УМНЫЙ ВЫБОР
МЕНЯЮЩИХСЯ
ТЕХНОЛОГИЙ

ООО «ИБС Экспертиза»
ОГРН 1067761849704, ИНН/КПП 7713606622/771301001

Россия, 127434, Москва, Дмитровское шоссе, дом 9Б,
этаж 5, помещение XIII, комната 6
телефон/факс: +7 (495) 967 80 80
ibs@ibs.ru, www.ibs.ru

Руководство администратора

системы

«Информационная система автоматизированного формирования спецификаций на разработку при помощи искусственного интеллекта для аналитиков в проектах»

Оглавление

Термины и сокращения

Термины и сокращения, используемые в настоящем документе, приведены в таблице.

Термин, сокращение	**Определение**
ИИ	Искусственный интеллект
LLM	Большие языковые модели
RAG	Метод оптимизации ответов большой языковой модели путем интеграции внешних, актуальных данных из баз знаний, документов или сети Интернет
Векторизация текста	Процесс преобразования текстовых данных в числовые векторы, понятные алгоритмам машинного обучения
Артефакт	Любой искусственно созданный объект, результат работы команды разработки или проектирования: документы, исходный код, файлы, отчеты или иные элементы
Чат-бот	Компьютерная программа, имитирующая диалог с человеком в текстовом или голосовом формате, работающая в мессенджерах, приложениях или на сайтах
Чанкинг	Метод, заключающийся в разбиении больших объемов информации на мелкие смысловые группы (блоки)
Оркестрация	Автоматизация управления, координация и настройка взаимодействия множества контейнеров, сервисов, приложений или данных, заставляющая их работать как единое целое
Графовая индексация	Метод структурирования данных, при котором узлам (сущностям) и ребрам (связям) присваиваются уникальные идентификаторы или числовые значения
Wiki-страница	Веб-страница, содержимое которой можно создавать, редактировать, форматировать и структурировать через браузер, используя специальную разметку
Брокер сообщений	Промежуточное программное обеспечение, выступающее посредником при обмене данными между компонентами распределенных систем (микросервисами, приложениями)

1 Назначение Системы

1.1 Назначение Системы

Информационная система автоматизированного формирования спецификаций на разработку при помощи искусственного интеллекта для аналитиков в проектах предназначена для автоматизированного формирования спецификаций на разработку при помощи искусственного интеллекта для аналитиков в проектах в соответствии с заданным шаблоном.

Система позволяет автоматически формировать и поддерживать актуальное описание информационной системы на основании артефактов, разработанных аналитиком (бизнес-требования, технические задания, диаграммы, пользовательские сценарии и т.д.), задач в системах управления задачами (Jira или аналогах), программного кода и конструктора алгоритмов. Описание представляет собой сгенерированную wiki-страницу о решении, описывающую бизнес-функции информационной системы, архитектуру решения, интеграции, компоненты и пользовательские инструкции.

1.2 Функции Системы

Система обеспечивает следующие функциональные возможности:

- авторизация в системе с корпоративными учетными данными;
- загрузка внешних данных из одного или нескольких источников;
- проверка качества данных и их валидация;
- обработка ошибок;
- актуализация шаблонов для генерации документов;
- актуализация перечня используемых для генерации документов проектов;
- загрузка данных из выбранного файла;
- загрузка данных из выбранной задачи Jira;
- указание заголовка для генерируемой страницы;
- генерация документа на основании указанных параметров;
- просмотр сгенерированного документа;
- просмотр версий сгенерированного документа;
- формирование запроса к LLM для внесения изменений в сформированный документ посредством диалога с чат-ботом;
- выгрузка сформированного документа в форма Markdown;
- выгрузка любой версии сформированного документа.

1.3 Функции модуля администрирования

В Системе отсутствует отдельный модуль администрирования.

Используемые для индексации проекты, настраиваются в стороннем сервисе RagnaRock (не является компонентом описываемой системы). Синхронизация с Системой осуществляется через брокера сообщений RabbitMQ, в который RagnaRock записывает ссылки на проекты в Confluence.

