Экз. № ____

Р7 СЕРВЕР - НОВАЯ МАЖОРНАЯ ВЕРСИЯ ПО "Р7 ОФИС ПРОФЕССИОНАЛЬНЫЙ (СЕРВЕРНАЯ ВЕРСИЯ)"

Инв. № подл	Подп. и дата	Взам. инв №	Инв. № дубл	Подп. и дата

Листов 4

2023

1 Описание архитектуры

Схема архитектуры приведена на рисунке ниже

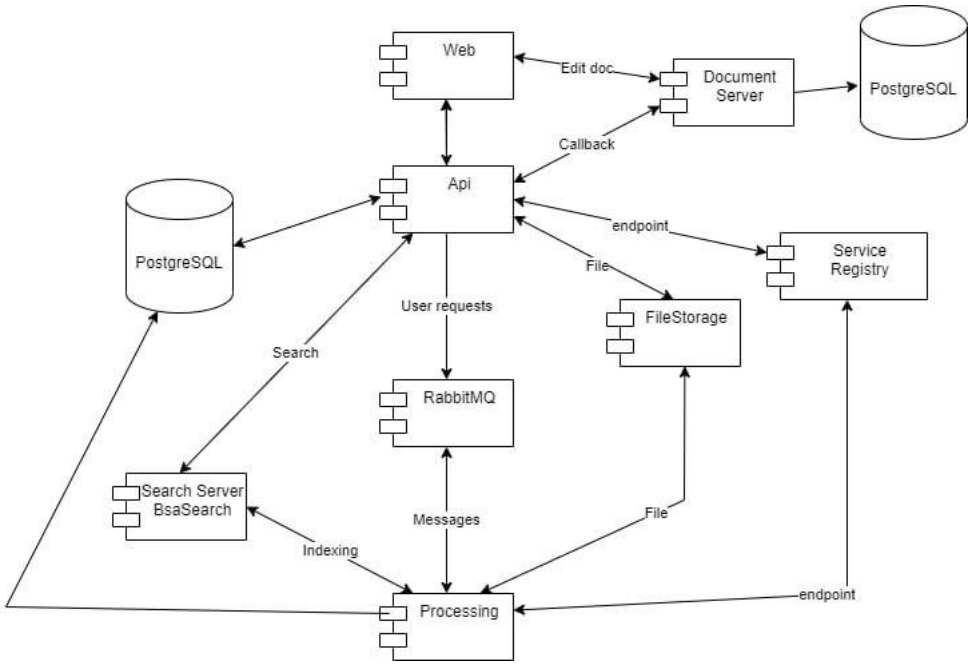

Рисунок. Схема архитектуры

На схеме отображены следующие компоненты:

- **Web** — веб-сервер, к которому обращаются клиентские приложения;
- **Document Server** — сервер документов, отдельный компонент, внедренный в веб-страницу при помощи JS скриптов;
- **API** — программный интерфейс для взаимодействия между собой компонентов решения;
- **PostgreSQL** — БД сервера документов;
- **PostgreSQL** — БД, где хранятся пользовательские данные, идентификаторы документов и файлов, права доступа к ним, данные календарей и почты (модули «Диск», «Почта», «Модуль администратор» и «Календарь», пользовательские события и сообщения, вложения к сообщениям;
- **RabbitMQ** — брокер сообщений для взаимодействия между собой компонентов решения;
- **FileStorage** — файловая система. Это REST API, поддерживающее доступ по протоколам TCP и HTTP. В запросе в FileStorage передается идентификатор файла. В ответ на запрос FileStorage возвращает файл. FileStorage имеет древовидную структуру хранения. Глубина хранения (вложенность папок) задается в настройках:

```
{
  "RootDirectory": "/FileStorage",
  "IsMainNode": false,
  "ElementsCount": 6,
  "TreeDepth": 3,
  "Connection": {
    "Tcp": {
      "Address": "127.0.0.1",
```

```
        "Port": 11581
      }
    }
  }
```

