://www.gharchive.org/ (дата обращения: 02.02.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ и спецификацию предметной области.

4.2. Выполнить проектирование пользовательского интерфейса и архитектуры системы.

4.3. Реализовать систему поиска шаблонов.

4.4. Провести тестирование системы поиска шаблонов.

4.5. Провести эксперименты с применением разработанной системы.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.ф.-м.н., доцент М.Л. Цымблер

Задание принял к исполнению Е.В. Елисеев

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы

В современном мире IT-индустрии огромные массивы накапливающихся данных играют важнейшую роль в исследованиях, на основе которых принимаются важные бизнес-решения. Данные портала GitHub [0], который является крупнейшим в мире хранилищем кода, включают в себя информацию о репозиториях и действиях разработчиков. Поиск зависимостей в этих данных может помочь в понимании тенденций разработки, популярности определенных технологий, поведения разработчиков и многого другого. Данные могут быть использованы с целью анализа как самими сотрудниками портала, так и сторонними пользователями.

Было создано огромное число методов, позволяющих извлекать зависимости из данных. Поиск шаблонов [2] является важным методом по нахождению зависимостей и закономерностей в данных и применяется в множестве сфер от маркетинга до генетики. Шаблон может быть определен, как правило: «если A, то B». Например, для анализа корзины в супермаркете может быть найдено правило «Если молоко, то хлеб», которое указывает, что человек, купив хлеб, покупает и молоко. Применение метода поиска шаблонов в данных GitHub может помочь выявить полезные шаблоны. Например, можно постараться выявить, какого типа репозитории становятся популярными, а какого нет, какие факторы положительно влияют на популярность, а какие, наоборот, чаще всего оказывают отрицательное влияние и делают репозиторий неинтересным для других пользователей.

Создание системы – рабочего инструмента для поиска шаблонов в данных GitHub, с помощью которого можно было бы получать и анализировать самые свежие данные, – могло бы помочь руководителям проектов или другим пользователям лучше понимать текущее состояние сферы разработки, применять важные бизнес-решения в проектах, опираясь на актуальную информацию, а также понять, как поднять популярность репозитория и привлечь разработчиков или спонсоров в свой проект.

Постановка задачи

Целью выпускной квалификационной работы является разработка системы поиска шаблонов в данных портала GitHub. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ и спецификацию предметной области;
2) выполнить проектирование пользовательского интерфейса и архитектуры системы;
3) реализовать систему поиска шаблонов;
4) провести тестирование системы поиска шаблонов;
5) провести эксперименты с применением разработанной системы.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения и списка литературы. Объем работы составляет 43 страницы, объем списка литературы – 22 источника.

В первой главе приводятся формальные определения, а также проводится анализ существующих аналогов и методов, решающих похожие задачи.

Во второй главе описывается предметная область, функциональные и нефункциональные требования к системе.

В третьей главе приведена диаграмма вариантов использования, выполнено проектирование пользовательского интерфейса.

В четвертой главе описана реализация, которая включает в себя архитектуру системы и описание модулей, а также выполнено функциональное тестирование приложения.

В пятой главе проведены вычислительные эксперименты системы, а также эксперименты поиска интересных шаблонов.

В заключении подводится итог проделанной работы.

В приложении представлены листинги классов основных модулей и результаты функционального тестирования.

1. МЕТОДЫ ПОИСКА ШАБЛОНОВ

1.1. Формальные определения

Поиск шаблонов [3] − это метод анализа данных, используемый для выявления сильных связей и закономерностей между элементами в больших наборах данных, который может применяться в различных областях, таких как маркетинг, биоинформатика, медицина и т.д.

Элемент (i) – это отдельный объект, который может присутствовать в транзакции. Элементы могут быть любыми вещами, например, продуктами в корзине покупателя.

Множество всех возможных элементов (I) – универсальное множество, содержащее все возможные элементы i, которые могут присутствовать в транзакции.

Набор элементов $(itemset)$ – любая возможная комбинация элементов i из множества всех возможных элементов I. Набор также может состоять только из одного элемента.

Транзакция (t) – это заданный набор различных элементов i, где $i \subseteq I$. Например, транзакцией может являться покупка, совершенная одним человеком.

Набор данных (D) – это коллекция транзакций t, обозначенная как $D = \{t_1 \dots t_k\}$, в котором k транзакций и где каждая транзакция t состоит из элементов i.

Разреженная матрица – это формат представления набора данных, где каждая транзакция представляет собой множество всех элементов с булевым значением, обозначающим присутствие данного элемента в транзакции. Формат разреженной матрицы необходим для дальнейшего применения алгоритмов поиска шаблонов.

Шаблоны (или ассоциативные правила) определяются как выражения вида «если A, то B» или «A → B». Слева находится антецедент, справа – консеквент.

Антецедент – это условие или предпосылка правила, которое предшествует символу «→» в выражении вида «A → B» или является «A» в выражении «если A, то B» и определяет условие для истинности высказывания.

Консеквент – это результат или вывод импликации, представляющий собой часть высказывания, которая следует после символа «→» в выражении вида «A → B» или является «B» в выражении «если A, то B» и указывает на результат или следствие, которое происходит, если условие (антецедент) истинно.

Априори (apriori) − алгоритм поиска ассоциативных правил. Одной из главных идей алгоритма является то, что он использует свойство антимонотонности множеств, например, если набор элементов X не является частым, то набор элементов Y, который состоит из набора X и нового элемента, заведомо не будет частым и его можно отбросить.

Алгоритм apriori или другие аналоги используют некоторые метрики, которые помогают сделать выводы из набора элементов.

Поддержка ($support$) – метрика, которая позволяет определить, насколько часто шаблон встречается в транзакциях и вычисляется по формуле (1):

$$supp(X \to Y) = \frac{|\{t \in D \mid (X \cup Y) \subseteq t\}|}{|D|}, \quad (1)$$

где X и Y – наборы элементов, t – транзакция, D – датасет.

Достоверность ($confidence$) – метрика, которая помогает определить вероятность того, что набор элементов X следует набору Y, и показывает, насколько часто данный шаблон встречается среди всех шаблонов с таким же антецедентом, достоверность вычисляется по формуле (2):

$$conf(X \to Y) = \frac{supp(X \cup Y)}{supp(X)}. \quad (2)$$

Подъем ($lift$) – метрика, которая показывает, насколько оба набора элементов зависят друг от друга, то есть показывает корреляцию, и вычисляется по формуле (3):

$$lift(X \to Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}. \quad (3)$$

Подъем может являться ключевой метрикой в случае оценки зависимости наборов элементов. Если $lift > 1$, можно говорить о том, что между наборами есть зависимость. а если $lift < 1$, то наборы данных влияют друг на друга негативно.

1.2. Обзор работ по тематике

Анализу данных портала GitHub посвящено много работ. В основном при анализе используются различные алгоритмы машинного обучения, такие, как KNN или нейронные сети. Ниже приведен обзор некоторых работ.

Целью работы [4] является анализ данных GitHub для моделирования развивающихся репозиториев и динамических трендов в программировании. Авторы используют данные архива GH Archive [5] о 9350 популярных репозиториев с 2015 по 2018 год, включая количество часто встречающихся событий, таких, как fork, pull request, push и т.д., язык программирования. Анализ проводится с помощью метода анализа временных рядов на основе нейронной сети LSTM [6].

Проблема анализа данных GitHub и прогнозирования трендов разбивается на три подзадачи: прогнозирование трендов репозиториев, прогнозирование трендов языков программирования и прогнозирование трендов областей применения. В каждой из этих подзадач используются различные данные для построения временных рядов.

Производительность экспериментов измеряется в терминах среднеквадратической ошибки RMSE и коэффициента детерминации R2. Модели показали хорошую производительность с очень низким значением RMSE и высоким значением R2, которое лежит между 0 и 1.

По результатам анализа было замечено, что репозитории в областях веб-майнинга, машинного обучения и информационной безопасности становятся все более популярными. Из языков программирования JavaScript является наиболее трендовым языком.

В работе [7] изучается влияние пандемии COVID-19 на работу удаленных команд. Основные задачи включают изучение общего эффекта пандемии, оценку эффектов пандемии на отдельные команды и анализ свойств команд, которые связаны с устойчивостью под воздействием шока.

Авторы использовали данные о 71928 репозитории для проверочного набора и 87914 репозиториев для целевого набора из долгосрочного журнала событий GH Archive и зеркала официального источника GHTorrent [8]. На основе данных о количестве push событий репозитория вычисляется метрика, отражающая продуктивность команды, а на основе количества участников вычисляется рост команды.

Были обучены модели машинного обучения градиентного бустинга деревьев решений и случайного леса для прогнозирования результатов за определенный месяц в 2019 году, а затем были сделаны контрфактические прогнозы результатов целевых репозиториев за шесть месяцев 2020 года. Эффект пандемии затем измерялся по разнице между наблюдаемыми и контрфактическими результатами.

В исследовании было обнаружено, что продуктивность команды сразу снизилась после шока (1–3 месяцы), а затем восстановилась к норме через несколько месяцев (4–6 месяцы). Количество активных участников команды, с другой стороны, не сразу упало, когда наступила пандемия (1–3месяцы), но оно снижается постепенно, когда команды реагируют в более долгосрочной перспективе (5–6 месяцы).

Из приведенных выше работ, можно заметить, что основным и самым популярным источником данных является GH Archive. В качестве данных авторы используют события репозитория, такие как push, commit и т.д., а также другие данные, например, язык репозитория. Также авторы собирают данные за срок от полу года до нескольких лет. Эти особенности стоит учесть при будущей разработке системы.

2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1. Портал GitHub

GitHub – это облачная платформа для хранения репозиториев, основанная на системе контроля версий Git [9]. Обычно репозиторий GitHub может включать в себя исходный код программы, документацию, графические ресурсы и другие элементы проекта. Репозиторий также отслеживает все изменения, которые вносятся в файлы, и кто их внес, что облегчает совместную работу и управление версиями. Каждый репозиторий имеет свой уникальный URL, который можно использовать для доступа к нему через веб-интерфейс GitHub или через Git на локальной машине.

В таблице 1 представлена семантика основных команд Git.

Таблица 1 – Семантика команд в Git

№	Команды	Семантика
1	`init`	Инициализация нового Git репозитория
2	`add`	Добавление файлов в индекс для последующего коммита
3	`commit`	Фиксация изменений в репозитории
4	`push`	Отправка изменений в удаленный репозиторий
5	`merge`	Слияние изменений из одной ветки в другую
6	`branch`	Создание новой ветки в репозитории
7	`checkout`	Переключение между ветками или коммитами в репозитории
8	`clone`	Клонирование удаленного репозитория на локальную машину

2.2. Описание исходных данных

Когда пользователь выполняет какое-либо действие с репозиторием в GitHub, это действие сохраняется и связывается с конкретным событием. Портал GitHub, а также сторонние сервисы, предоставляют доступ к событиям и другим данным, если они касаются публичного репозитория. Ниже приведены возможные способы получения данных.