Настройка следующих сущностей осуществляется непосредственно в базе данных Системы:

- шаблоны для генерации документов;
- перечень пространств Jira.

2 Архитектура и взаимодействие модулей

2.1 Архитектура Системы

Целевая архитектура Системы приведена на Рис. 1.

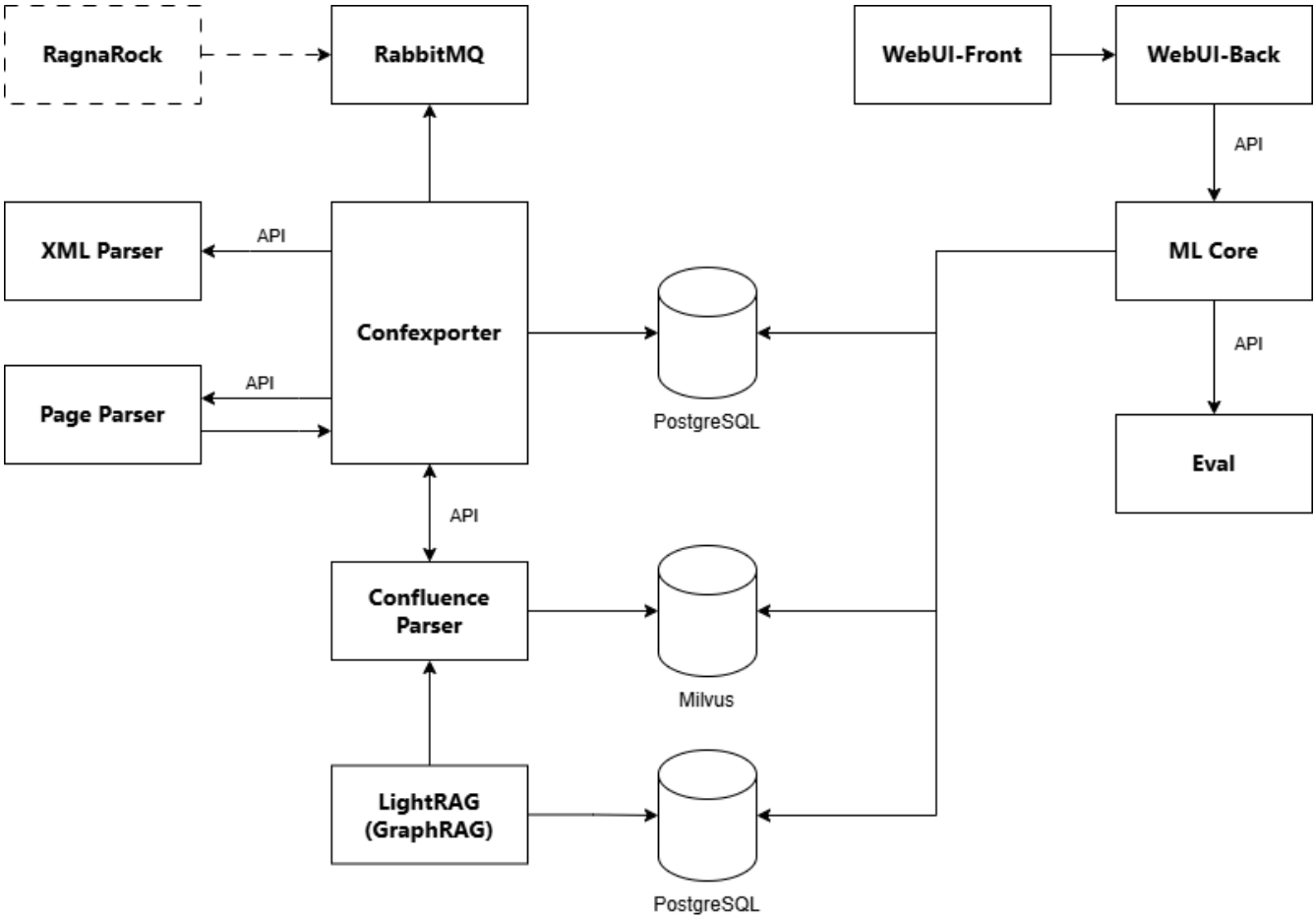

Рис. 1 – Взаимодействие модулей.

