Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 5

по курсу «Анализ алгоритмов»

на тему: «Организация параллельных вычислений по конвейерному принципу»

Студент ИУ7-53Б _____ Князев Д. Ю.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Кормановский М. В.
(Подпись, дата) (И. О. Фамилия)

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Система конвейерной (потоковой) обработки данных — система, состоящая из вычислительной и потребляющей частей, в которой задержка между отправкой и приёмом сообщений составляет порядка секунд–минут [1].

Данная система хорошо согласуется с сервис-ориентированной архитектурой, при которой функциональность приложения представляется в виде набора слабо связанных автономных компонентов, называемых сервисами [2].

Цель работы — получение навыка организации параллельных вычислений по конвейерному принципу.

Задачи работы:

— анализ предметной области;

— разработка алгоритма обработки данных;

— создание ПО, реализующего разработанный алгоритм;

— исследование характеристик созданного ПО.

1 Входные и выходные данные

Входными данными являются:

— URL начальной страницы;

— множество URL начальных путей страниц, исключаемых из этапа обработки (может быть пустым);

— максимальное количество загружаемых страниц, где ноль обозначает отсутствие лимита.

Выходными данными являются загруженные в коллекцию СУБД $MongoDB$ $JSON$-документы, содержащие следующую информацию:

— id — уникальный идентификатор рецепта;

— $issue\_id$ — номер задачи из $Redmine$;

— url — URL страницы рецепта;

— *title* — название рецепта;

— *ingredients* — массив ингредиентов, каждый ингредиент — словарь вида (пример на *JSON*) `{"name": название, "unit": единица измерения, "quantity": количество}`;

— *steps* — шаги рецепта, массив строк, одна строка — одно предложение;

— *image_url* — *URL* основного изображения рецепта (если есть).

На рисунке А.1 представлен пример пользовательского интерфейса.

2 Преобразование входных данных в выходные

На рисунке 2.1 изображена диаграмма архитектуры приложения.