GH Archive – это архив, который занимается сбором и хранением данных публичных репозиториев, предоставляющий доступ к этим данным по часам. Получение данных на этом портале не предполагает возможности составления детализированного запроса и позволяет загрузить только данные всех событий, которые произошли за час, что усложняет отбор на более ши-

рокий промежуток времени. Также данные GH Archive доступны в публичном датасете Google BigQuery [10], который позволяет производить SQL-запросы, что освобождает от скачивания лишних данных, однако BigQuery имеет довольно строгие ограничения для запросов, которые не позволяют сканировать и запрашивать большой объем данных бесплатно.

Одним из лучших вариантов запроса данных GitHub с небольшим числом ограничений является сервис ClickHouse Playground [11], с помощью которого можно получить доступ к уже предобработанным данным из GH Archive путем отправки SQL-запросов.

Например, для каждого репозитория можно получить такие атрибуты, как:

– количество отправок изменений репозитория (pushes);

– средний объем отправок изменений репозитория (avg_push_size);

– количество отправленных запросов на слияние в репозитории (pull_requests);

– число отслеживаний репозитория или звезд (watches);

– количество открытых запросов на улучшение или исправление ошибки репозитория (issue).

В SQL-запросе к ClickHouse Playground можно также регулировать параметры данных, например, количество репозиториев, временной промежуток, за который произошли события GitHub, минимальное кол-во звезд, которое набрал репозиторий в этот промежуток, и т.д.

Рассмотрим еще один вариант получения данных портала GitHub. GitHub API [12] – это официальный набор способов и правил, по которым различные программы общаются с GitHub и обмениваются данными. С его помощью также можно выполнять HTTP-запросы с целью получения различных данных в формате JSON о каком-либо репозитории, пользователе или событии, если они носят публичный характер.

Ниже представлены примеры, атрибутов, которые можно получить:

– name – название репозитория;

– description – описание репозитория;

– is_fork – является ли репозиторий ветвлением другого репозитория;

– language – язык, использующийся в репозитории;

– license_name – имя лицензии репозитория.

2.3. Применение алгоритма поиска шаблонов

Чтобы составить список транзакций можно использовать следующие из вышеупомянутых атрибутов, которые будут указывать на события в репозитории:

– количество отправок изменений репозитория, отражающее его активность (pushes);

– количество вопросов к репозиторию, отражающее насколько часто у пользователей возникают вопросы, которые могут быть связаны с проблемами использования кода (issues);

– количество поставленных звезд репозиторию, отражающее его популярность (watches).

Пример списка транзакций представлен в таблице 2.

Таблица 2 – Пример списка транзакций

№	Изменения	Вопросы	Звезды
1	10	0	100
2	10	5	100
3	5	10	25
4	5	10	50
5	10	0	100

Каждому событию или атрибуту соответствует название колонки в таблице. Каждая строка представляет собой данные одного репозитория. Числа в колонке отражают количество событий для каждого репозитория. При поиске шаблонов название события и его кол-во будут объединены в элемент вида «Событие кол-во», например «Изменения 10».

Применив алгоритм apriori при пороговых значениях поддержки и достоверности 0,1 можно получить следующие шаблоны, представленные в таблице 3.

Таблица 3 – Пример списка шаблонов

№	Антецедент	Консеквент	Поддержка	Достоверность
1	Изменения 10, Вопросы 0	Звезды 100	0,4	1
2	Изменения 5, Вопросы 10	Звезды 50	0,2	0,5

Запись «Изменения 10, Вопросы 0 → Звезды 100» первого шаблона означает, что из количества отправок изменений в репозиторий 10 и количества вопросов к репозиторию 0 следует количество звезд репозитория 100. Исходя из набора транзакций, сам шаблон может указывать на то, что при высоком числе изменений репозитория и низком числе вопросов к репозиторию, репозиторий получает высокое число звезд. Значение поддержки 0,4 указывает, что шаблон присутствует в 40% транзакций, а значение достоверности 1, что при данном антецеденте правило работает во всем списке транзакций. Шаблон под вторым номером указывает на обратную ситуацию: при низком числе изменений репозитория и высоком числе вопросов к репозиторию, репозиторий получает низкую оценку.

Шаблоны такого типа определяют, как активность команды в совокупности с качеством кода может влиять на популярность репозитория.

2.4. Особенности работы с данными для поиска шаблонов

Для числовых данных, обычно, чтобы снизить разнообразие элементов транзакций, применяется дискретизация. Дискретизация представляет собой преобразование непрерывных числовых данных в дискретные или категориальные данные. Такое преобразование помогает уменьшить время работы алгоритма поиска шаблонов и упрощает анализ последующих шаблонов.

В поиске шаблонов дискретизация происходит следующим образом. Данные для дискретизации сортируются в порядке убывания, разбиваются

на квартили (4 промежутка) или децили (10 промежутков), каждое измерение заменяется на порядковое значение квартиля или дециля, в которое оно попало. В результате мы получаем фиксированное число элементов для большого числа разных измерений. В текущей задаче будет необходима дискретизация для кол-ва push, commit, issues и других числовых данных репозитория.

Однако стоит учесть, что дискретизация может привести к потере информации, поскольку некоторые нюансы непрерывных данных могут быть утеряны в процессе преобразования их в дискретные. Поэтому важно предоставить пользователю системы выбирать метод дискретизации отдельно для каждого атрибута данных.

В поиске шаблонов обычно регулируют состав транзакций путем добавления или исключения каких-либо элементов в транзакциях. Состав транзакций может зависеть от шаблонов, которые хочет найти аналитик.

Для данных GitHub, например, для исследования влияния активности на популярность репозитория из сервисов можно получать только атрибуты числа изменений (push), запросов на слияния (pull request) и поставленных звезд (watches).

Для хранения данных, обычно, используется база данных. В текущем случае база данных позволит получить быстрый доступ к данным из сервисов, которые были сохранены ранее. Также база данных позволит аналитику хранить сразу множество наборов данных с разными параметрами для будущего поиска.

2.5. Требования к системе

В ходе анализа предметной области были определены функциональные и нефункциональные требования к разрабатываемой системе.

Функциональные требования

Функциональные требования устанавливают требования о том, как

должна себя вести система и прямым образом относятся к функционалу системы.

Разрабатываемая система должна удовлетворять следующим функциональным требованиям.

1. Система должна предоставлять возможность запроса и хранения выборок данных GitHub.
2. Система должна предоставлять возможность выбора параметров отбора запрашиваемых данных.
3. Система должна предоставлять возможность поиска шаблонов в выборках данных GitHub.
4. Система должна предоставлять возможность выбора параметров поиска шаблонов.
5. Система должна предоставлять возможность просмотра и фильтрации шаблонов.
6. Система должна предоставлять возможность сохранения шаблонов.
7. Система должна предоставлять удобный пользовательский интерфейс.

Нефункциональные требования

Нефункциональные требования устанавливают требования ограничений для системы, которые не влияют прямым образом на функционал системы.

Система должна удовлетворять следующим нефункциональным требованиям.

1. Система должна представлять собой веб-приложение.
2. Система должна быть написана на языке Python.

3. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

3.1. Диаграмма вариантов использования

Для описания способов взаимодействия с системой была разработана диаграмма вариантов использования на языке UML, представленная на рисунке 1.

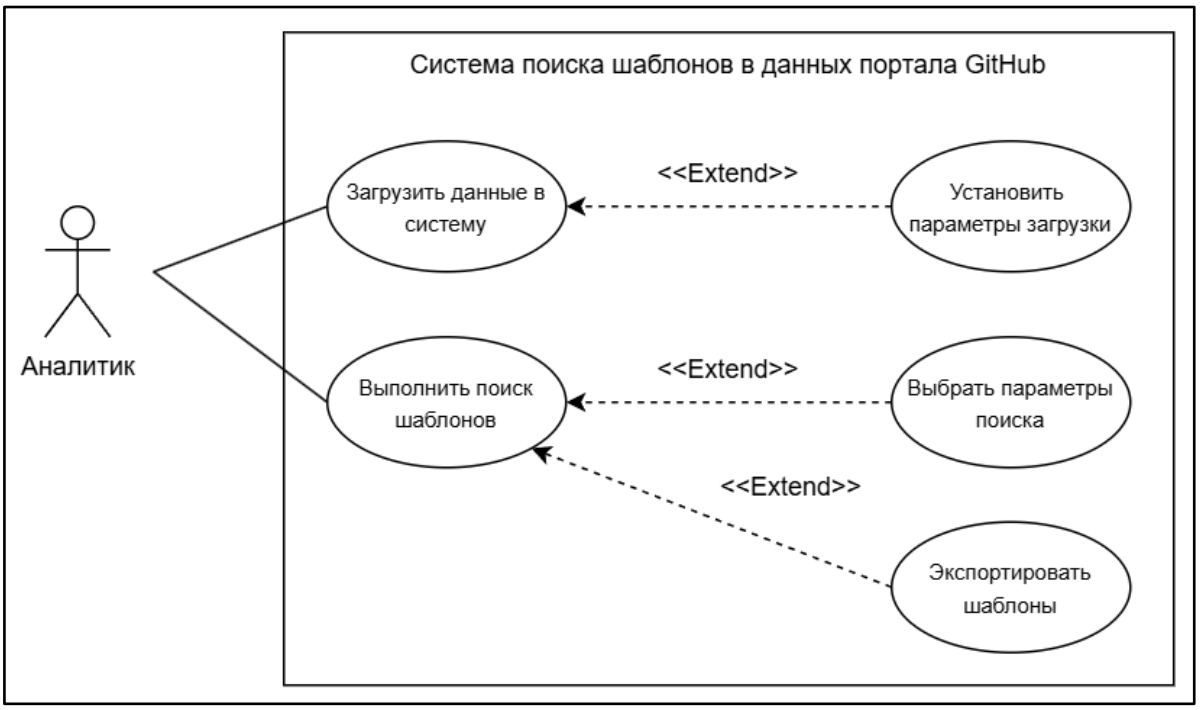

Рисунок 1 – Диаграмма вариантов использования

Единственным актером системы является аналитик, который может использовать все функции приложения. Ниже приведены эти функции.

1. Установить параметры данных. Аналитик может выбрать параметры запрашиваемых из сторонних сервисов данных GitHub.

2. Загрузить данные в систему. Аналитик может выполнить запрос к сервисам, которые хранят данные и сохранить результаты запроса в систему.

3. Выбрать параметры поиска. Аналитик выбирает сохраненные в систему данные, порог поддержки и достоверности, а также другие параметры поиска шаблонов.

