Разработка системы анализа производительности СУБД и MPP-кластеров

Python Developer. Basic

Меня хорошо видно & слышно?

Защита проекта
Тема: Разработка системы анализа производительности СУБД и MPP-кластеров

Игорь Щербаков

Разработчик баз данных

План защиты

- Цель и задачи проекта
- Какие технологии использовались
- Что получилось
- Выводы
- Вопросы и рекомендации

Цель и задачи проекта

Цель проекта: создать систему DBBS – Data Base Benchmark System – для анализа производительности запросов СУБД и MPP-кластеров

1. Разработать вспомогательную БД для хранения:
   - запросов,
   - групп запросов,
   - результатов выполнения запросов
2. Создать утилиту для работы с системой DBBS через командную строку
3. На основе фреймворка Django создать приложение для администрирования DBBS
4. Обеспечить представление замеров производительности
5. Выполнить примеры замеров времени выполнения запросов

Какие технологии использовались

1. Python
2. Библиотека psycopg2 для работы с БД
3. Фреймворк Django
4. СУБД Postgres
5. MPP-кластер ArenadataDB

MPP - Massively Parallel Processing - массивно-параллельная обработка.
Такие кластеры используются для построения хранилищ данных.

Когда может потребоваться анализ производительности запросов?

- При изменении настроек производительности СУБД
- При переносе информационной системы на другой сервер
- Если БД или кластер находится в облаке, и администраторы облака делают настройки, влияющие на производительность
- При выборе СУБД или MPP-кластера

Запросы, которые можно использовать при анализе:

- характерные для конкретной информационной системы
- использующие разные возможности построителя запросов СУБД

