

# ***АГриЧата – Ваше решение, когда чаты в кармане***

**Сбор, обработка, аналитика, статистика общения и подготовка для  
обучения нейросети и возможное предсказание поведения**

Шуруповёрт вместо отвёртки: собери свои беседы, чаты, переписки и исследуй их вместе с аналитикой и ИИ.





# Краткое содержание:

Вводная часть

Общая архитектура, основные процессы

Основные составляющие проекта

Реализованный модуль

Patroni

PostgreSQL

ETCD

Backup

Prometheus

Grafana

Схема БД

Основные функции и возможности

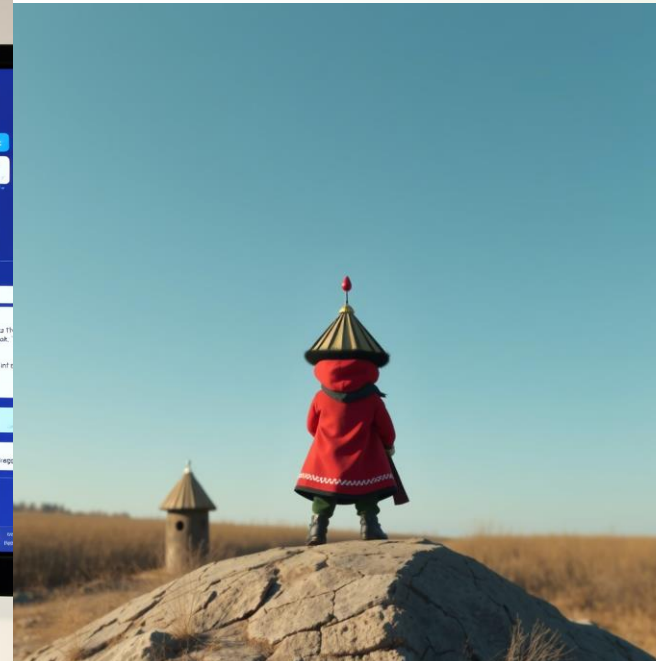
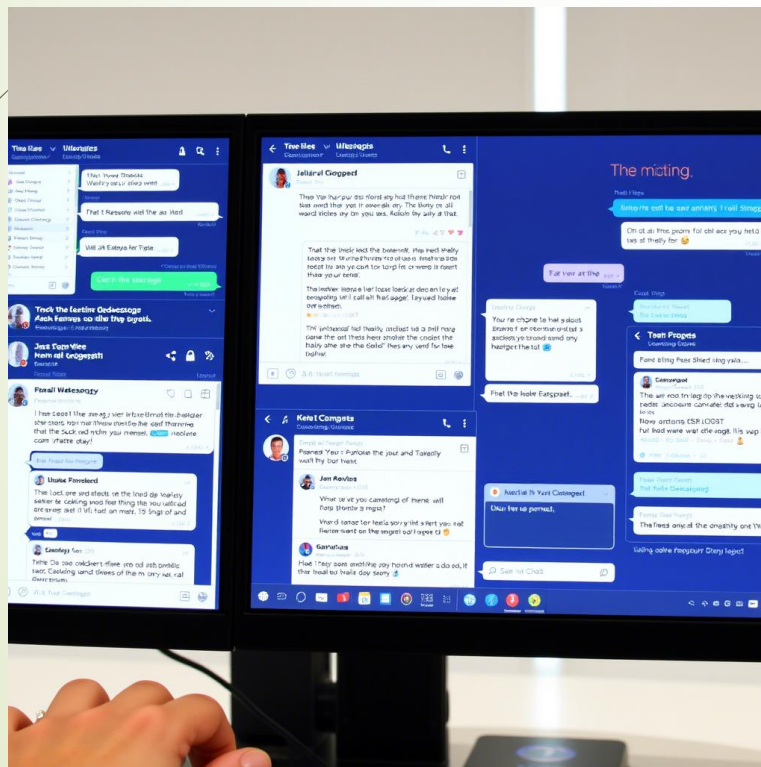
Перспективы развития, гипотезы и возможные решения