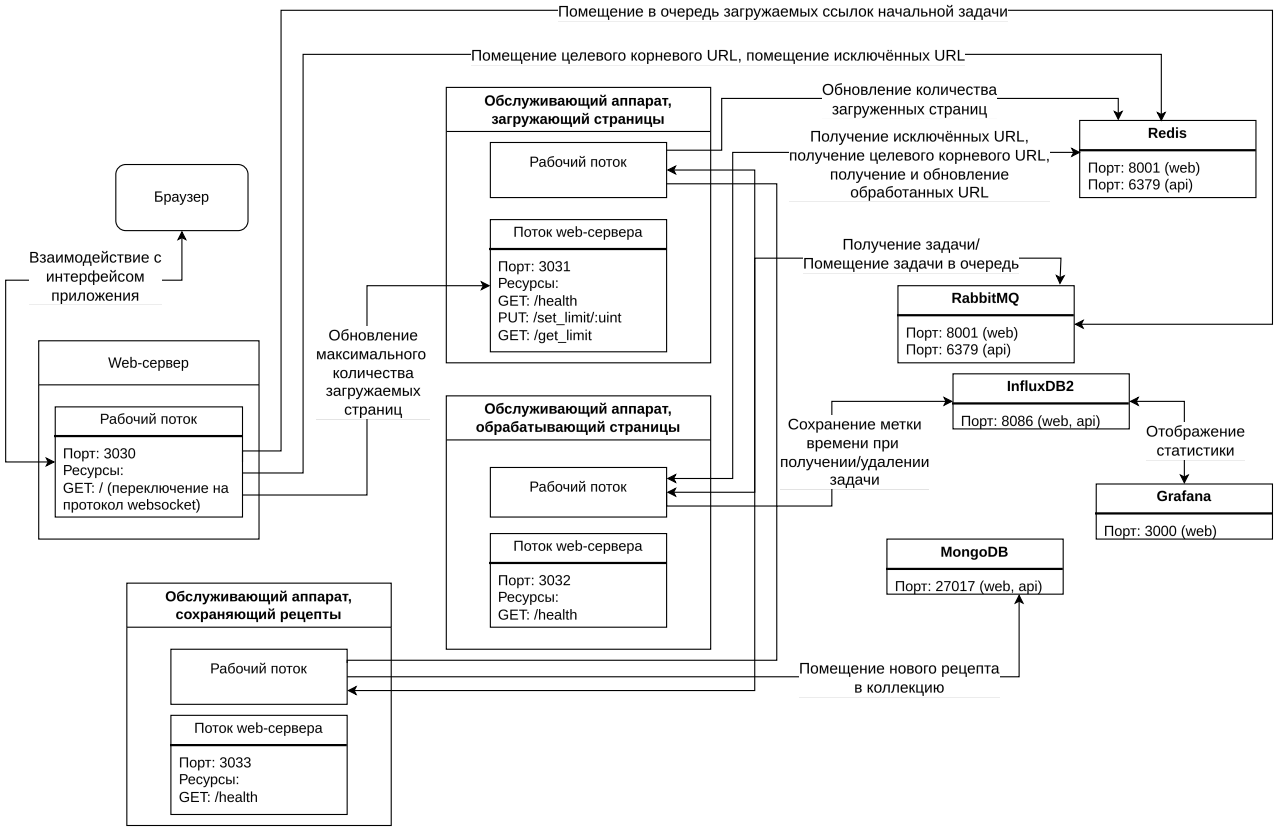

Рисунок 2.1 – Диаграмма архитектуры приложения

Приложение включает в себя следующие сервисы:

— *web*-сервер для получения, обработки и сохранения входных данных пользователя, сервис помещает в очередь новых ссылок первую задачу;

— *RabbitMQ* — брокер сообщений, обслуживающий очередь ссылок страниц, которые необходимо загрузить, также называемую *new_links_queue*, очередь новых страниц *new_documents_queue*, которые необходимо обработать и очередь новых рецептов *new_recipes_queue*, которые необходимо поместить в базу данных документов;

— *MongoDB* — предназначенная для работы с документами СУБД, хранящая обработанные рецепты в формате *JSON*;

— обслуживающий аппарат, загружающий страницы, также называемый *load_automaton*. Сервис получает *URL* из сообщения в очереди *new_links_queue*, загружает соответствующую *HTML*-страницу и помещает в очередь *new_documents_queue* сообщение следующего формата `<url>:<html>`, где `<url>` и `<html>` — *URL* загруженной страницы и её текстовое представление, всё содержимое сообщение кодируется в формате $UTF-8$;

— обслуживающий аппарат, обрабатывающий страницы, также называемый *parse_automaton*. Сервис получает *URL* страницы и её *HTML* наполнение. В результате обработки сервис извлекает множество ссылок, удовлетворяющих входным данным, и добавляет новые ссылки в очередь *new_links_queue*, при наличии на странице рецепта, извлекает его, помещает в соответствующую структуру, сериализует и помещает в очередь *new_documents_queue*;

— обслуживающий аппарат, сохраняющий рецепты, также называемый *store_automaton*. Сервис получает сериализованную структуру-представление *JSON* рецепта, десериализует её и помещает в базу данных *parsed_recipes* в коллекцию *recipes* СУБД *MongoDB*;

— *Redis* — СУБД, хранящая множество обработанных ссылок для исключения их повторной обработки, целевой *URL*, с которым производится сравнение обрабатываемых ссылок (обрабатываемая ссылка не подлежит загрузке, если она не начинается с целевого *URL*), множество исключённых корневых *URL* (обрабатываемая ссылка не подлежит загрузке, если она начинается хотя бы с одного исключённого *URL*), а также

максимальное количество загружаемых страниц и текущее количество загруженных страниц;

— $InfluxDB2$ — СУБД, предназначенная для работы с временными рядами. Данный сервис хранит временные метки событий, связанных с началом и концом обработки задачи обслуживающими аппаратами, а также временные метки поступления задачи в очередь обслуживающего аппарата, загружающего страницы;

— $Grafana$ — BI-инструмент, позволяющий получить такую информацию, как среднее время обработки задачи отдельными обслуживающими аппаратами, среднее время ожидания задачи в каждой очереди и среднее время от начала до конца существования задачи. Перечисленная статистика доступна только для задач, прошедших через все обслуживающие аппараты, остальные задачи в статистике не учитываются.