2.2 Взаимодействие модулей

Целевая архитектура Системы построена на основе микросервисной архитектуры.

Система реализована в виде интегрированных между собой программных компонентов (подсистем и сервисов), имеющих следующие основные функции.

Имя сервиса	**Описание**
WebUI-Front	Пользовательский интерфейс для генерации документов
WebUI-Back	Пользовательский интерфейс для генерации документов
Confexporter	Сервис-оркестратор для маршрутизации и сохранения первичных данных из Confluence
RabbitMQ	Брокер сообщений для асинхронной передачи задач
Page Parser	Сервис очистки HTML-страниц и конвертации в Markdown
XML Parser	Обработчик XML схем
Confluence Parser	Сервис чанкинга и векторизации документов
LightRAG (GraphRAG)	Сервис графовой индексации знаний
ML Core	Сервис генерации документации с использованием LLM/RAG
Eval	Сервис расчета метрик качества генерации

2.2.1 Загрузка внешних данных

Описание взаимодействия модулей при загрузке внешних данных из Confluence:

1. Сервис Confexporter вычитывает контент настроенных проектов из очереди RabbitMQ.
2. Сервис Confexporter определяет тип полученного из очереди элемента контента.
3. Если текущий элемент является XML-схемой, формируется и направляется запрос в сервис XML Parser. На этом обработка текущего элемента контента завершается.
4. Если текущий элемент является HTML-страницей, формируется и направляется запрос в сервис Page Parser для осуществления преобразования в Markdown.
5. Сервис Page Parser осуществляет парсинг и очистку полученного HTML-кода: удаление навигационных элементов, скриптов и стилей, внешних ссылок. В результате выполнения данных операций формируется чистый Markdown-текст и возвращается в виде ответа.

6. Сервис Confexporter записывает полученные данные в БД PostgreSQL confexporter.
7. Сервис Confexporter формирует и направляет запрос, передавая полученный Markdown-текст в сервис Confluence Parser.
8. Сервис Confluence Parser разбивает полученный текст на чанки и записывает в БД Milvus.
9. Сервис GraphRAG по настроенному расписанию формирует и направляет запрос в сервис Confexporter для получения необходимых страниц целиком.
10. Сервис Confexporter извлекает запрошенные страницы из своих БД PostgreSQL и возвращает в качестве ответа на запрос.
11. Сервис GraphRAG на основании полученных данных строит граф знаний (сущности и связи) и сохраняет индексированные данные в БД PostgreSQL postgres-lightrag-prod.

2.2.2 Генерация документа

Описание взаимодействия модулей при генерации документа:

1. Сервис WebUI-Front формирует и направляет запрос в сервис WebUI-Back для передачи введенных и выбранных пользователем на форме параметров формирования документа.
2. Сервис WebUI-Back формирует и направляет запрос в сервис ML Core.
3. Сервис ML Core осуществляет валидацию полученных входных данных.
4. Сервис ML Core осуществляет формирование промпта и передачу его в LLM.
5. Сервис ML Core осуществляет пост-процессинг сгенерированного документа и отображение его пользователю.
6. Кроме того, сервис ML Core осуществляет логирование процесса генерации документа. Собранные данные передаются в сервис Eval для расчета метрик качества генерации.

3 Выполнение Системы

3.1 Развертывание системы

Сервисы Системы развернуты на одном хосте без помощи системы оркестрации. Для изоляции приложений и поставки зависимостей используется контейнеризация при помощи Docker.

Таким образом, запуск сервиса сводится к одной команде:

```
docker run -d --name socrates.service --env-file /opt/service/.env -p 8020:8000 -v /opt/service/history:/app/history --restart unless-stopped docker_image:tag
```

Автоматизация развертывания выполняется в репозитории socrates/infra_release-orchestrator. Этот проект представляет собой GitLab CI, предназначенный для автоматизации процесса

деплоя сервисов Системы, обеспечивает централизованное управление и развертывание, а также конфигурирование сервисов.