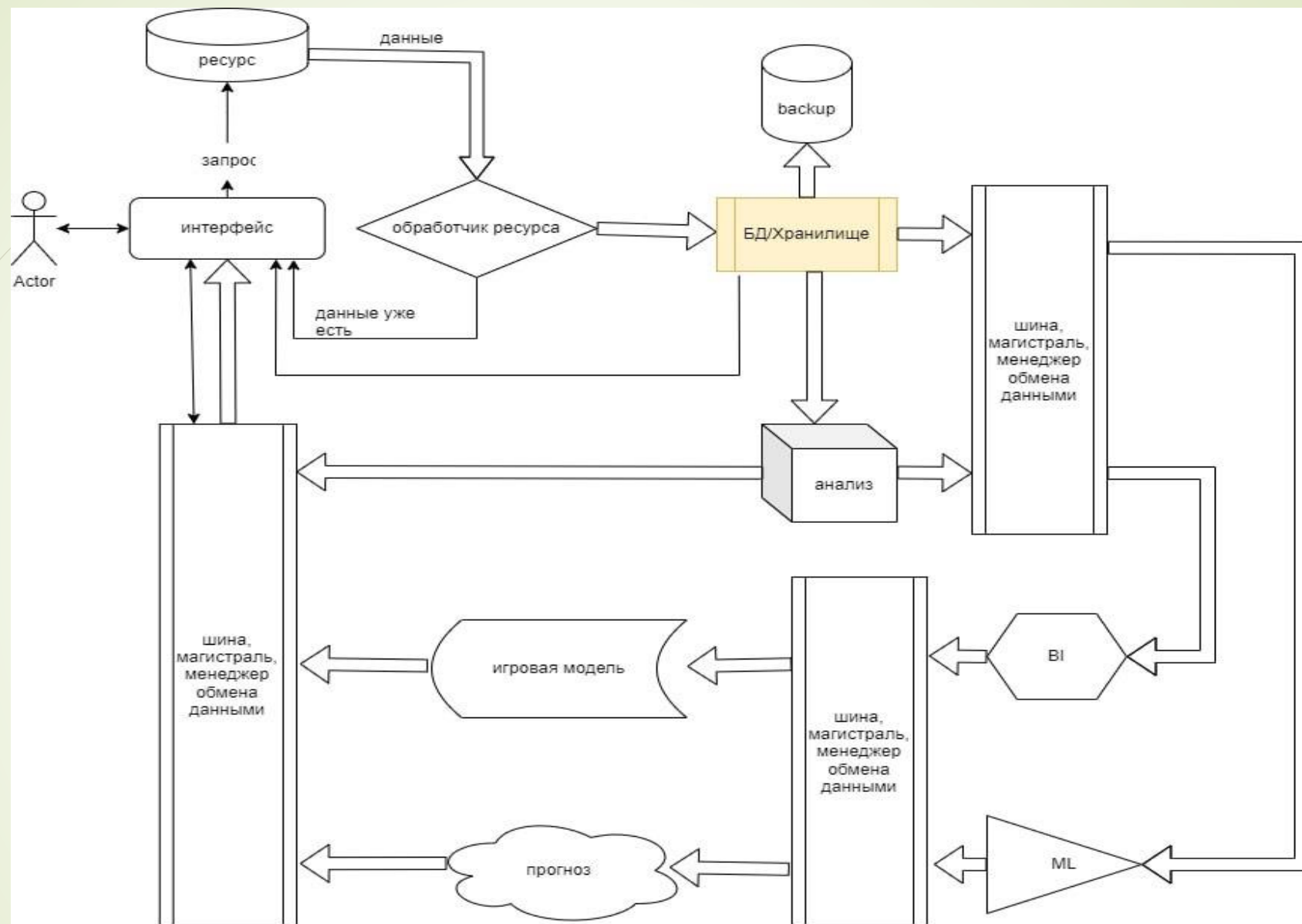
Рефлексия

# Вводная

Сегодня мы окружены тысячами чатов, сообщений, переписок, бесед. Точно знаем, что в этом ворохе повседневных деловых и не очень полезных бесед, под серостью рабочих будней скрыто золото информации, алмазы решений, изумруды алгоритмов. Надо всего лишь отыскать, раскопать полезное. Много ли времени есть у нас на это?