- **Service Registry** — реестр сервисов. Данный компонент реализован по методологии SOA (Service Oriented Architecture). Service Registry хранит данные о местоположении сервисов. API и Processing обращаются к Service Registry с запросами о том, как обратиться к тому или иному сервису, Service Registry предоставляет данную информацию (endpoint сервиса). При перемещении того или другого сервиса на другие аппаратные мощности достаточно обновить эти данные в Service Registry. Также Service Registry выполняет балансировку запросов методом Round Robin и выдает в ответ на запросы разные endpoint;
- **Search Server BsaSearch** — поисковый сервер. Хранит поисковые индексы для документов и почтовых сообщений. В качестве поискового сервера используется библиотека .NET Core, обеспечивающая индексацию и быстрый полнотекстовый поиск, подсветку объектов. Поисковый движок проверен на большой нагрузке и подходит для сложных аналитических запросов. Обрабатывает до 100 тысяч простых запросов в минуту на индексе из 40 млн документов, или примерно 8 тысяч сложных аналитических запросов. К поисковому серверу напрямую обращаются компоненты:
  - **Processing** — для индексации;
  - **API** — для поиска.
- **Processing** — осуществляет индексацию документов и почтовых сообщений.

1.1 Общий порядок взаимодействия

Клиент направляет запрос к веб-серверу. Далее пользовательские запросы маршрутизируются:

- Запросы на редактирование документов направляются напрямую на Document Server.
- Веб-сервер обращается к API для генерации временной ссылки, с помощью которой можно скачать документ. Веб-сервер передает URL, по которому можно получить документ. Document Server обращается к API, скачивает этот документ и отображает его. Callback – информирует, что с документом произошли какие-то изменения. У Document Server есть свой брокер сообщений RabbitMQ и БД PostgreSQL.
- Прочие пользовательские запросы, кроме поисковых, становятся в очередь в RabbitMQ. Поисковые запросы направляются на поисковый сервер.

1.2 Порядок работы пользователя с документом

Пользователь открывает документ в веб-интерфейсе. После проверки прав доступа ему передается ссылка на этот файл.

Файл скачивается, создается версия файла.

Доступны 2 варианта сохранения:

- при закрытии документа — сохранение происходит, когда документ закрывают все пользователи;
- периодическое сохранение с настраиваемым интервалом времени.

2 Преимущества

Текущее решение представляет собой N-звенную архитектуру.

Все компоненты могут быть установлены на разных серверах, что обеспечивает возможность горизонтального масштабирования. Под управлением сертифицированных Astra Linux 1.7, ОС РЕД ОС 7.3. Не использующие при работе с данными Elasticsearch MySQL. Реализация веб-интерфейса с использованием React и Node.js веб-сервера.

Веб-сервер не зависит от API, для реализации API используется локально собранный фреймворк .NET Core, который является свободным и открытым кросс-платформенным решением, а так же в РедОС и других дистрибутивах имеются уже протестированные и одобренные пакеты .NET Core. При взаимодействии веб-сервера и API не сохраняется информация о предыдущих состояниях или сеансах пользователя. Каждый запрос рассматривается как отдельное изолированное взаимодействие.

Такое решение обеспечивает кратную масштабируемость и отказоустойчивость.

Может быть развернуто любое количество веб-серверов и таким образом снижена нагрузка на веб-часть.

Также может быть развернуто любое количество API.

Принцип CQRS (The Command and Query Responsibility Segregation) разделяет назначение запросов и команд на обработку данных.

Операции над данными при использовании данного принципа разделяются на две категории:

- команды, которые вносят изменения в состояние системы;
- запросы — операции получения данных, без внесения изменений в состояние.

Например, в CQRS операция чтения (Read) будет являться запросом, т.к. с ее помощью просто предоставляются данные для просмотра. А операции Create, Update, Delete в данном подходе будут являться командами, которые так или иначе изменяют состояние.

Использование данного подхода улучшает производительность, масштабируемость и безопасность приложения.