Порядок действий:

1. Перейти в раздел меню Build → Pipelines, нажать кнопку "New pipeline".
2. Заполнить переменные только для нужных сервисов версией сервиса.
3. Нажать кнопку "New pipeline".
4. Выбрать сервисы для деплоя / перезапуска / переконфигурации.

В качестве значений переменных используется версия сервиса, которая передается как параметр tag для выбора docker-образа. Возможное значение - существующая версия образа в registry (Harbor). Пример значения: 1.0.0

Если оставить значение пустым, сервис затронут не будет. Если задать какую-то из переменных или все сразу, произойдет деплой и перезапуск соответствующего сервиса.

Переменная	**Сервис**
CONFEXPORTER_VERSION	Обрабатывает документы Confluence, записывает их в базу, передает другим сервисам для обработки
CONFLUENCE_PARSER_VERSION	Предоставляет инструменты для обработки документов в Markdown формате и индексации в векторной базе данных Milvus
EVAL_VERSION	Сервис для расчета различных метрик качества текста
GRAPHRAG_VERSION	Сервис для обработки документов и их индексации в графовой базе знаний с использованием LightRAG
MLCORE_VERSION	Сервис для генерации бизнес-аналитической документации (спецификаций) с использованием методологий RAG (Retrieval-Augmented Generation), Few-Shot Learning и мультимодальных моделей
PAGE_PARSER_VERSION	Сервис для конвертации HTML-контента в формат Markdown с дополнительной очисткой
WEBUI_BACK_VERSION	Бэкенд для фронтенда, обеспечивает авторизацию, хранение данных пользователя и т.д.
WEBUI_FRONT_VERSION	Основной фронтенд

XML_PARSER_VERSION	Сервис для парсинга диаграмм баз данных и индексации их в векторной базе данных Milvus

Переменные окружения задаются в разделе Settings → CI/CD → Variables и могут быть изменены в случае необходимости. Обязательные переменные для всех сервисов.

Переменная	**Комментарий**
DEPLOY_HOST	Адрес сервера, на котором должны быть развернуты сервисы
DEPLOY_USER	Технический пользователь для удаленного доступа через SSH, с помощью которого происходит деплой
SSH_PRIVATE_KEY	Ключ для авторизации на удаленном сервисе, который должен быть добавлен в разрешенные
CI_REGISTRY	Адрес хранилища образов registry
CI_REGISTRY_USER	Пользователь для загрузки образов из registry
CI_REGISTRY_PASSWORD	Пароль пользователя registry

Следующие переменные необходимы для деплоя соответствующих сервисов, подгружаются в качестве файла с переменными сервиса в Docker, они должны иметь тип "файл" и содержать в себе список необходимых переменных:

- ENV_CONFEXPORTER
- ENV_CONFLUENCE_PARSER
- ENV_EVAL
- ENV_GRAPHRAG
- ENV_MLCORE
- ENV_PAGE_PARSER
- ENV_WEBUI_BACK
- ENV_WEBUI_FRONT
- ENV_XML_PARSER

Должны быть развернуты следующие сервисы Системы. Взаимодействие между сервисами осуществляется по протоколу TCP с использованием REST-API.

Имя сервиса	**Порт внутри контейнера**	**Опубликованный порт**
WebUI-Front	3000, 9229	3120, 9229

Имя сервиса	Порт внутри контейнера	Опубликованный порт
WebUI-Back	80	8505
Confexporter	8080, 8081	8102, 8103
Page Parser	8000	8111
XML Parser	8000	8100
Confluence Parser	8000	8101
LightRAG (GraphRAG)	8000	8888
ML Core	8000	8001
Eval	8000	8099

3.1.2 Базы данных

3.1.2.1 PostgreSQL

Для сервисов проекта используется инстанс PostgreSQL версии 15 с OpenSource-дополнениями Apache AGE и pgvector. Требования сервисов к базам данных приведены в таблице.