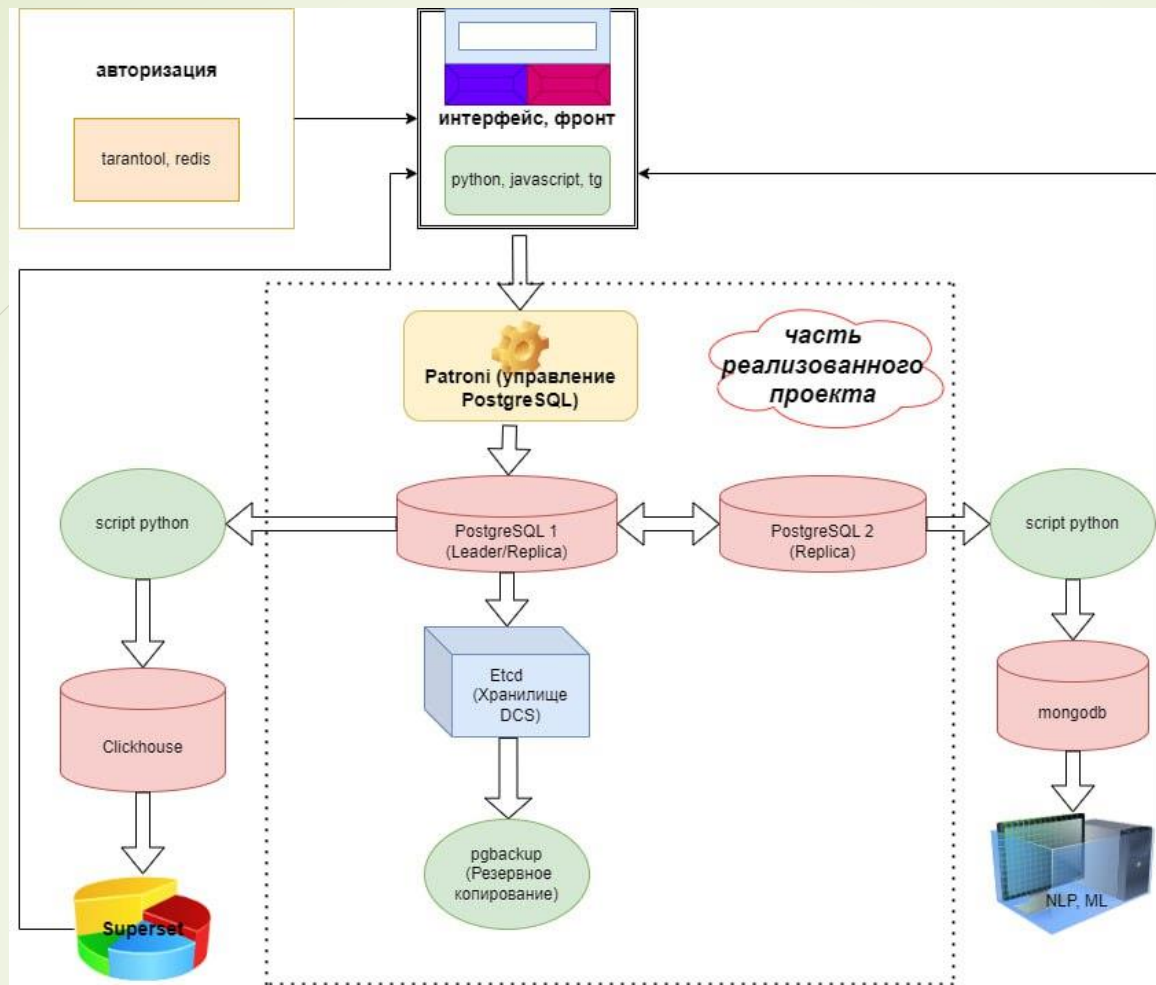
С помощью инструмента «АГриЧата» Вы можете увереннее ориентироваться в мультичатах и мегапереписках, выделяя для себя главное и отсеивая ненужное, ускоряя работу в несколько раз.