4. Выполнить поиск шаблонов. Аналитик выполнят поиск шаблонов с заданными параметрами.

5. Экспортировать шаблоны. Аналитик может экспортировать найденные шаблоны в формате таблицы Excel.

3.2. Разработка пользовательского интерфейса

Интерфейс приложения будет представлять собой две страницы: страница получения данных GitHub и страница поиска шаблонов. Макет страницы получения данных GitHub представлена на рисунке 2.

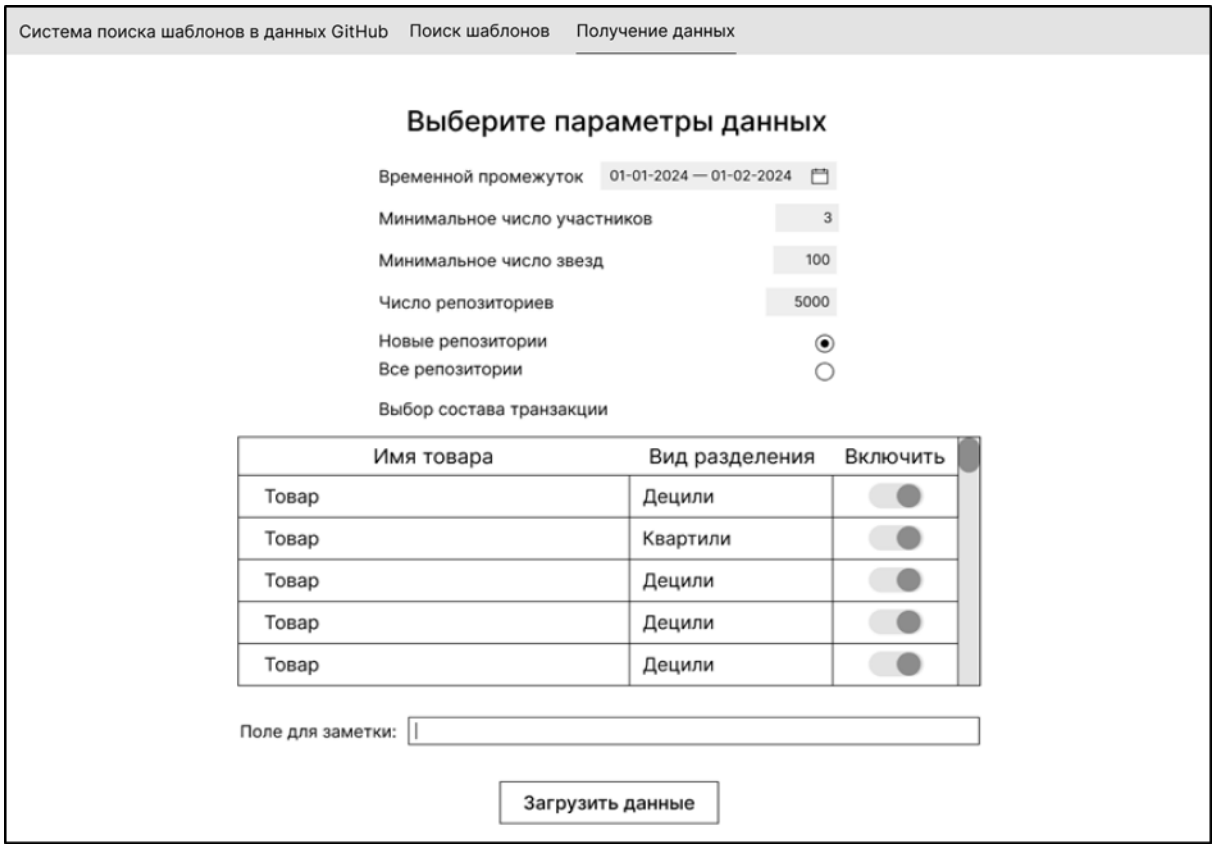

Рисунок 2 – Макет страницы получения данных

На странице получения данных GitHub расположен интерфейс для выбора параметров получаемых данных. Аналитик выбирает параметры отбора репозиториев, данные которых будут загружены в систему. Параметры включают в себя временной промежуток, за который отбираются репозитории, минимальное число участников (в число входит владелец, а также участники, присоединившиеся за выбранный временной промежуток к данному репозиторию), минимальное число звезд (репозитории получившие заданное число звезд и более за выбранный временной промежуток), новые

репозитории (созданные за временной промежуток) или все репозитории. Также аналитик может выбрать атрибуты загружаемых данных, выбрать децили или квартили для дискретизации, указать заметку о сохраняемых данных. После приведенных выше действий аналитик нажимает кнопку «Загрузить данные», после чего данные загружаются и сохраняются в систему.

На рисунке 3 изображен макет страницы поиска шаблонов.

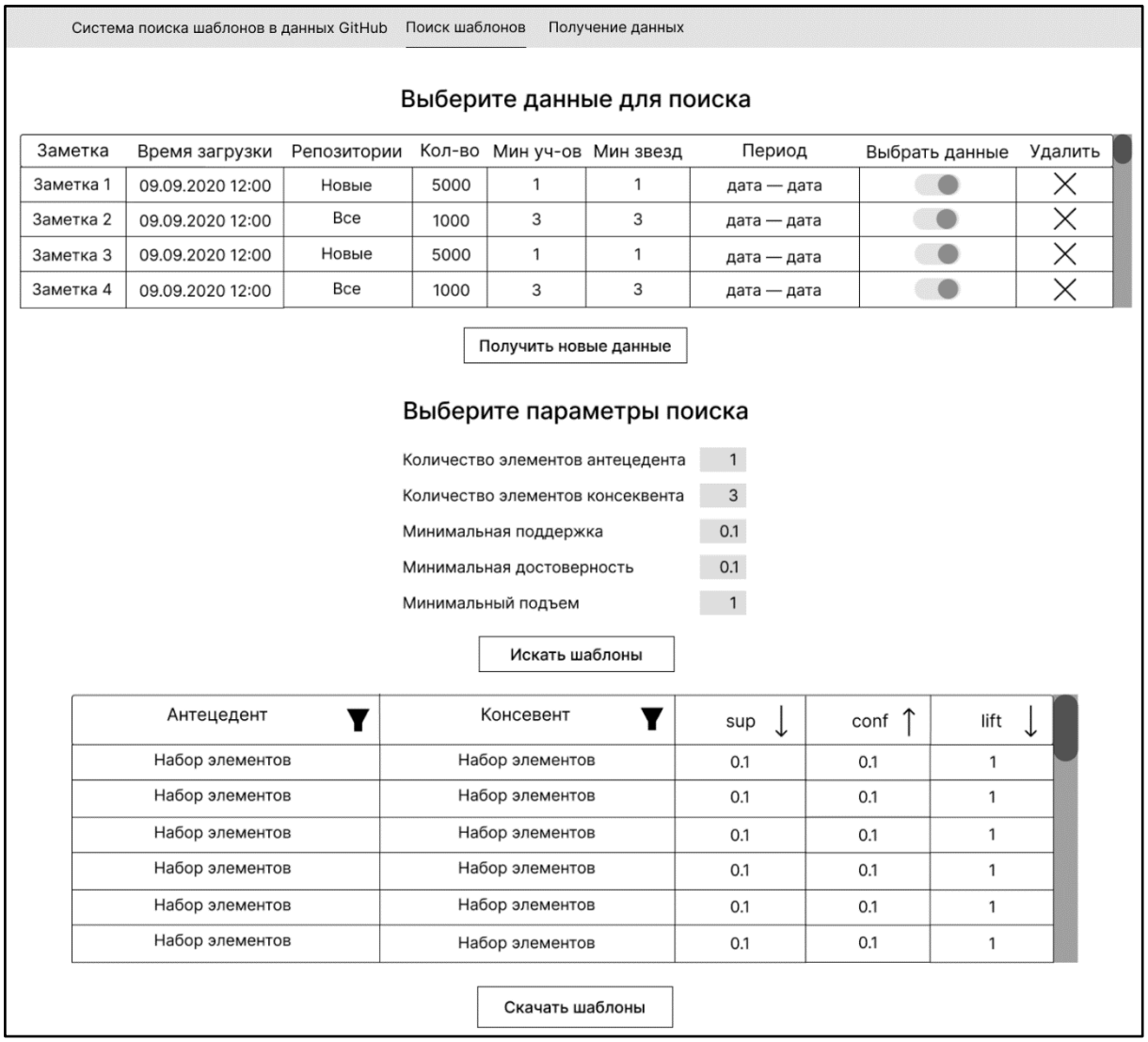

Рисунок 3 – Макет страницы поиска шаблонов

В самом начале в окне выбора данных аналитик должен выбрать ранее полученные и сохраненные наборы данных из GitHub. В окне отображаются параметры информация о каждом наборе, включающая заметку, время загрузки и другие параметры, задаваемые на странице получения данных.

Если наборы будут содержать одинаковые атрибуты и параметры дискретизации, то их можно будет объединить. Если аналитику нужно будет получить новый набор данных, он может нажать на кнопку «Получить новые данные» и перейти на страницу получения данных.

После выбора данных аналитик выбирает параметры поиска шаблонов. Параметры включают в себя количество элементов в антецеденте и консеквенте, чтобы регулировать вид получаемых в итоге шаблонов, а также пороги поддержки, достоверности и подъема. После выбора параметров поиска шаблонов аналитик нажимает кнопку «Искать шаблоны», выполняется поиск и отображается таблица шаблонов.

Таблица шаблонов содержит колонки с антецедентом, консеквентом, а также метриками поддержки, достоверности и подъема. В колонках «Антецедент» и «Консеквент» расположен значок фильтра по элементам, который позволит искать нужный тип шаблонов по ключевым словам. В колонках поддержки, достоверности и подъема расположен значок сортировки, при нажатии на который будет применяться сортировка по возрастанию или убыванию. Также аналитик может нажать кнопку «Скачать шаблоны», после чего в папку загрузки выполнится скачивание шаблонов в формате xlsx. Будет выполнена загрузка только тех шаблонов, которые отображаются непосредственно в таблице. Аналитик может отфильтровать шаблоны и загрузить только те, которые ему нужны.

4. РЕАЛИЗАЦИЯ СИСТЕМЫ

4.1. Архитектура системы

Для разработки системы был выбран язык Python версии 3.11.7. Разработка велась в редакторе кода Visual Studio Code [13]. Веб-приложение реализовано с помощью фреймворка Django [14]. Также для пользовательского интерфейса использовались HTML, CSS и JavaScript.

В качестве основной структуры для табличных данных в приложении был выбран Dataframe из библиотеки Pandas [15]. Данная библиотека в совокупности с библиотекой Numpy [16] предоставляет широкий спектр всевозможных операций над табличными данными.

Система представляет собой веб-приложение, состоящее из нескольких модулей: модуль интерфейса, модуль запроса данных из сервисов, модуль предобработки данных, модуль поиска шаблонов и база данных.