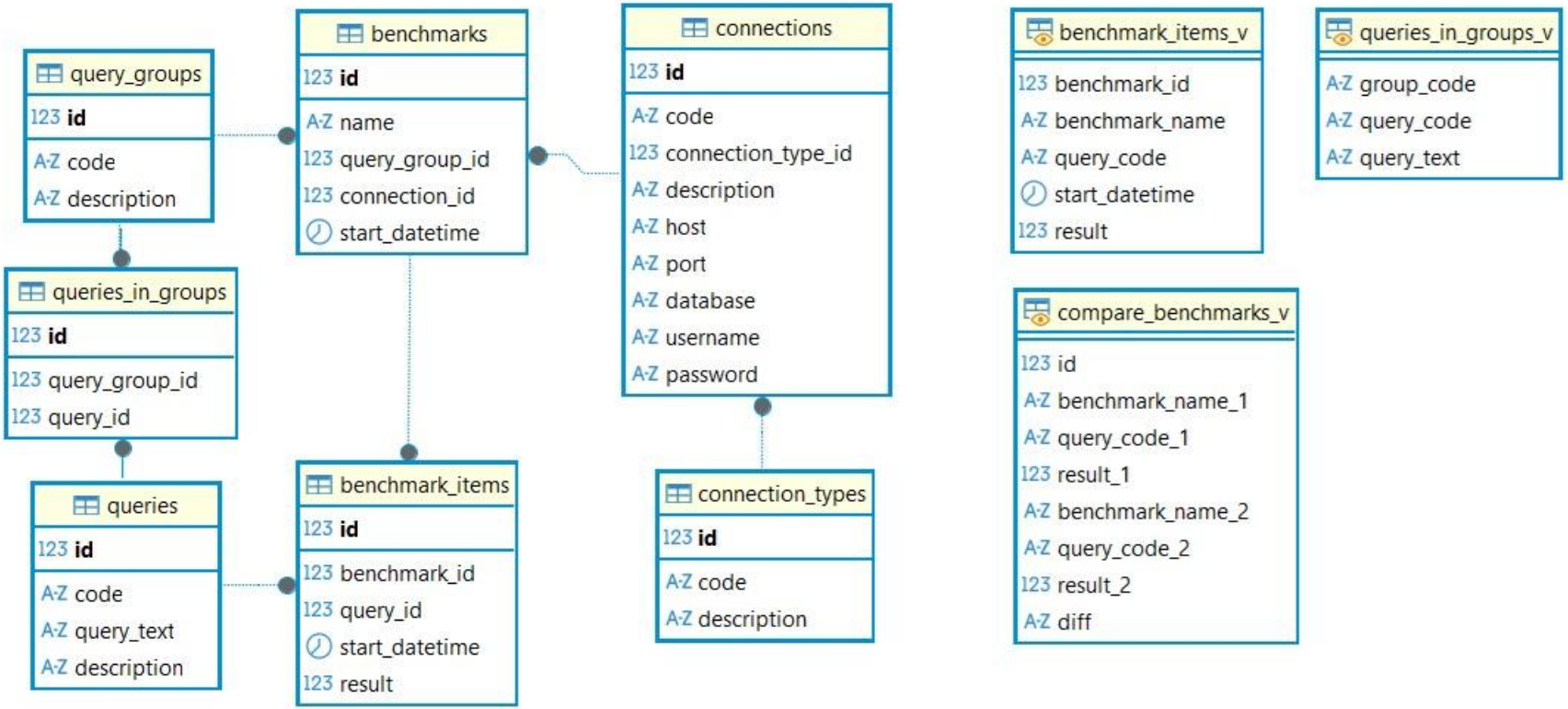
План запроса и время выполнения

Команда explain analyze дает точное значение времени выполнения запроса

```
explain analyze
select * from tickets t join ticket_flights tf on tf.ticket_no = t.ticket_no
where t.ticket_no in ('0005432312163','0005432312164')
```

```
Nested Loop  (cost=0.99..46.10 rows=6 width=136) (actual time=0.032..0.051 rows=8 loops=1)
  ->  Index Scan using tickets_pkey on tickets t  (cost=0.43..12.90 rows=2 width=104) (actual time=0.014..0.020 rows=2 loops=1)
        Index Cond: (ticket_no = ANY ('{0005432312163,0005432312164}'::bpchar[]))
  ->  Index Scan using ticket_flights_pkey on ticket_flights tf  (cost=0.56..16.57 rows=3 width=32) (actual time=0.010..0.013 rows=4 loops=2)
        Index Cond: (ticket_no = t.ticket_no)
Planning Time: 0.213 ms
Execution Time: 0.077 ms
```

Схема вспомогательной БД



Утилита для работы из командной строки

python **dbbs_cli**.py <команда> <параметр_1> <параметр_2> <параметр_3>
Команды:
  -b - выполнить замер производительности,
    <параметр_1> - код группы запросов
    <параметр_2> - наименование замера производительности
    <параметр_3> - код соединения со вспомогательной БД
  -r - вывести результаты замера производительности,
    <параметр_1> - наименование замера производительности
  -c - сравнить два замера производительности
    <параметр_1> - наименование замера производительности № 1
    <параметр_2> - наименование замера производительности № 2
  -h - вывести помощь по этой утилите
Например:
  python dbbs_cli.py -b gr1 benchmark01 pg1
  python dbbs_cli.py -r benchmark01
  python dbbs_cli.py -c benchmark01 benchmark02
  python dbbs_cli.py -h