## Общая архитектура, основные процессы

Прототип прототипа



## Основные составляющие проекта, модуль загрузки



Patroni



Prometheus



Grafana





## Реализованный модуль

High-Availability PostgreSQL кластер с Patroni

### Архитектура решения

[Docker Host]

- etcd (хранилище конфигурации)
- Patroni 1 (PostgreSQL Master)
- Patroni 2 (PostgreSQL Replica)
- pgbackup (сервис бэкапов)

### Компоненты:

etcd: Координация кластера

Patroni: Управление PostgreSQL (+авто-фейловер)

pgbackup: Ежедневные бэкапы

## Настройка Patroni

scope: my\_cluster

restapi:

listen: 0.0.0.0:8008

etcd:

host: etcd:2379

postgresql:

pg\_hba:

- host all all 0.0.0.0/0 md5

## Преимущества:

Автоматическое переключение при сбоях

Централизованное управление через etcd

## Система бэкапов

Схема работы:

Скрипт backup-script.sh запускается по cron

pg\_dump создает бэкап

Файлы сохраняются в ./backups

## Пример команды:

pg\_dump -h patroni1 -p 5432 -U postgres -F c -f /backups/backup\_\$(date +"%Y%m%d").sql



## Docker-реализация

services:

patroni1:

image: registry.opensource.zalan.do/acid/spilo-14:2.1-p7

environment:

PATRONI\_POSTGRESQL\_DATA\_DIR: /var/lib/postgresql/data/pgdata

pgbackup:

image: postgres:14

volumes:

- ./backups:/backups

## Преимущества:

Изоляция сервисов

Простота развертывания

## Проверка работы

### Команды для мониторинга:

# Статус кластера

curl http://localhost:8008

# Проверка репликации

docker exec -it patroni1 psql -c "SELECT \* FROM pg\_stat\_replication;"

# Просмотр бэкапов

ls -lh ./backups





## Дополнительные возможности

### Расширения системы:

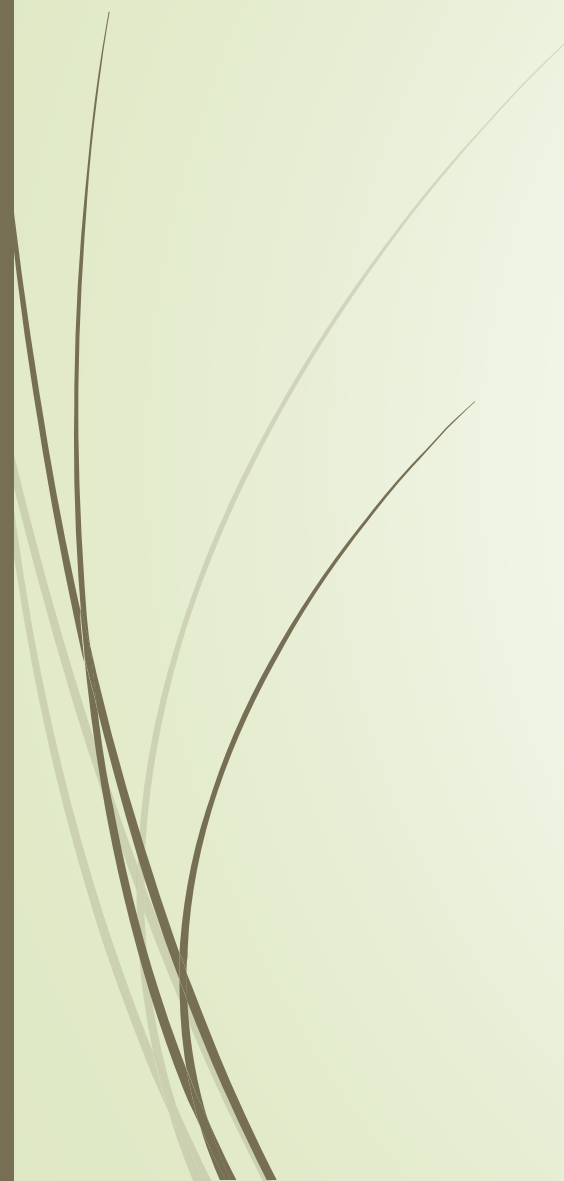
- Добавьте мониторинг (Prometheus + Grafana)
- Настройте алертинг при сбоях
- Интегрируйте с S3 (для хранения бэкапов)