Сервис	База данных	Расширения
Confexporter	confexporter	
LightRAG (GraphRAG)	postgres-lightrag-prod	Apache AGE, pgvector
ML Core	postgres-log-prod	
WebUI-Back	postgres-log-prod webui-history	

Инстанс базы данных Postgres развернут на том же хосте с использованием контейнеризации в Docker. Деплой инстанса автоматизирован с помощью ansible-роли в репозитории socrates/infra_ansible в директории roles/postgres_docker.

Сборка для контейнерного запуска модифицированного образа postgres-15:

```
FROM postgres:15
LABEL maintainer="Socrates Postres 15 with Apache age and pgvector."
RUN apt-get update \
```

```
    && apt-get install -y curl \
    && apt-get -y install postgresql-15-pgvector \
    && apt-get -y install postgresql-15-age
```

Сервисы не применяют миграции перед стартом, поэтому БД для соответствующих сервисов должны быть подготовлены перед запуском во избежание ошибок.

Скрипт для первичной инициализации БД postgres-lightrag-prod, оригинал расположен в репозитории socrates/graphrag:

```
\c template1;
-- Увеличение времени до определения deadlock
ALTER SYSTEM SET deadlock_timeout = '5s';
-- Отключение параллельного создания индексов по умолчанию
ALTER SYSTEM SET maintenance_work_mem = '1GB';
-- Параллельные воркеры для создания индексов
SET max_parallel_maintenance_workers = 4;
-- Увеличение лимита блокировок
ALTER SYSTEM SET max_locks_per_transaction = 128;
CREATE EXTENSION IF NOT EXISTS vector;
CREATE EXTENSION IF NOT EXISTS age;
LOAD 'age';
ALTER DATABASE template1 SET search_path = public, ag_catalog, "$user";
SET search_path = public, ag_catalog, "$user";
CREATE EXTENSION IF NOT EXISTS btree_gin;
\c postgres-lightrag-dev;
CREATE EXTENSION IF NOT EXISTS vector;
CREATE EXTENSION IF NOT EXISTS age;
LOAD 'age';
ALTER DATABASE "postgres-lightrag-prod" SET search_path = public, ag_catalog,
"$user";
SET search_path = public, ag_catalog, "$user";
CREATE EXTENSION IF NOT EXISTS btree_gin;
-- Применить изменения настроек
SELECT pg_reload_conf();
```