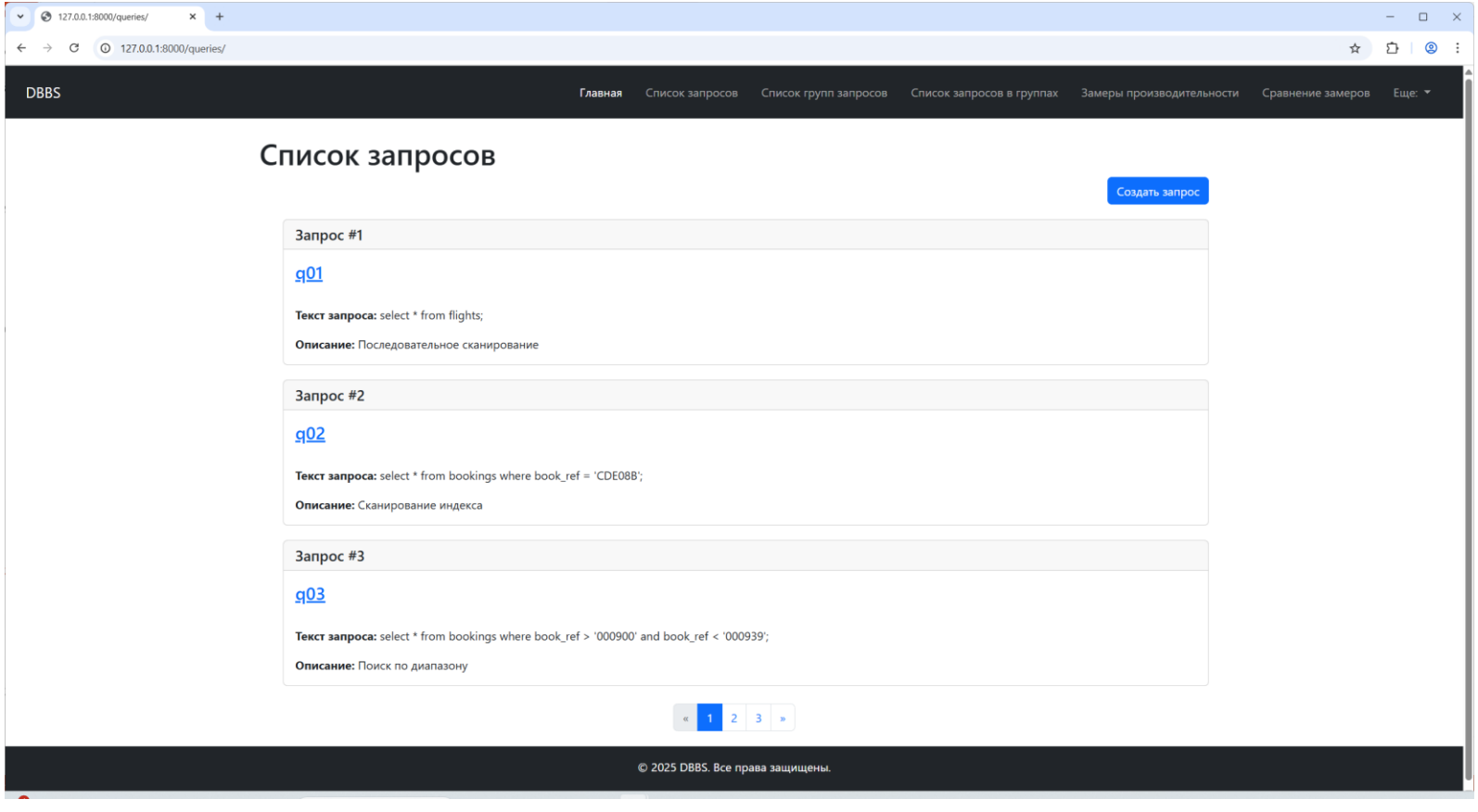
Замер производительности

Код запроса	Дата и время	Результат, мс
q01	2025-07-24 15:51:50.652868	57.12
q02	2025-07-24 15:51:50.658750	0.77
q03	2025-07-24 15:51:50.660322	0.167
q04	2025-07-24 15:51:50.663698	0.026
q05	2025-07-24 15:51:50.850891	184.718
q06	2025-07-24 15:51:51.505990	653.276
q07	2025-07-24 15:51:51.514461	2.708
q08	2025-07-24 15:51:51.519823	0.028
q09	2025-07-24 15:51:51.549912	27.131
q10	2025-07-24 15:51:51.555275	0.815
q11	2025-07-24 15:51:51.823066	264.585
q12	2025-07-24 15:51:51.909996	83.662
q13	2025-07-24 15:51:51.941785	27.899
q14	2025-07-24 15:51:51.945607	0.011
q15	2025-07-24 15:51:54.135687	2189.005