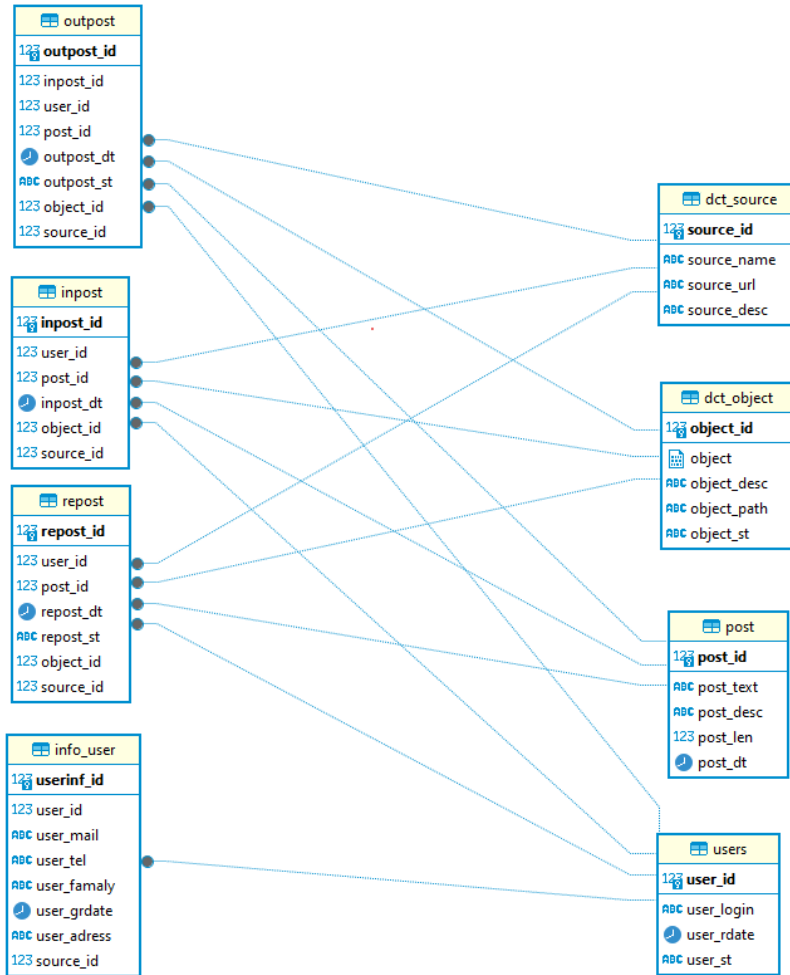
### Статистика:

- Доступность 99.99%
- Время восстановления < 30 сек

## Итоги

### Что достигнуто:

- Отказоустойчивый PostgreSQL-кластер
  - Автоматические бэкапы
  - Простое управление через Docker
- 