На рисунках 2.2–2.3 изображена диаграмма последовательности, визуализирующая пример взаимодействия сервисов системы при обработке первой страницы, содержащей 2 новые ссылки и рецепт.

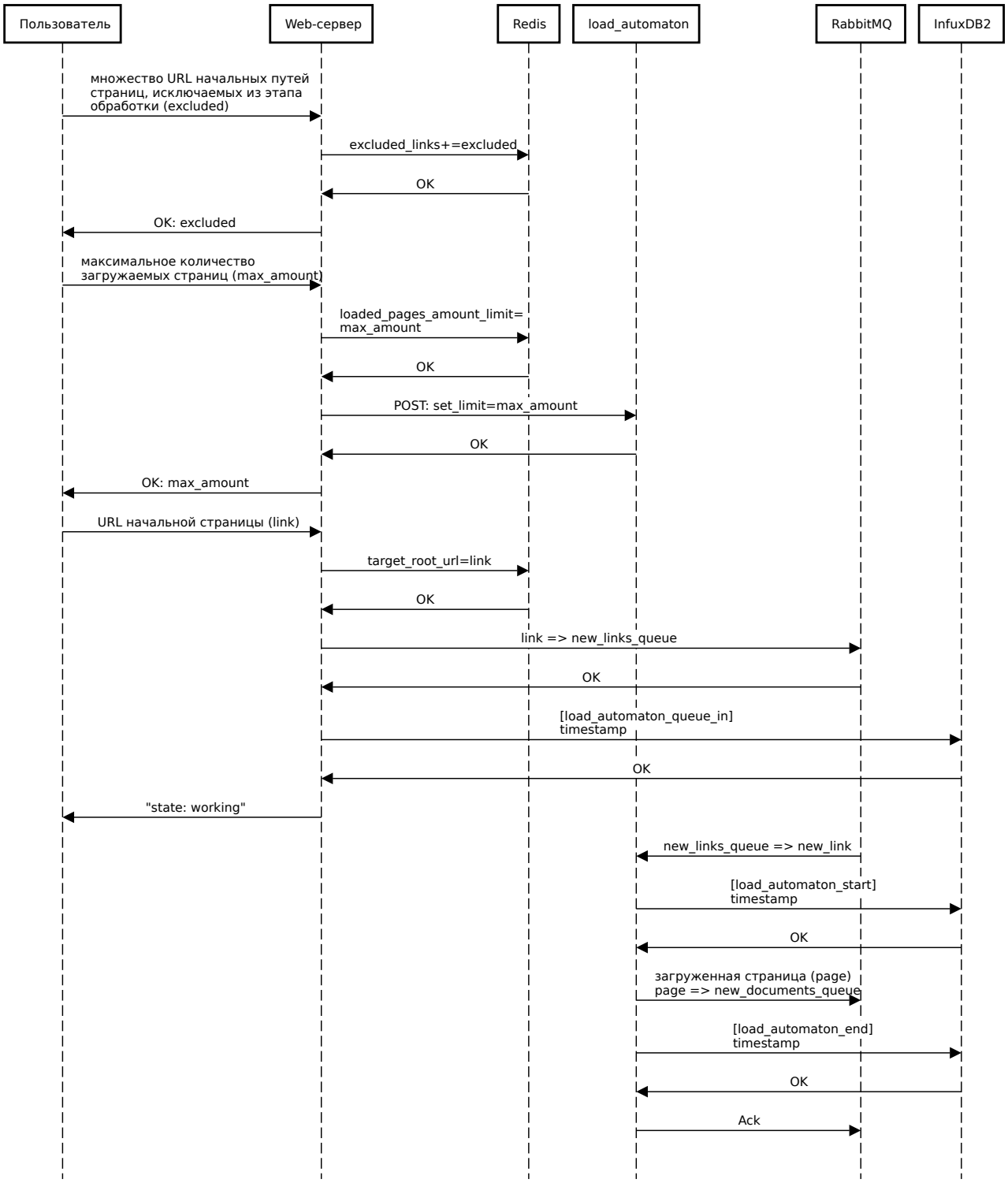