На рисунке 4 изображена диаграмма компонентов системы.

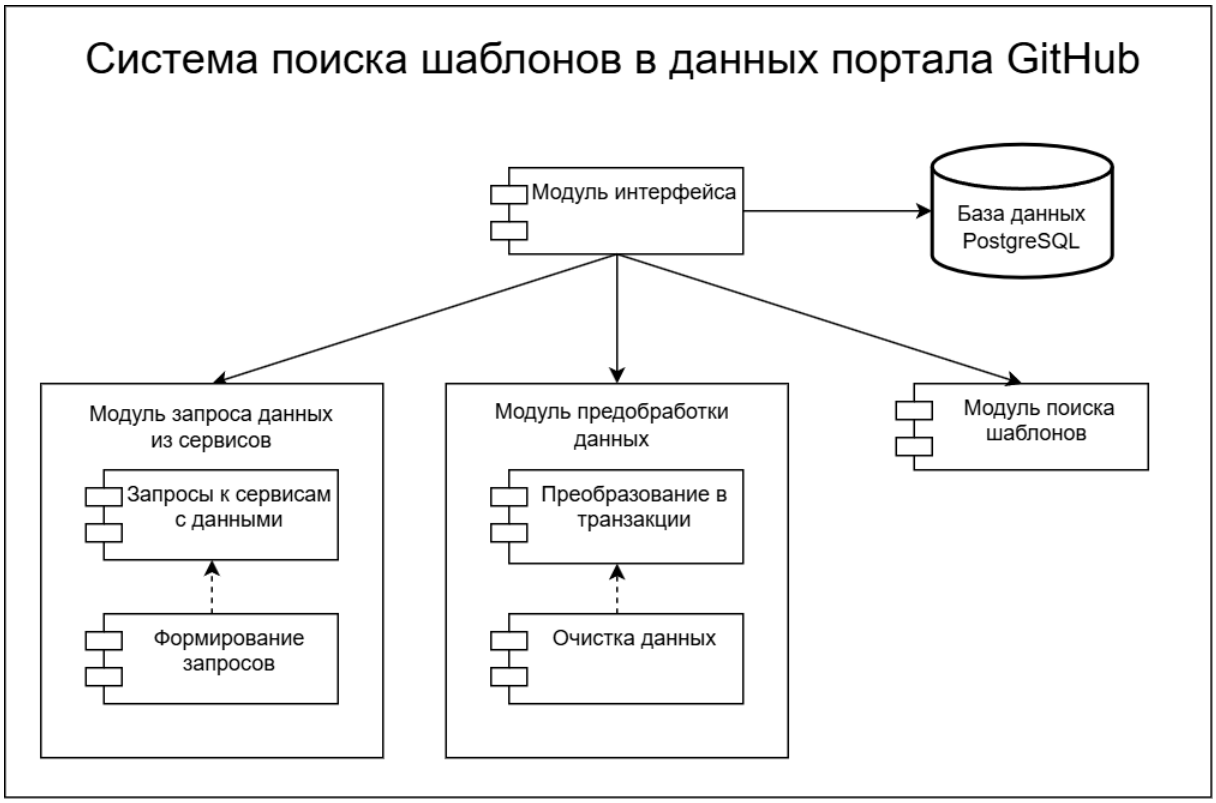

Рисунок 4 – Диаграмма компонентов системы

Ниже приведено подробнее описание реализации каждого модуля. Исходный код доступен в публичном репозитории GitHub [17]. Интерфейсы основных модулей, которые представляют из себя классы, приведены в листингах 1–3 в приложении А.

Реализация модуля запроса данных из сервисов

Модуль запроса данных из сервисов предоставляет функции для формирования запросов и отправки запросов в Clickhouse Playground и Github API. Модуль принимает на вход параметры загрузки данных, которые включают в себя запрашиваемые поля, промежуток времени, лимит числа репозиториев, минимальные число звезд репозиториев, минимальное число присоединившихся участников к репозиториям и булево значение, которое указывает на отбор только созданных за промежуток времени репозиториев. По указанным параметрам формируются и производятся запросы. Первый SQL-запрос производится в Clickhouse Playground, запрос возвращает данные для каждого репозитория. По имени каждого репозитория производятся асинхронные HTTP-запросы в Github API, чтобы получить данные, которых нет в Clickhouse, результат объединяется с предыдущим запросом. Для реализации запросов в Clickhouse Playground используется библиотека Selenim [18], а для асинхронных запросов в Github API используются библиотеки Asyncio [19] и Aiohttp [20]. Структура данных, которые возвращаются на выходе из модуля, представлена в таблице 4.

Таблица 4 – Структура выходных данных модуля запроса данных

Атрибут	**Сервис**	**Тип данных**	**Семантика**
pushes	Clickhouse Playground	Целое Число	Количество отправленных изменений (pushes) в репозиторий
avg_push_size	Clickhouse Playground	Вещественное число	Средний размер отправленных изменений (pushes)
pull_requests	Clickhouse Playground	Целое Число	Количество запросов на внесение изменений (pull requests)
merged_pull_requests_ratio	Clickhouse Playground	Вещественное число	Отношение успешно принятых запросов на внесение изменений (pull requests) к общему числу таких запросов

Окончание таблицы 4

Атрибут	Сервис	Тип данных	Семантика
issues	Clickhouse Playground	Целое Число	Количество созданных вопросов или проблем (issues)
new_members_count	Clickhouse Playground	Целое Число	Количество новых участников репозитория
closed_issues_ratio	Clickhouse Playground	Вещественное число	Отношение успешно закрытых вопросов или проблем (issues) к общему числу таких вопросов
watches	Clickhouse Playground	Целое Число	Количество пользователей, отслеживающих обновления репозитория (поставивших звезду)
forks	Clickhouse Playground	Целое Число	Количество ветвлений (forks) репозитория
language	GitHub API	Строка	Язык программирования, на котором написан код в репозитории
license_name	GitHub API	Строка	Название лицензии, под которой распространяется код в репозитории
is_deleted_or_private	GitHub API	Булево	Индикатор, указывающий, удален репозиторий или является ли он приватным

Реализация модуля преобразования данных

Модуль преобразования данных принимает на вход вид таблицы, представленной выше, а также словарь, который содержит метки разделения на квартили или децили для каждого элемента транзакции, который входит в ее состав. Сначала модуль очищает данные, удаляя возможные дубликаты и пустые значения, затем для каждого числового атрибута в исходной таблице модуль извлекает значения децилей или квартилей (по выбору пользователя) и сравнивает с каждым значением этого атрибута. Это значение меняется на порядковый номер квартиля или дециля (в обратном порядке), в котором находится значение. После вышеописанных операций транзакции преобразуются в разреженную матрицу для дальнейшего поиска шаблонов и подаются на выход из модуля.

Реализация модуля поиска шаблонов

Модуль поиска шаблонов принимает на вход разреженную матрицу

транзакций, число элементов в антецеденте и консеквенте, а также минимальные значения поддержки, достоверности и подъема. Модуль выполняет поиск ассоциативных правил с заданными метриками. В самом начале происходит поиск частых наборов, в котором используется алгоритм apriori с заданным порогом поддержки. Далее в частых наборах происходит поиск шаблонов с порогом подъема, затем фильтрация с порогом достоверности, а также по числу элементов в антецеденте и консеквенте. На выходе из модуля возвращается таблица, имеющая столбцы консеквента, антецедента, значений метрик поддержки, достоверности и подъема. Для поиска частых наборов и ассоциативных правил была использована библиотека Mlxtend [21].

Реализация базы данных

Для возможности загрузки, хранения и изъятия данных GitHub в системе реализованы следующие объекты (классы).

RepositoryData – представляет собой объект данных репозитория и содержит такие поля, как имя репозитория, идентификатор выборки, а также атрибуты репозитория, такие как «pushes», «pull_requests» и т.д., которые описаны ранее в реализации модуля запроса данных. В одной выборке может быть много объектов RepositoryData.

SampleParams – хранит параметры каждой выборки загруженных данных, включая время сохранения выборки, даты начала и конца временного промежутка, из которого получены репозитории, количество репозиториев, минимальное количество участников и звезд репозитория, а также булево значение, указывающее на то, что взятые за временной промежуток репозитории созданы в этот промежуток. Кроме того, для каждого числового атрибута репозитория из класса RepositoryData хранится информация о его делении (децили или квартили), и для каждого атрибута хранится информация о его статусе (включен в транзакцию или выключен).

Для каждого класса с помощью Django создается таблица в базе данных, которая представлена на рисунке 5. В качестве базы данных была использована реляционная база данных PostgeSQL [22].