Скрипт для первичной инициализации БД postgres-log-prod и webui-history, оригинал расположен в репозитории socrates/mlcore:

```
-- Основная таблица с метаданными запроса
CREATE TABLE generation_requests (
    id UUID PRIMARY KEY,
    timestamp TIMESTAMPTZ DEFAULT NOW(),
    document_type VARCHAR(1000) NOT NULL,
    project VARCHAR(1000) NOT NULL,
    page_name VARCHAR(1000) NOT NULL,
    attached_link VARCHAR(1000)
);
-- Таблица входных данных
CREATE TABLE generation_inputs (
    request_id UUID PRIMARY KEY REFERENCES generation_requests(id) ON DELETE
CASCADE,
    user_question TEXT,
    jira_text TEXT,
    previous_generations_summary TEXT,
    few_shot_examples JSONB
);
-- Таблица контекстных данных
CREATE TABLE generation_contexts (
    request_id UUID PRIMARY KEY REFERENCES generation_requests(id) ON DELETE
CASCADE,
    document_context JSONB,
    table_context TEXT,
    graph_context TEXT
);
-- Таблица результатов
CREATE TABLE generation_results (
    request_id UUID PRIMARY KEY REFERENCES generation_requests(id) ON DELETE
CASCADE,
    generated_answer TEXT,
    timings JSONB,
    metrics JSONB
);
-- Таблица правок пользователей
CREATE TABLE generation_edits (
    edit_id UUID PRIMARY KEY,
```

```
    request_id UUID NOT NULL REFERENCES generation_requests(id) ON DELETE
CASCADE,
    edited_answer TEXT NOT NULL,
    user_question TEXT,
    timestamp TIMESTAMPTZ DEFAULT NOW()
);
-- Таблица оценок пользователей
CREATE TABLE generation_ratings (
    edit_id UUID PRIMARY KEY REFERENCES generation_edits(edit_id) ON DELETE
CASCADE,
    user_rating INTEGER
);
-- Индекс для ускорения поиска по request_id
CREATE INDEX idx_generation_edits_request_id ON generation_edits(request_id);
\c postgres;
-- Создаем новую базу данных для истории
CREATE DATABASE "webui-history";
\c "webui-history";
-- Создаем таблицы в новой базе данных
CREATE TABLE document_history (
    id UUID PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    template_type VARCHAR(255) NOT NULL,
    page_name TEXT NOT NULL,
    project_name VARCHAR(255) NOT NULL,
    jira_filename VARCHAR(255),
    response TEXT NOT NULL,
    request_id VARCHAR(255),
    edit_id VARCHAR(255),
    attached_link TEXT,
    context_documents JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    jira_issue_key varchar(50)
);
-- Создание индексов для оптимизации поиска
CREATE INDEX idx_document_history_username ON document_history(username);
```

```
CREATE INDEX idx_document_history_project_page ON document_history(project_name, page_name);
CREATE INDEX idx_document_history_template ON document_history(template_type);
-- Создание таблицы для хранения версий документов
CREATE TABLE document_versions (
    id SERIAL PRIMARY KEY,
    doc_id UUID NOT NULL REFERENCES document_history(id) ON DELETE CASCADE,
    version INT NOT NULL,
    content TEXT NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    template_type VARCHAR(255) NOT NULL,
    page_name TEXT NOT NULL,
    request_id VARCHAR(255),
    edit_id VARCHAR(255),
    source VARCHAR(100) NOT NULL,
    user_question TEXT,
    attached_link TEXT,
    context_documents JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    jira_issue_key varchar(50),
    CONSTRAINT unique_doc_version UNIQUE(doc_id, version)
);
-- Создание индексов для оптимизации поиска версий
CREATE INDEX idx_document_versions_doc_id ON document_versions(doc_id);
CREATE INDEX idx_document_versions_edit_id ON document_versions(edit_id);
-- Создание таблицы для хранения текстов + изображений
CREATE TABLE text_images_history(
    id SERIAL PRIMARY KEY,
    request_id VARCHAR(255),
    request_text TEXT NOT NULL,
    images JSONB
);
CREATE INDEX idx_text_images_history_request_id ON text_images_history(request_id);
```

3.1.2.2 Milvus DB

Для хранения векторизированных данных из внешних систем используется векторная база данных Milvus DB, решения с открытым исходным кодом от MilvusIO. Версия не ниже 2.5.4.

Перечень сервисов, которым требуется подключение к векторной базе данных.

1. Confexporter
2. Confluence Parser
3. ML Core
4. XML Parser

В случае запуска Системы с пустым хранилищем необходима первичная инициализация пространств проектов через сторонний сервис RagnaRock (3.2 Инициализация проектов).

Разворачивание экземпляра Milvus производится с помощью shell-скрипта для запуска standalone_embed.sh, опубликованного в репозитории socrates/infra_ansible в директории docker_manual/milvus_bd. Скрипт запускает Milvus в виде контейнера Docker (должен быть установлен на машине). Для первого запуска потребуется скачать образ milvusdb/milvus:v2.5.4 с Docker Hub. Порядок работы со скриптом следующий:

1. Перед запуском необходимо перенести содержимое этой директории на целевой сервер. Например, будем использовать директорию /opt/milvus/.
2. Файл standalone_embed.sh должен быть исполняемым - используйте команду chmod a+x /opt/milvus/standalone_embed.sh для установки соответствующего признака.
3. Выполните команду для запуска milvus: /opt/milvus/standalone_embed.sh start.
4. Другие доступные команды: restart | stop | upgrade | delete.