Рисунок 2.2 – Диаграмма последовательности — Начало

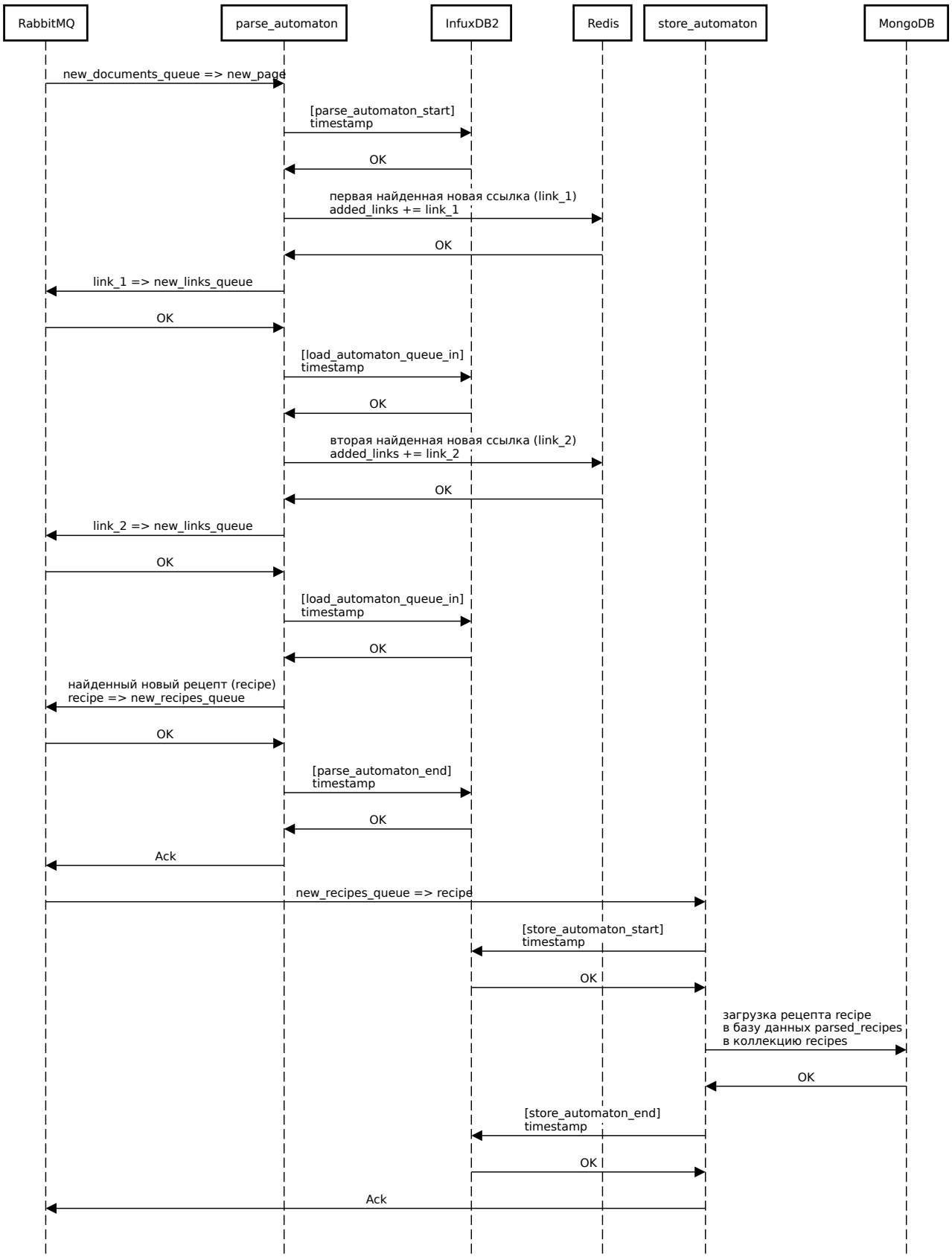

Рисунок 2.3 – Диаграмма последовательности — Конец

3 Тестирование

В листинге 3.1 представлен содержащий рецепт $HTML$-файл, на котором проводилось тестирование.