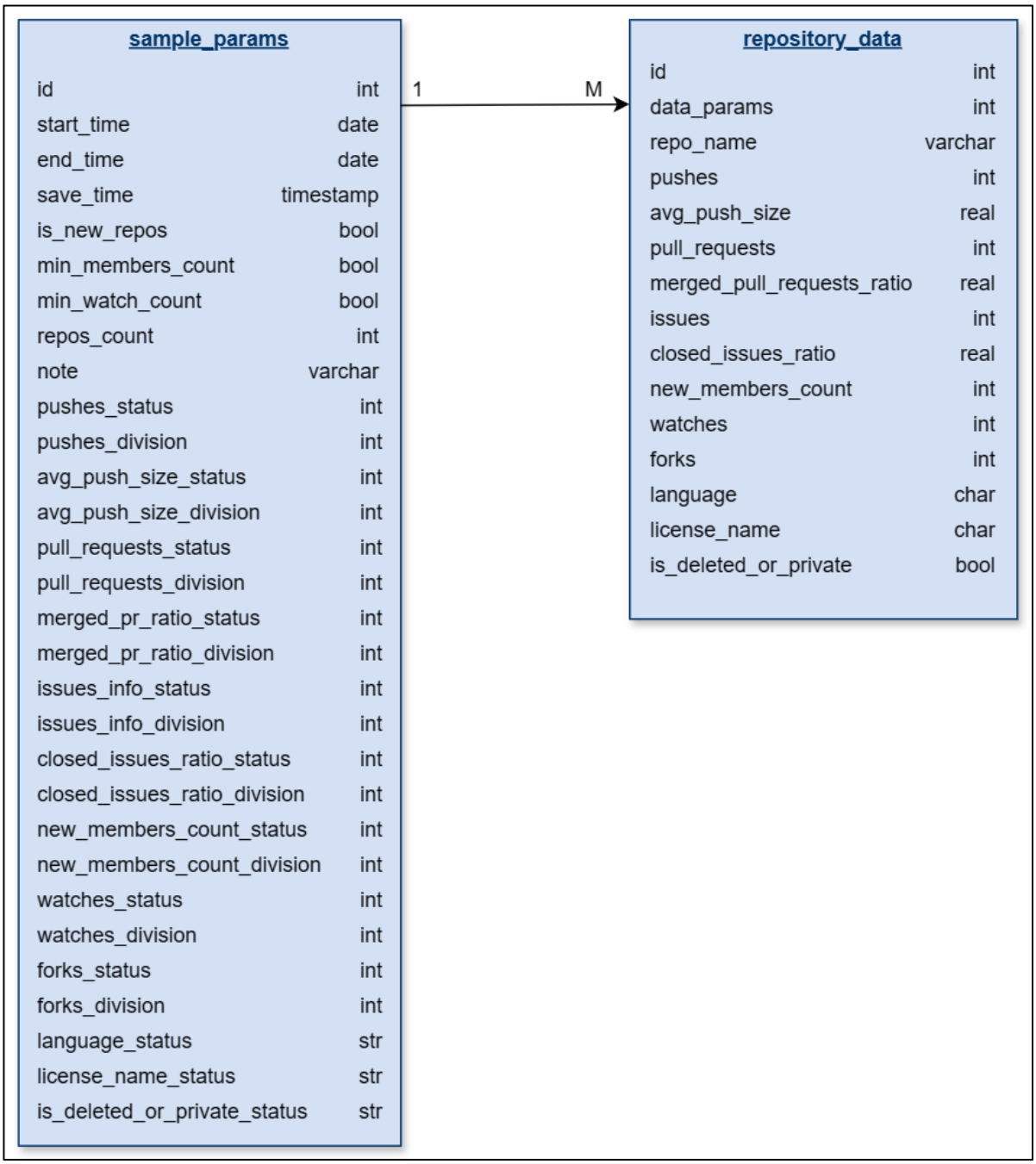

Рисунок 5 – Схема базы данных

4.2. Реализация интерфейса

В рамках работы реализовано две страницы интерфейса: страница получения данных GitHub и страница поиска шаблонов. Страница получения данных GitHub представлена на рисунке 6.

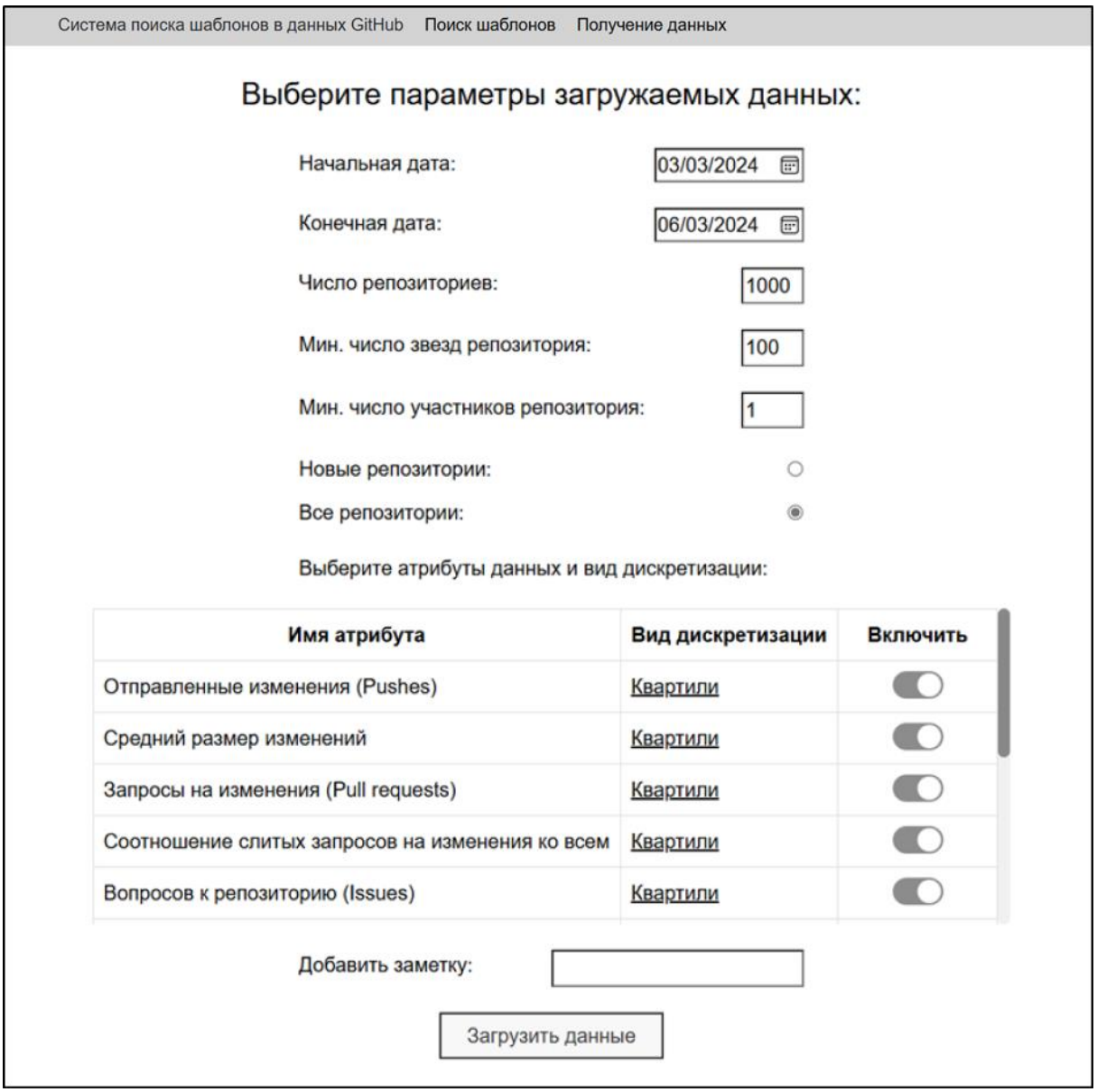

Рисунок 6 – Страница получения данных

На странице получения данных реализован весь функционал по изъятию и сохранению данных в качестве выборок в систему. Аналитик выбирает параметры отбираемых данных, может написать себе заметку и загрузить данные в программу, после чего выведется уведомление об успешной операции загрузки, либо ошибка, если параметры заданы некорректно. Подробное описание страницы и ее элементов приведено в разделе 3.2.

На рисунке 7 изображена страница поиска шаблонов.

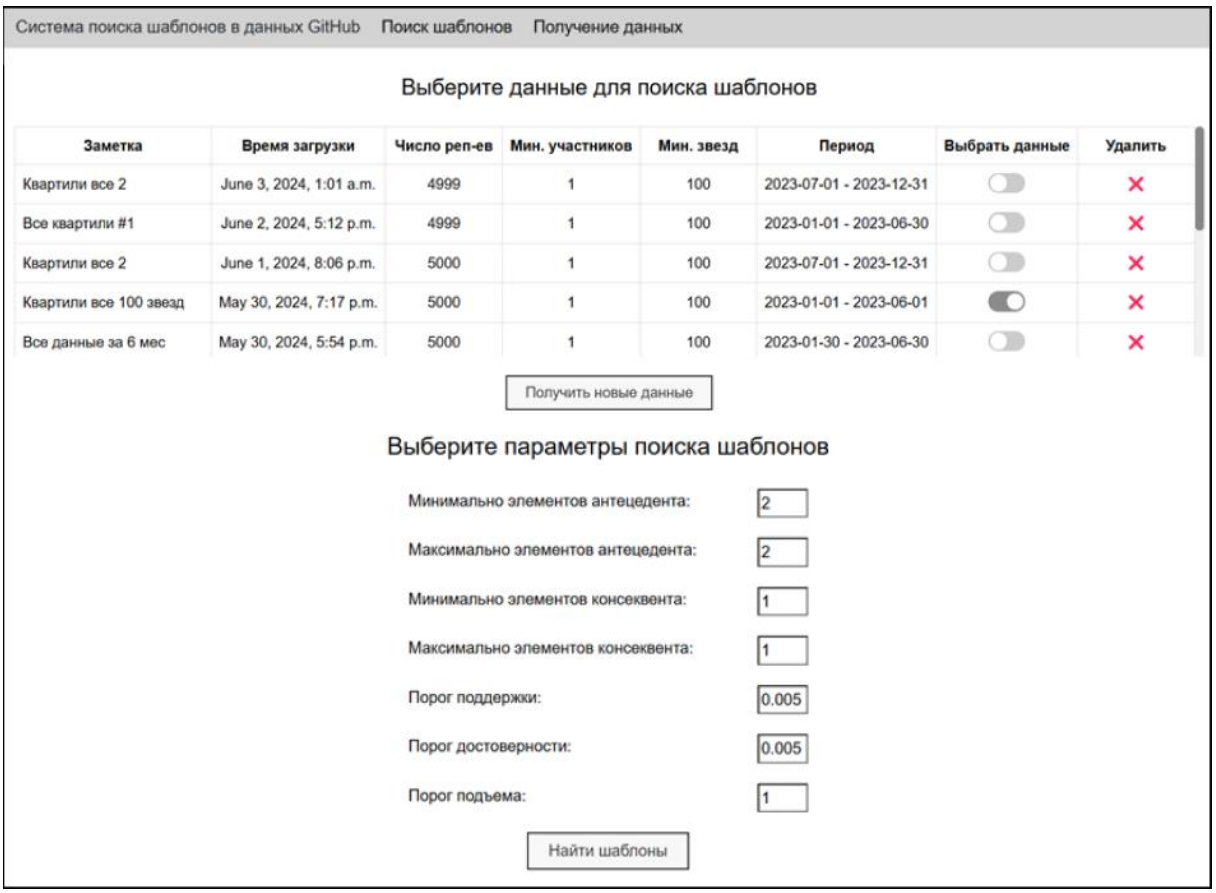

Рисунок 7 – Страница поиска шаблонов

На странице реализован весь функционал для поиска шаблонов. Аналитик выбирает нужные ему наборы данных, параметры поиска и нажимает на кнопку «Найти шаблоны». В случае успешного поиска ниже отобразится таблица, которая содержит найденные шаблоны. В случае, если были заданы некорректные параметры, не выбраны данные для поиска и не было найдено ни одного шаблона, отобразится ошибка. Подробное описание страницы и ее элементов приведено в разделе 3.2. На рисунке 8 изображена таблица, содержащая шаблоны, которая отобразится снизу после успешного поиска шаблонов.

Таблица содержит поля фильтрации шаблонов по подстроке для антецедента и консеквента. Чтобы найти шаблоны, исключающие введенную подстроку, в начале подстроки необходимо установить символ «~». Чтобы выполнить фильтрацию, использовав несколько подстрок, их необходимо

записать в поле фильтра через символы «&». Также была сделана возможность сортировки шаблонов по убыванию или возрастанию поддержки, достоверности и подъема, при нажатии на стрелки.