33

--создание информационной схемы

34

--DROP schema IF EXISTS info;

35

CREATE SCHEMA info AUTHORIZATION user\_info;

36

--создание целевой схемы

37

DROP schema IF EXISTS calc;

38

CREATE SCHEMA calc AUTHORIZATION user\_admin;

39

--создание целевой схемы

40

DROP schema IF EXISTS sapu;

41

CREATE SCHEMA sapu AUTHORIZATION user\_sapu;

42

--создание схемы расширений

43

CREATE SCHEMA extensions;

44

45

46

47

--создание таблицы в нужном нам табличном пространстве

48

DROP TABLE IF EXISTS info.info\_users ;

49

CREATE TABLE info.info\_users(users\_id serial PRIMARY KEY, users\_text jsonb) ;--TABLESPACE info\_space;

50

51

--создание последовательности для таблицы user

52

CREATE SEQUENCE sapu.seq\_user\_id

53

START WITH 1

54

INCREMENT BY 1

55

NO MINVALUE

56

NO MAXVALUE

57

CACHE 1;

58

--создание таблицы пользователей

59

CREATE TABLE sapu.users (

60

user\_id bigint DEFAULT nextval('sapu.seq\_user\_id'::regclass) NOT NULL PRIMARY key, -- уникальный идентификатор

61

user\_login character varying(50) NOT NULL, -- логин

62

user\_rdate timestamp without time zone DEFAULT now() NOT NULL, -- дата регистрации

63

user\_st varchar(50) DEFAULT 'ACTIVE' NOT NULL -- статус пользователя

64

);

65

66

--создание последовательности для идентификатора таблиц информации о пользователях

67

CREATE SEQUENCE sapu.seq\_userinf\_id

68

START WITH 1

69

INCREMENT BY 1

70

NO MINVALUE

71

NO MAXVALUE

72

CACHE 1;

73

--создание таблицы информации о пользователях

74

CREATE TABLE sapu.info\_user(

75

\*\*\*Сохраняет данные в базу данных.\*\*\*

76

connection = psycopg2.connect(\*\*DB\_CONFIG)

77

cursor = connection.cursor()

78

insert\_query = """

79

call sapu.create\_inpost(%s, %s, 1, %s::timestamp);

80

"""

81

for message in messages:

82

cursor.execute(insert\_query, (message['author'], message['text'],message

83

connection.commit()

84

cursor.close()

85

connection.close()

86

if \_\_name\_\_ == '\_\_main\_\_':

87

# Путь к папке с HTML-файлами

88

folder\_path = r'C:\Python\messages'

89

# Получаем список всех HTML-файлов в папке

90

html\_files = [f for f in os.listdir(folder\_path) if f.endswith('.html')]

91

total\_messages\_saved = 0

92

for html\_file in html\_files:

93

file\_path = os.path.join(folder\_path, html\_file)

94

# Парсим файл

95

messages = parse\_html(file\_path)

96

# Сохраняем данные в базу данных

97

save\_to\_db(messages)

98

# Подсчитываем общее количество сохраненных сообщений

99

total\_messages\_saved += len(messages)

100

print(f'Файл {html\_file} обработан. Сохранено {len(messages)} сообщений.

101

...

Язык: [Русский]

PostgreSQL » postgres » otus » sapu » Выбрать: users

Adminer 5.0.6 5.1.0

DB: [otus]

Схема: [sapu]