Требования к машине для запуска этой векторной БД:

1. Не менее 2 ядер CPU и 6 Гб RAM.
2. Не менее 10 Гб свободного места на диске (в дальнейшем размер БД будет увеличиваться при добавлении новых проектов).
3. Производительный диск для директории с хранилищем. В случае запуска скрипта из /opt/milvus/standalone_embed.sh это будет /opt/milvus/volumes/milvus. Хранилище состоит из множества мелких файлов (чанков коллекций внутри базы и других сущностей). Рекомендуется размещение на машине с NVME-диском или виртуальным диском в быстром хранилище.
4. Возможно разворачивание в режиме использования GPU на машине с видеокартой. Текущий скрипт не предусматривает этого, нужна модификация команды запуска и подключение "nvidia-container-runtime" в файле /etc/docker/daemon.json.

3.2 Инициализация проектов

Для отображения проекта из базы знаний в качестве источника при генерации проект должен присутствовать в виде коллекции (с UID) в базе инстанса Milvus, с которым работает сервис ML Core, а также в БД confexporter. Если проект отсутствует в списке, или если Система развернута на пустых БД (например, при новой инсталляции), перед созданием генераций документов необходимо инициировать проект через сервис Ragnarock.

Порядок инициализации нового проекта:

1. Залогиниться в сервисе Ragnarock.
2. Добавить ссылку на проект в Confluence.
3. Нажать кнопку "Запустить индексацию".
4. Дождаться появления страниц, выделить все и нажать кнопку «Разрешить».
5. Дальнейшая обработка осуществляется в соответствии с 2.2.1 Загрузка внешних данных.
6. При необходимости, для принудительной синхронизации данных после очистки очереди RabbitMQ можно в командной строке выполнить запрос:

```
curl -X POST "http://<адрес_сервиса_ LightRAG>:8888/sync/force"
```

3.3 Настройки сервисов

3.3.1 Сервис Confluence Parser

Настройки сервис Confluence Parser задаются в фале config.yaml.

Пример файла:

```
text_processing:
  vector_dim: 1024
  chunk_size: 1000
  chunk_overlap: 150
  batch_size: 20
```

Задаются настройки чанкинга и векторизации документов.

Параметр	**Описание**
vector_dim	Размерность вектора
chunk_size	Размерность чанка
chunk_overlap	Перекрытие чанков
batch_size	Размерность пакета

3.3.2 Сервис ML Core

Настройки сервиса ML Core задаются в файле config/default_config.yaml.

Пример файла:

```
gen_params:
```

```
  num_generations: 1
  is_few_shot: true
  is_context: true
  is_summary: true
  is_table: true
  is_graph: true
models_names:
  embedder: "deepvk/USER-bge-m3"
  reranker: "DiTy/cross-encoder-russian-msmarco"
  LLM: "qwen2.5-72b-instruct-gptq-int8"
  VLM: "Qwen2.5-VL-7B-Instruct"
models_params:
  embedder_out_chunks: 100
  reranker_out_chunks: 15
  attached_out_chunks: 5
  num_table_out: 5
  graph_out_chunks: 10
```

Задаются настройки LLM.

Параметр	**Описание**
num_generations	Количество генераций для данного запроса
is_few_shot	Даем ли модели пример входа и выхода
is_context	Собираем ли контекст из векторной БД
is_summary	Учитываем ли саммаризацию
is_table	Используем ли поиск по таблицам
is_graph	Используем ли поиск по графу знаний
embedder	Используемый эмбедер
reranker	Используемый реранкер

Параметр	Описание
LLM	Используемая LLM
VLM	Используемая VLM
embedder_out_chunks	Количество чанков, получаемых из векторной БД
reranker_out_chunks	Количество чанков, получаемых от реранкера
attached_out_chunks	Количество чанков, получаемых со страницы, указанной пользователем в параметрах генерации документа
num_table_out	Количество таблиц, получаемых из векторной БД
graph_out_chunks	Количество чанков, получаемых из графового поиска

3.4 Перезапуск сервисов

Для перезапуска сервисов необходимо:

1. Перейти в настроенный GitLab.
2. Открыть соответствующий репозиторий.
3. Перейти в необходимую ветку.
4. Открыть CI/CD → Pipelines
5. Перезапустить job ‘deploy’