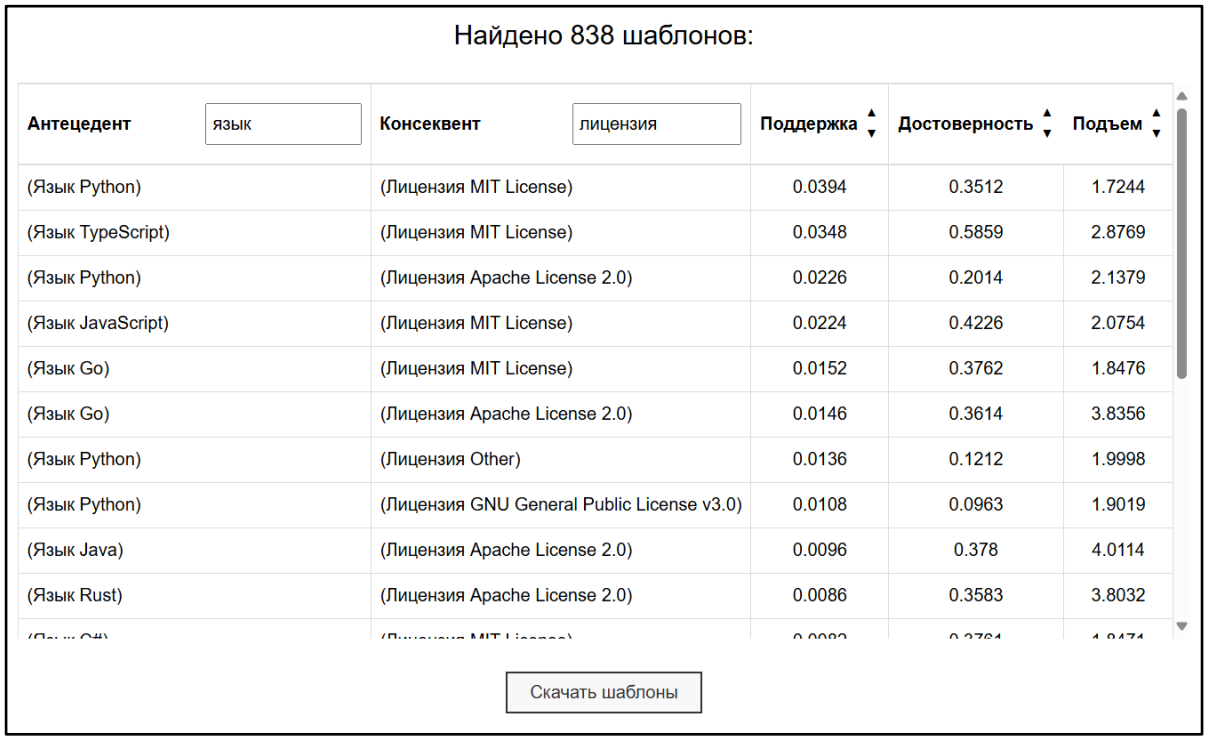

Рисунок 8 – Таблица шаблонов

4.3. Функциональное тестирование

В рамках работы было выполнено функциональное тестирование приложения, которое проверяет работоспособность системы, включая ввод-вывод данных и взаимодействие с другими компонентами. Данный вид тестирования позволяет убедиться, что система выполняет требуемые функции и работает правильно.

Результаты функционального тестирования представлены в таблице 1 приложения Б.

5. ЭКСПЕРИМЕНТЫ

Для демонстрации работы системы и проверки ее эффективности в данном разделе выполнены эксперименты. Характеристики системы, на которой производились эксперименты, указаны в таблице 5.

Таблица 5 – Характеристики системы

Компонент	Спецификация
Операционная система	Windows 11
Версия языка Python	3.11.7
Процессор	Intel Core i7-12700f
Оперативная память	32 Гб

5.1. Вычислительные эксперименты

Первый эксперимент выполнен с целью сравнения быстродействия поиска шаблонов при изменяемом пороге поддержки. Данные представляют собой 5000 репозиториев за период с 01-01-2023 по 30-06-2023 с минимальным числом звезд 100. Порог поддержки изменяется в промежутке от 0,01 до 0,3 с шагом 0,01, достоверность и подъем равны 0,01 и 1 соответственно. Результат в виде графика представлен на рисунке 9.

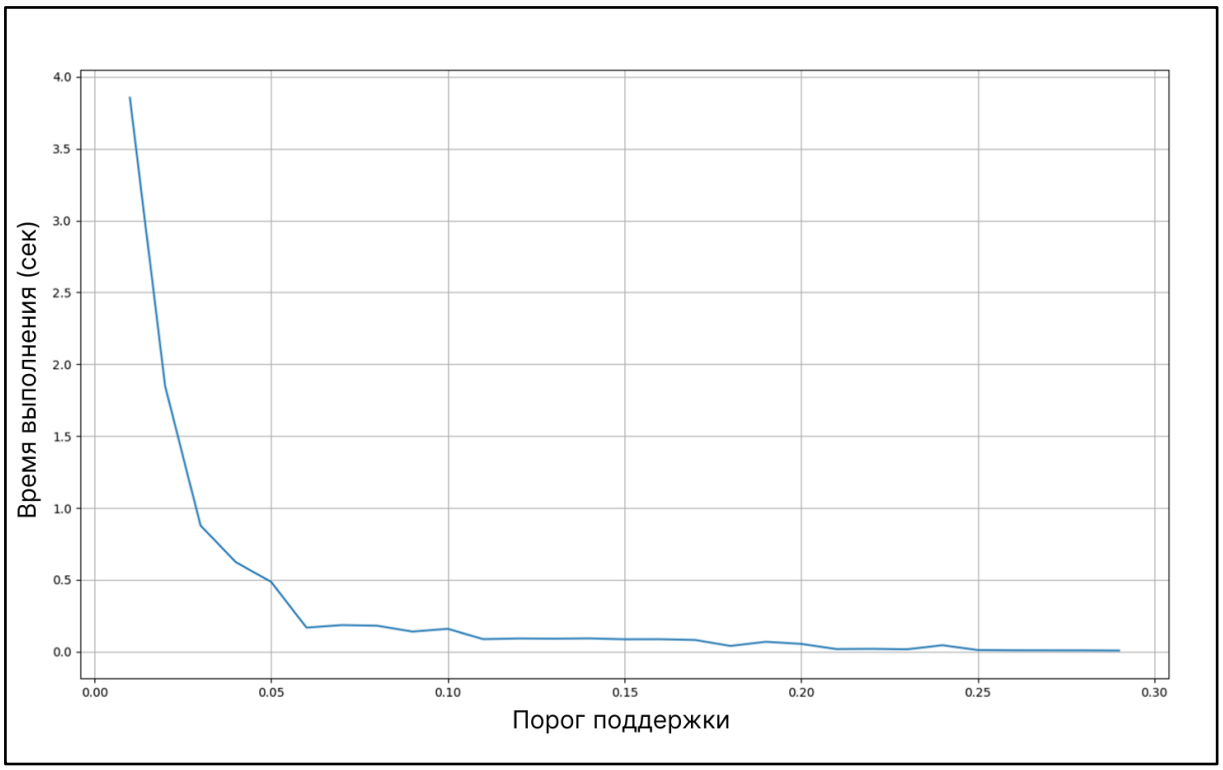

Рисунок 9 – График зависимости времени поиска шаблонов от порога поддержки

График имеет характерную форму «локтя», где время поиска шаблонов резко снижается при достижении порога поддержки около 0,025. После этого значения время поиска шаблонов продолжает уменьшаться, но уже более плавно. Резкий спад времени поиска объясняется тем, что при низком пороге поддержки алгоритм генерирует большое количество кандидатов (потенциальных шаблонов), проверка которых требует значительных временных затрат. По мере увеличения порога поддержки количество кандидатов уменьшается, что ведет к снижению времени их обработки. Когда порог поддержки достигает значения около 0,25, количество шаблонов становится настолько небольшим, что время поиска приближается к нулю. Максимальное время поиска при этом не превышало 4 секунд. Стоит отметить, что в качестве параметров дискретизации в эксперименте были выбраны децили для всех атрибутов.

Второй эксперимент был проведен для сравнения быстродействия поиска шаблонов при изменении объема набора транзакций. Для этого использовался набор данных, состоящий из около 7000 репозиториев за период с 19-01-2023 по 30-06-2023 с минимальным числом звезд 100. На каждом этапе объем набора транзакций увеличивался с шагом в 5% до 100% от исходного набора данных, и фиксировалось время работы алгоритма. Параметры поддержки, достоверности и подъема были заданы статичными и равнялись 0,01, 0,01 и 1 соответственно. В качестве параметров дискретизации были выбраны децили для всех атрибутов. Стоит отметить, что за время текущего эксперимента другие параметры данных, такие как состав транзакции, не изменялись. Результат второго эксперимента представлен в виде графика на рисунке 10.

График имеет скачкообразный вид. Это может объяснятся тем, что размер базы транзакций не оказывает полного влияния на время работы алгоритма и есть другие факторы, которые добавляют хаотичность. Однако, по мере увеличения размера базы транзакций по минимумам и максимумам можно отследить, что время поиска плавно увеличивается при росте числа

транзакций. В среднем время работы занимает около 3 секунд, наибольшее время, которое алгоритм работал – чуть больше 3,5 секунд.