SQL-запрос

Импорт

Экспорт

Создать таблицу

выбрать dct\_object

выбрать dct\_source

выбрать info\_user

выбрать inpost

выбрать outpost

выбрать post

выбрать repost

выбрать users

Выбрать: users

Выбрать

Показать структуру

Изменить таблицу

Новая запись

Выбрать

Поиск

Сортировать

Лимит

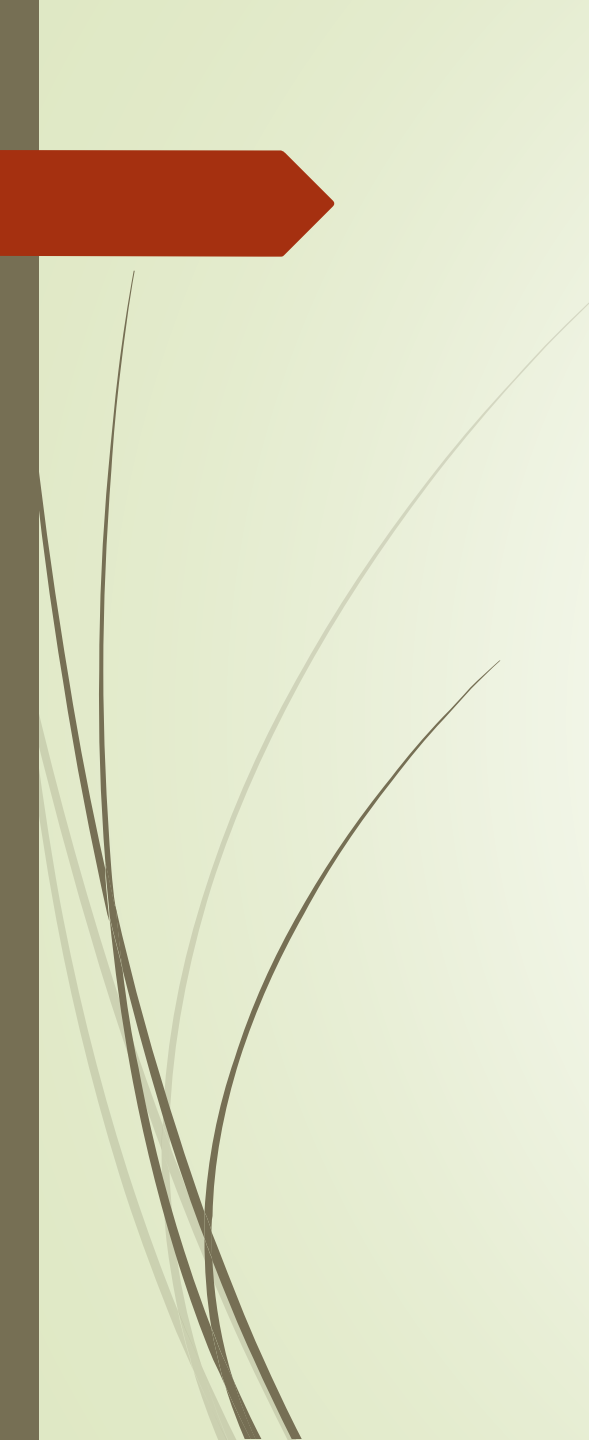
Длина текста

Действие

SELECT \* FROM "users" LIMIT 50 OFFSET 750 (0.001 s)

Редактировать

	user_id	user_login	user_rdate	user_st
<input type="checkbox"/> Изменить	751	Natasha	2025-03-26 08:44:07.544089	ACTIVE
<input type="checkbox"/> редактировать	752	Vitali Efimov	2025-03-26 08:44:07.544089	ACTIVE
<input type="checkbox"/> редактировать	753	Rumila Sinelnikova	2025-03-26 08:44:07.544089	ACTIVE
<input type="checkbox"/> редактировать	754	R.	2025-03-26 08:44:07.544089	ACTIVE
<input type="checkbox"/> редактировать	755	Святослав Смирнов	2025-03-26 08:44:07.544089	ACTIVE
<input type="checkbox"/> редактировать	756	Roman Solomatn	2025-03-26 08:44:07.544089	ACTIVE
<input type="checkbox"/> редактировать	757	Alexander Polyakov	2025-03-26 08:44:07.544089	ACTIVE




Данный программный комплекс возможно создать на микросервисной архитектуре, что по сути и преследуется в данном проекте.