Листинг 3.1 – Тестовый $HTML$-файл. Многоточием обозначен несущественный текст

```
...
<h2 class="ny__set__title">
Гриль-чиз с луковым джемом </h2>
</div>
...
<div class="ny__details-container__root h2">
<div class="ny__dish-details__title-content__container">
<h1 itemprop="name" class="ny__details-subtitle__root">
Гриль-чиз с луковым джемом </h1>
</div>
...
<div class="ny__dish-details__content-root">
<div class="ny__dish-details__left-side">
<div class="ny__details-container__root content">
<p class="ny__details-subtitle__root"> В наборе</p></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">тостовый хлеб, соус
бешамель, сыр моцарелла, луковый джем</p></br>
<p class="ny__details-subtitle__root">На вашей кухне</p>
<ul></br>
</ul></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">* <img
src="/recipes/images/icons/board.png" width="16" /> <b>доска</b></p></br>
<p class="ny__details-subtitle__root"> Как готовить</p></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Включаем духовку на
**200°C** (режим верх-низ)</p></br>
<hr></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Ломтики тостового
хлеба кладем на противень</p></br>
<hr></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Соусом бешамель
смазываем <b>хлеб</b></p></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Луковый джем
равномерно выкладываем на <b>соус</b></p></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Сыр моцарелла
раскладываем сверху</p></br>
<hr></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Отправляем в
духовку на <img src="/recipes/images/icons/time.png" width="16" /> <b>10 минут</b>
</p></br>
<hr></br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Разрезаем пополам
наискосок и кладем на тарелку</p></br>
<hr></br>
**Если нет духовки, рецепт для сковороды <img src="/recipes/images/icons/pan.png"
width="16" /> можно посмотреть по QR коду на пакете блюда</br>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">или титульном
листе**</p></br>
<hr>
<p class="ny__dishes-details__content" itemprop="recipeInstructions">Корректируйте время
приготовления в соответствии с вашей кухонной техникой</p></br>
...
```

В листинге 3.2 представлен результат работы приложения на тестовых данных.

Листинг 3.2 – Полученный рецепт в формате *JSON*


```
{
        "_id": {
                "$oid": "673fe6706267963c9f480bbf"
        },
        "id": {
                "$numberLong": "1732241008828292228"
        },
        "issue_id": {
                "$numberLong": "9176"
        },
        "url": "https://elementaree.ru/gril-ciz",
        "title": "Гриль-чиз с луковым джемом",
        "ingredients": [
        {
                "unit": "none",
                "amount": "none",
                "name": "В наборе"
        },
        {
                "name": "тостовый хлеб",
                "amount": "none",
                "unit": "none"
        },
        {
                "unit": "none",
                "name": "соус бешамель",
                "amount": "none"
        },
        {
                "name": "сыр моцарелла",
                "unit": "none",
                "amount": "none"
        },
        {
                "name": "луковый джем",
                "unit": "none",
                "amount": "none"
        }
        ],
        "steps": [
        "Включаем духовку на 200°C (режим верх-низ)",
        "Ломтики тостового хлеба кладем на противень",
        "Соусом бешамель смазываем хлеб",
        "Луковый джем равномерно выкладываем на соус",
        "Сыр моцарелла раскладываем сверху",
        "Отправляем в духовку на 10 минут",
        "Разрезаем пополам наискосок и кладем на тарелку или титульном листе**"
        ],
        "image_url": "https://static.elementaree.ru/003241/thumb_m/21fe2f0e0e8e3a6685027e7931e8733e.jpg"
}
```

4 Описание исследования

В ходе исследования требуется сформировать лог обработки задач, найти среднее время обработки задачи в каждом обслуживающем аппарате, среднее время ожидания в каждой очереди, среднее время существования задачи, расчёт среднего времени учитывает только задачи, прошедшие через все обслуживающие аппараты.

Для формирования результирующих данных в *Grafana* были созданы информационные панели, пример которых приведён на рисунке А.2.

В таблице 4.1 отображено среднее время ожидания задач в очередях. Среднее время ожидания задач в очереди новых ссылок на 5 порядков больше по сравнению с остальными очередями.