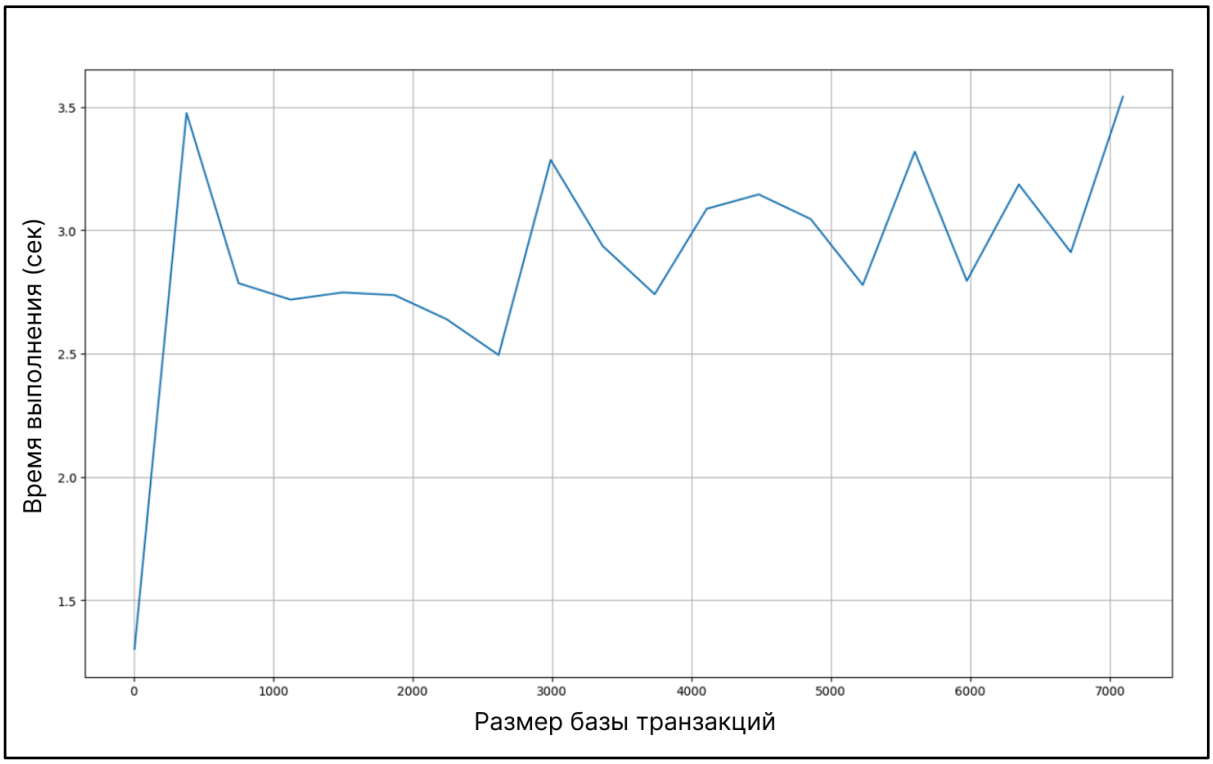

Рисунок 10 – График зависимости времени поиска шаблонов от размера базы транзакций

5.2. Поиск интересных шаблонов

С целью демонстрации полезности системы был выполнен поиск шаблонов, показывающих зависимость между лицензией, языком программирования и популярностью, которая измеряется числом звезд репозитория.

Эксперимент проводился на наборе данных из 10000 репозиториев за первое полугодие 2023 года, которые имеют минимальное число звезд 100. Параметры поддержки, достоверности и подъема равны 0,002, 0,1 и 1 соответственно, в качестве параметров дискретизации для числовых атрибутов были выбраны квартили.

Шаблоны вида «Язык программирования, Лицензия → Звезды {Номер квартиля}», где номер квартиля указывает на позицию репозитория по числу звезд (от самого популярного до менее популярного), представлены в

таблице 6. В первый квартиль попадает число звезд большее, чем 350, в четвертый – меньшее, чем 130. Шаблоны отсортированы по убыванию поддержки, показывающей частоту встречаемости.

Таблица 6 – Шаблоны «Язык программирования, Лицензия → Звезды»

№	Антецедент	Консеквент	Поддержка	Достоверность	Подъем
1	Язык Python, Лицензия MIT	Звезды 1	0,0132	0,3350	1,3377
2	Язык TypeScript, Лицензия MIT	Звезды 1	0,0116	0,3333	1,3309
3	Язык Python, Лицензия Apache 2.0	Звезды 1	0,0082	0,3628	1,4487
4	Язык Go, Лицензия Apache 2.0	Звезды 1	0,0070	0,4795	1,9144
5	Язык JavaScript, Лицензия MIT	Звезды 4	0,0066	0,2946	1,1812
6	Язык JavaScript, Лицензия MIT	Звезды 1	0,0058	0,2589	1,0339
7	Язык Python, Лицензия Apache 2.0	Звезды 4	0,0058	0,2566	1,0288
8	Язык Go, Лицензия MIT	Звезды 1	0,0056	0,3684	1,4710
9	Язык C++, Лицензия MIT	Звезды 4	0,0030	0,3947	1,5824
10	Язык Java, Лицензия Apache 2.0	Звезды 4	0,0030	0,3125	1,2528
11	Язык C#, Лицензия MIT	Звезды 3	0,0026	0,3171	1,2701
12	Язык C, Лицензия MIT	Звезды 4	0,0024	0,4444	1,7746

Из текущих шаблонов можно сделать следующие выводы о влиянии лицензии и языка на популярность репозиториев.

Лицензия MIT часто встречается в популярных репозиториях с языками программирования Python и TypeScript, например, в шаблонах №1 и №2, имея положительную корреляцию с высокой популярностью, которая измеряется подъемом. Однако, в шаблонах №5 и №6 из сочетания языка программирования JavaScript и лицензии MIT следует как и высокая, так и низкая популярность, в обоих случаях имеется более низкая корреляция с популярностью относительно других шаблонов. В сочетании лицензии MIT с языками программирования C++, C# и C в шаблонах №9, №11 и №12 репозитории получают меньшую популярность и имеют высокую корреляцию с меньшей популярностью.

Лицензия Apache 2.0 часто встречается в популярных репозиториях с языками программирования Go и Python (шаблоны №3 и №4) и имеет высокую корреляцию с высокой популярностью. Также имеется шаблон №7, где

из сочетания Python и Apache 2.0 следует меньшая популярность, однако шаблон имеет меньшую частоту встречаемости и меньшую корреляцию с низкой популярностью. Сочетание лицензии Apache 2.0 с языком Java в шаблоне №10 дает меньшую популярность, относительно других подобных сочетаний.

Исходя из этого, наибольший успех в популярности наблюдается у проектов на TypeScript с лицензией MIT и у проектов Go с лицензией Apache 2.0, а также у Python как и с первой, так и со второй лицензией. Эти комбинации также имеют высокую частоту встречаемости, исходя из поддержки, а также положительную корреляцию с популярностью, измеряемой звездами.

Худшими сочетаниями с точки зрения влияния на популярность являются языки C++, C# и C с лицензией MIT, а также Java с лицензией Apache 2.0. Данные сочетания имеют более низкую частоту встречаемости, высокую корреляцию с низкой популярностью.

Выводы экспериментов

Первый эксперимент, направленный на оценку влияния порога поддержки на производительность, показал, что алгоритм способен довольно быстро работать при повышении порога поддержки, демонстрируя значительное уменьшение времени поиска. При уменьшении порога время резко возрастает, однако в текущей задаче время поиска остается приемлемым даже при низких значениях порога поддержки. Это свидетельствует о том, что алгоритм способен эффективно обрабатывать большие объемы данных даже при небольшом пороге поддержки.

Во втором эксперименте, направленном на оценку влияния объема транзакций на производительность, алгоритм продемонстрировал способность к масштабированию, поскольку с ростом объема данных время поиска увеличивалось незначительно. Это указывает на высокую эффективность алгоритма в условиях увеличивающегося объема данных и его пригодность для работы с большими наборами транзакций.

Результаты вычислительных экспериментов подтверждают, что система и использованный алгоритм Apriori является эффективным и быстрым инструментом для поиска шаблонов в запрашиваемых данных объемом несколько тысяч строк и обладает способностью к эффективному масштабированию и адаптации к различным условиям поиска. Это делает его подходящим для использования в текущей системе, где требуется быстрый поиск ассоциативных правил.

В эксперименте поиска интересных шаблонов были найдены шаблоны, показывающие влияние языка программирования и лицензии репозитория на его популярность. Были выявлены сочетания языка и лицензии, из которых следует высокая или менее высокая популярность репозитория.

Результаты эксперимента поиска интересных шаблонов подтверждают возможность применения текущей системы для нахождения интересных закономерностей в данных репозиториев.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана система, осуществляющая извлечение и хранение данных репозиториев платформы GitHub, а также позволяющая осуществлять поиск шаблонов в сохраненных данных.

Были выполнены следующие задачи.

1. Выполнен анализ и спецификация предметной области.
2. Выполнено проектирование пользовательского интерфейса и архитектуры системы.
3. Выполнена реализация системы поиска шаблонов.
4. Проведено тестирование системы поиска шаблонов.
5. Проведены эксперименты с применением разработанной системы, включающие вычислительные эксперименты и эксперимент поиска интересных шаблонов.

В дальнейшем система может использоваться для различных задач как руководителями проектов, так и другими пользователями в качестве системы поиска сложных зависимостей в данных платформы GitHub.

ЛИТЕРАТУРА

1. GitHub. [Электронный ресурс] URL: https://github.com/ (дата обращения: 02.02.2024 г.).

2. Agrawal R., Imielinski T., Swami A. Mining Association Rules between Sets of Items in Large Databases. // ACM SIGMOD Record, 1993. – V. 22. – 207–216 pp.

3. Agrawal R., Srikant R. Fast Discovery of Association Rules. // In Proc. of the 20th International Conference on VLDB, 1994. – 487–499 pp.

4. Varuna T., Anuraj M. Trend Prediction of GitHub using Time Series Analysis. // 10th ICCCNT, 2019. – 1–7 pp.

5. Github Archive. [Электронный ресурс] URL: https://www.gharchive.org/ (дата обращения: 02.02.2024 г.).

6. Schmidhuber J., Hochreiter S. Long Short-Term Memory. // Neural Computation, 1997. –V. 9. – 1735–1780 pp.

7. Xuan L., Wei A., Yixin W., Qiaozhu M. Team Resilience under Shock: An Empirical Analysis of GitHub Repositoriesduring Early COVID-19 Pandemic. // ICWSM, 2023. –V. 17. – 578–589 pp.

8. GHTorrent. [Электронный ресурс] URL: https://github.com/ghtorrent/ghtorrent.org/tree/master (дата обращения: 02.02.2024 г.).

9. Git documentation. [Электронный ресурс] URL: https://git-scm.com/doc (дата обращения: 02.02.2024 г.).

10. Google BigQuery. [Электронный ресурс] URL: https://cloud.google.com/bigquery (дата обращения: 02.02.2024 г.).

11. ClickHouse Playground. [Электронный ресурс] URL: https://play.clickhouse.com/play?user=play (дата обращения: 02.02.2024 г.).

12. Github API documentation. [Электронный ресурс] URL: https://docs.github.com/ru/rest?apiVersion=2022-11-28 (дата обращения: 02.02.2024 г.).

13. Visual Studio Code. [Электронный ресурс] URL: https://code.visualstudio.com/ (дата обращения: 02.02.2024 г.).

14. Django. [Электронный ресурс] URL: https://www.djangoproject.com/ (дата обращения: 02.02.2024 г.)

15. Pandas. [Электронный ресурс] URL: https://pandas.pydata.org (дата обращения: 02.02.2024 г.).

16. NumPy. [Электронный ресурс] URL: https://numpy.org (дата обращения: 02.02.2024 г.).

17. GitHub Pattern Mining System. [Электронный ресурс] URL: https://github.com/Santiperro/final-qualifying-work (дата обращения: 02.02.2024 г.).