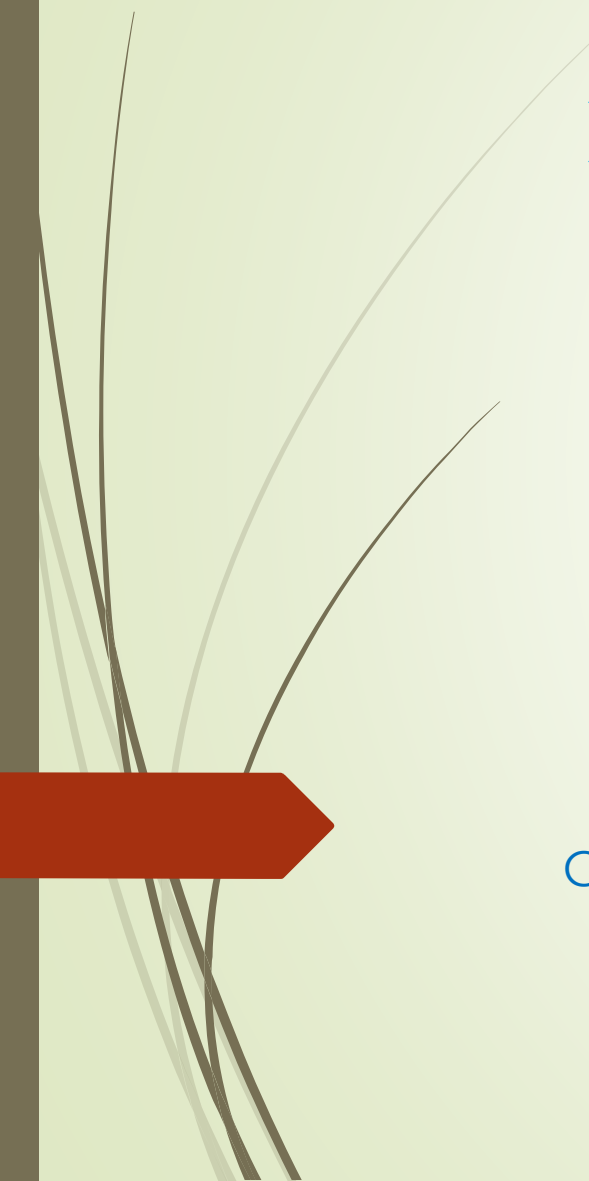
В дальнейшем планируется рассмотреть возможность использования Kubernetes, подключение средств мониторинга работы (добавлено в текущем, но многие тесткейсы не пройдены), добавление скриптов для взаимодействия (API) между модулями (сервисами) (частично реализовано, но не представлено в проекте), обучение в коробочном варианте AI-моделей (не представлено в рамках данного проекта), разработке модели анализа и прогнозирования поведения собеседника, сбора и предоставления статистики (прорабатывается техзадание), гипотетически получать так называемый «почерк автора».

К примеру, в процессе создания проекта возникло желание местами использовать MySQL (есть преимущества), как бы я не любил PostgreSQL (но возможная работа с json и массивами отодвинула MySQL).

Интересные решения приходили в голову с использованием Cassandra и MongoDB, ClickHouse как таковой включен в архитектуру, но можно было бы «поиграться», GreenPlum хотелось применить, но на другом уровне.



Проект является прототипом, но даже на текущем уровне снимает часть затрат, во всяком случае в моём списке задач. Курс позволил изучить помимо тем указанных в списке часть возможностей докер, с которым был всего лишь знаком на словах, bash-скрипт, Linux, Markdown и некоторые возможности инструментов программных средств, как то Patroni, ETCD, Prometheus, Grafana (до этого использовал на коленке написанные скрипты для локальных задач), вспомнить Python, задуматься о Javascript и Kubernetes или аналог.



# Это не конец, это только начало Пути!

Спасибо лекторам за курс!