Таблица 4.1 – Среднее время ожидания задач в очередях

	Среднее время ожидания, мс
Очередь новых ссылок	187536
Очередь новых страниц	4,96
Очередь новых рецептов	9,87

В таблице 4.2 отображено среднее время обработки задач обслуживающими аппаратами. Среднее время обработки обслуживающим аппаратом, загружающим страницы всего на 2–3 порядка больше по сравнению с остальными аппаратами, а наименьшее количество времени занимает сохранение рецепта в базу данных новых рецептов.

Таблица 4.2 – Среднее время обработки задач обслуживающими аппаратами

	Среднее время обработки, мс
load_automaton	1672
parse_automaton	71,6
store_automaton	3,18

В таблице 4.3 приведён фрагмент лога обработки. Обозначения событий:

— la_queue_in — поступление задачи в очередь *new_links_queue*;

— la_start — начало обработки задачи *load_automaton*;

— la_end — окончание обработки задачи *load_automaton*;

— pa_start — начало обработки задачи *parse_automaton*;

— pa_end — окончание обработки задачи *parse_automaton*;

— sa_start — начало обработки задачи *store_automaton*;

— sa_end — окончание обработки задачи *store_automaton*.

Таблица 4.3 – Фрагмент лога обработки (начало)

Событие	ID записи	Метка времени, мкс
load_automaton_queue_in	1731879526378991099	1731879526380238300
load_automaton_start	1731879526378991099	1731879526393213700
load_automaton_end	1731879526378991099	1731879527114415900
parse_automaton_start	1731879526378991099	1731879527128372200
load_automaton_queue_in	1731879527262313032	1731879527262992000
load_automaton_queue_in	1731879527265136152	1731879527265691000
load_automaton_start	1731879527262313032	1731879527266213000
load_automaton_queue_in	1731879527267816238	1731879527268402000
load_automaton_queue_in	1731879527270563999	1731879527271080000
load_automaton_queue_in	1731879527397145903	1731879527397766700
load_automaton_queue_in	1731879527400048572	1731879527400800000
load_automaton_queue_in	1731879527403330479	1731879527403936500
load_automaton_queue_in	1731879527406050933	1731879527406680000
load_automaton_end	1731879527262313032	1731879528621236000
load_automaton_start	1731879527265136152	1731879528626275800
parse_automaton_start	1731879527262313032	1731879528628075000
load_automaton_queue_in	1731879528764736189	1731879528765623600
load_automaton_queue_in	1731879528768203245	1731879528769088000
load_automaton_queue_in	1731879528771304123	1731879528772109800
load_automaton_queue_in	1731879528774291300	1731879528774931700
load_automaton_queue_in	1731879528776817246	1731879528777399300
load_automaton_queue_in	1731879528779823281	1731879528780697300
load_automaton_start	1731879528764736189	1731879614476253700
parse_automaton_start	1731879527406050933	1731879614478183400

Таблица 4.3 – Фрагмент лога обработки (окончание)

Метка времени, мкс	Событие	ID записи
load_automaton_end	1731879528764736189	1731879616384393200
load_automaton_start	1731879528768203245	1731879616389581600
parse_automaton_start	1731879528764736189	1731879616391257900
parse_automaton_end	1731879528764736189	1731879616471264500
store_automaton_start	1731879528764736189	1731879616487485000
store_automaton_end	1731879528764736189	1731879616514776800

По результатам проведенного исследования сделан вывод о том, что события лога упорядочены по возрастанию временных меток, а также о том, что, несмотря на более долгий процесс загрузки отдельной страницы, решающим фактором, влияющим на скорость работы системы в целом является большое количество новых ссылок, подлежащих загрузке. Для увеличения производительности работы приложения следует увеличить количество рабочих потоков в обслуживающем аппарате, загружающем страницы и/или создать несколько экземпляров данного сервиса.

ЗАКЛЮЧЕНИЕ

Цель работы достигнута. Решены все поставленные задачи:

— анализ предметной области;

— разработка алгоритма обработки данных;

— создание ПО, реализующего разработанный алгоритм;

— исследование характеристик созданного ПО.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Пселтис Эндрю Дж.* Потоковая обработка данных. Конвейер реального времени. — ДМК Пресс, 2018. — С. 22—23. — ISBN 978-5-97060-606-3.
2. *Иванович Банокин Павел, Павлович Цапко Геннадий.* Методы и средства проектирования информационных систем и технологий. — Томск : Томский политехнический университет, 2012.