18. Selenium. [Электронный ресурс] URL: https://www.selenium.dev/ (дата обращения: 02.02.2024 г.).

19. Asyncio. [Электронный ресурс] URL: https://docs.python.org/3/library/asyncio.html (дата обращения: 02.02.2024 г.).

20. Aiohttp. [Электронный ресурс] URL: https://docs.aiohttp.org/en/stable/ (дата обращения: 02.02.2024 г.).

21. Mlxnend. [Электронный ресурс] URL: https://rasbt.github.io/mlxtend/ (дата обращения: 02.02.2024 г.).

22. PostgreSQL. [Электронный ресурс] URL: https://www.postgresql.org/ (дата обращения: 02.02.2024 г.).

ПРИЛОЖЕНИЯ

Приложение А. Интерфейсы основных классов

Листинг 1 – Интерфейс класса `ServiceConnector`

```
class ServiceConnector:
    Класс ServiceConnector предназначен для получения данных из различных
сервисов.

    Атрибуты:
    - CLICKHOUSE_REQUEST_URL (str): URL-адрес для запросов к ClickHouse.
    - GITHUB_API_REQUEST_URL (str): URL-адрес для запросов к API GitHub.
    - REQUEST_DELAY (int): Задержка между запросами.
    - DEFAULT_LIMIT (int): Стандартное ограничение на количество возвращае-
мых данных.
    - DEFAULT_WATCH_EVENT_COUNT (int): Стандартное количество событий
"watch".
    - DEFAULT_MIN_MEMBERS_COUNT (int): Стандартное минимальное количество
участников.
    - TYPES_DICT (dict): Словарь для преобразования типов данных.
    - github_api_token (str): Токен для доступа к API GitHub.
    - headers (dict): Заголовки для запросов к API GitHub.
    - data_configuration (DataFrame): Конфигурация данных.

    Эксопртируемые Методы:
    - async def get_data_from_services(self,
                        transaction_composition: list[str],
                        start_date: str,
                        end_date: str | None = None,
                        limit: int = DEFAULT_LIMIT,
                        min_watch_event_count: int = DE-
FAULT_WATCH_EVENT_COUNT,
                        min_members_count: int = DEFAULT_MIN_MEMBERS_COUNT,
                        is_new_repos: bool = False) -> pd.DataFrame:

        Получение данных из сервисов.

        Параметры:
        transaction_composition: список строк, определяющих состав транзак-
ции.
        start_date: начальная дата в формате 'YYYY-MM-DD'.
        end_date: конечная дата в формате 'YYYY-MM-DD'. Если None, то ис-
пользуется текущая дата.
        limit: максимальное количество записей для получения. По умолчанию
1000.
        min_watch_event_count: минимальное количество событий просмотра. По
умолчанию 10.
        min_members_count: минимальное количество участников. По умолчанию
3.
        is_new_repos: флаг, указывающий на то, являются ли репозитории но-
выми. По умолчанию False.

        Пример transaction_composition:
        ['pushes',
        'avg_push_size',
        'pull_requests',
        'merged_pull_requests_ratio',
        'issues',
        'closed_issues_ratio',
        'watches',
        'forks',
        'new_members',
```

```
        'language',
        'license_name',
        'is_deleted_or_private']

        Возвращает:

        DataFrame с данными.
```

Листинг 2 – Интерфейс класса `GithubDataConverter`

```
class GithubDataConverter:
    Класс GithubDataConverter предназначен для преобразования данных из
GitHub в транзакции.

    Атрибуты:
    - SEPARATION (dict): Словарь для определения квантилей.
    - data_configuration (DataFrame): Конфигурация данных.

    Экспортируемые Методы:
    - convert_data_to_transactions(self, repos_data, quantile_config):
      Метод преобразует данные репозиториев в транзакции.

        Параметры:
        repos_data (pd.DataFrame): DataFrame, содержащий данные репозито-
      риев.
        quantile_config (dict[str, str]): Словарь, определяющий конфигу-
      рацию квантилей для каждого атрибута.

        Пример quantile_config:
        {'pushes': 'dec',
        'avg_push_size': 'dec',
        'pull_requests': 'qua',
        'merged_pull_requests_ratio': 'dec',
        'issues': 'dec',
        'closed_issues_ratio': 'qua',
        'watches': 'dec',
        'forks': 'dec',
        'new_members': 'dec',
        'language': 'None',
        'license_name': 'None',
        'is_deleted_or_private': 'None'}

        Возвращает:
        transactions (pd.DataFrame): DataFrame, содержащий преобразован-
      ные данные репозиториев в виде транзакций.
```

Листинг 3 – Интерфейс класса `PatternMiner`

```
class PatternMiner:
    Класс PatternMiner предназначен для извлечения шаблонов из данных.

    Экспортируемые методы:
    def mine_patterns(self, transactions_matrix, min_supp, min_conf,
min_lift, max_left_elements=3, max_right_elements=1):

        Извлекает шаблоны из матрицы транзакций.

        Параметры:
        - transactions_matrix: Матрица транзакций.
```

```
        - min_supp (float): Минимальное значение поддержки для извлечения
частых наборов.
        - min_conf (float): Минимальное значение уверенности для извлечения
ассоциативных правил.
        - min_lift (float): Минимальное значение подъема для извлечения ас-
социативных правил.
        - max_left_elements (int): Максимальное количество элементов в ан-
тецедентах ассоциативных правил.
        - max_right_elements (int): Максимальное количество элементов в
консеквентах ассоциативных правил.

        Возвращает:
        - DataFrame: DataFrame, содержащий извлеченные ассоциативные пра-
вила.
```

Приложение Б. Функциональное тестирование

В таблице 1 представлены результаты функционального тестирования.

Таблица 1 − Функциональное тестирование

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Загрузка данных при некорректных параметрах.	На странице получения данных нужно выполнить следующие шаги. 1. В поле «Начальная дата» задать дату позже, чем в поле «Конечная дата». 2. Задать отрицательные или вещественные значения в полях «Мин. число звезд репозитория», «Мин. число участников репозитория», «Число репозиториев». 3. Нажать на кнопку «Загрузить данные».	При задаче каждого некорректного параметра будет показываться уведомление и выделяться поле некорректного параметра. Загрузка данных не будет выполнена.	Да
2	Загрузка данных репозиториев с отключением всех атрибутов данных.	На странице получения данных нужно выполнить следующие шаги. 1. В таблице для выбора атрибутов данных в колонке «включить» выключить все атрибуты. 2. Нажать на кнопку «Загрузить данные».	Будет показано уведомление о том, что нужно выбрать минимум 3 атрибута. Загрузка данных не будет выполнена	Да
3	Загрузка данных репозиториев с 13 марта 2024 года по 13 мая 2024 года с минимальным числом звезд 100.	На странице получения данных нужно выполнить следующие шаги. 1. В поле «Начальная дата» задать «03.13.2024». 2. В поле «Конечная дата» задать «05.13.2024». 3. В поле «Мин звезд репозитория» задать значение «100». 4. Нажать на кнопку «Загрузить данные».	На странице под кнопкой «Загрузить данные» отобразится сообщение «Данные успешно сохранены». На странице поиска шаблонов в таблице с выборками можно будет найти текущую выборку.	Да

Продолжение таблицы 1 приложения Б

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
4	Загрузка данных репозиториев с 13 марта 2024 года по 13 мая 2024 года с видом разделения «Квартили» и выключенным атрибутом «Отправленные изменения».	На странице получения данных нужно выполнить следующие шаги. 1. В поле «Начальная дата» задать «03.13.2024». 2. В поле «Конечная дата» задать «05.13.2024». 3. В таблице выбора атрибутов выключить атрибут «Отправленные изменения». 4. В этой же таблице выбрать другое разделение у всех оставшихся атрибутов. 5. Нажать на кнопку «Загрузить данные».	На странице под кнопкой «Загрузить данные» отобразится сообщение «Данные успешно сохранены». На странице поиска шаблонов в таблице с выборками можно будет найти текущую выборку.	Да
5	Проверка поиска шаблонов с минимальной поддержкой и достоверность ю 0.1 и подъемом 1 в загруженной выборке данных репозиториев из теста №1.	На странице поиска шаблонов нужно выполнить следующие шаги. 1. В таблице выборок по текущим параметрам и дате найти текущую выборку и нажать на свитч в колонке «Использовать выборку». 2. В поле «Минимальная поддержка» задать значение «0.1». 3. В поле «минимальная достоверность» задать значение «0.1». 4. В поле «Минимальный подъем» задать значение «1». 5. Нажать на кнопку «Найти шаблоны».	Выполнится поиск шаблонов. Внизу экрана отобразится таблица с найденными шаблонами.	Да
6	Проверка поиска шаблонов в нескольких выборках.	На странице поиска шаблонов нужно выполнить следующие шаги. 1. Загрузить в систему схожую по параметрам выборку из теста №1. 2. В таблице выборок по текущим параметрам и дате найти обе выборки и в колонке «Использовать выборку» нажать на свичи. 3. Оставить другие параметры по умолчанию и нажать на кнопку «Найти шаблоны».	Выборки корректно объединятся и выполнится поиск шаблонов. Внизу экрана отобразится таблица с найденными шаблонами.	Да

Окончание таблицы 1 приложения Б

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
7	Проверка поиска шаблонов в выборках с разным составом атрибутов.	На странице поиска шаблонов нужно выполнить следующие шаги. 1. В таблице выборок по текущим параметрам и дате найти выборки из теста №1 и теста №2, в колонке «Использовать выборку» нажать на свичи. 2. Оставить другие параметры по умолчанию и нажать на кнопку «Найти шаблоны».	Появится предупреждение «Параметры атрибутов выборок не совпадают», поиск шаблонов не выполнится.	Да
8	Поиск шаблонов при некорректных параметрах.	На странице поиска шаблонов нужно выполнить следующие шаги. 1. Задать отрицательные или вещественные значения в полях «Минимально элементов антецедента», «Максимально элементов антецедента», «Минимально элементов консеквента», «Максимально элементов консеквента». 2. Задать отрицательное число или число больше 1 в поле «Порог поддержки» или «Порог достоверности». 3. Задать отрицательное число в поле «Порог подъема». 4. Нажать на кнопку «Найти шаблоны».	При задаче каждого некорректного параметра будет показываться уведомление и выделяться поле некорректного параметра. Поиск шаблонов не будет выполнен.	Да
9	Поиск шаблонов при невыбранных данных для поиска.	На странице поиска шаблонов нужно выполнить следующие шаги. 1. В таблице выбора данных не выбирать наборы данных. 2. Нажать на кнопку «Найти шаблоны».	Будет показано уведомление о необходимости выбора данных для поиска. Поиск шаблонов не будет выполнен.	Да
10	Проверка скачивания шаблонов.	На странице поиска шаблонов нужно выполнить следующие шаги. 1. Выполнить поиск шаблонов. 2. Нажать на кнопку «Скачать шаблоны».	Выполнится скачивание файла Excel с находящимися внутри шаблонами.	Да