ПРИЛОЖЕНИЕ А

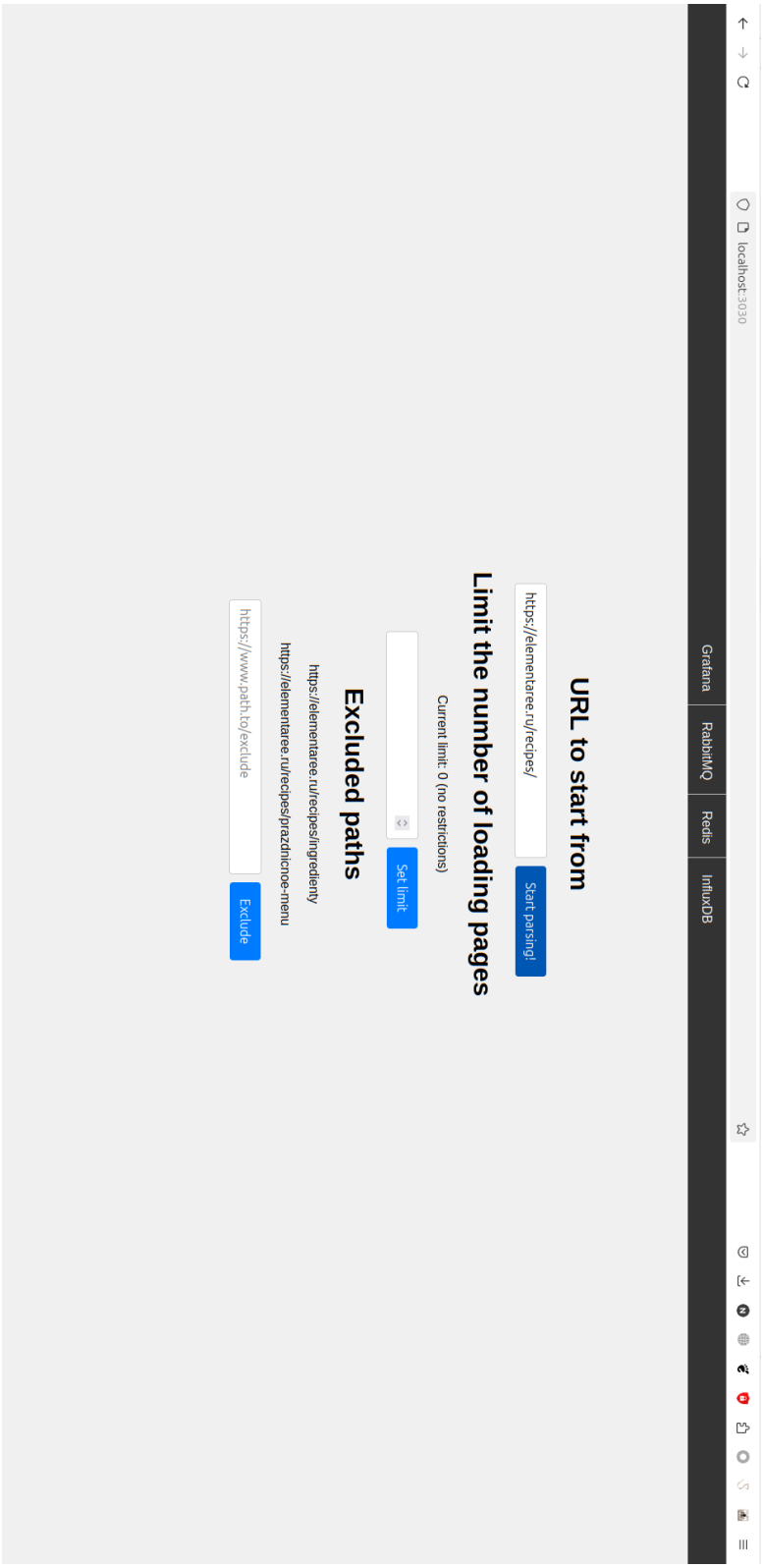

Рисунок А.1 – Пользовательский интерфейс

Рисунок А.2 – Информационная панель *Grafana*