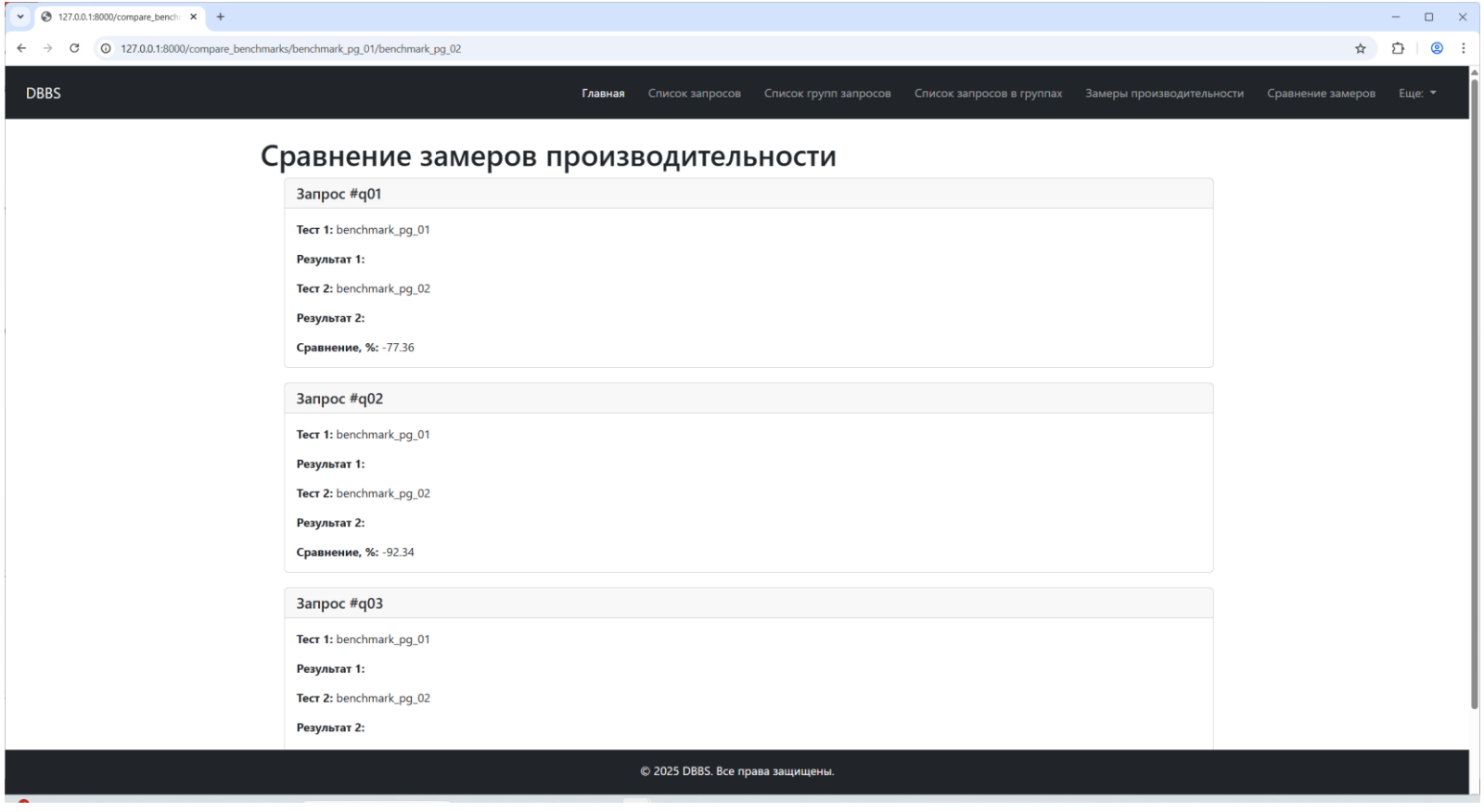
Сравнение замеров производительности

id	benchmark_name_1	query_code_1	result_1	benchmark_name_2	query_code_2	result_2	diff
7	benchmark_pg_01	q01	57,12	benchmark_pg_02	q01	12,932	-77.36
8	benchmark_pg_01	q02	0,77	benchmark_pg_02	q02	0,059	-92.34
9	benchmark_pg_01	q03	0,167	benchmark_pg_02	q03	0,014	-91.62
10	benchmark_pg_01	q04	0,026	benchmark_pg_02	q04	0,021	-19.23
11	benchmark_pg_01	q05	184,718	benchmark_pg_02	q05	5,936	-96.79
12	benchmark_pg_01	q06	653,276	benchmark_pg_02	q06	24,026	-96.32
13	benchmark_pg_01	q07	2,708	benchmark_pg_02	q07	0,884	-67.36
14	benchmark_pg_01	q08	0,028	benchmark_pg_02	q08	0,015	-46.43
15	benchmark_pg_01	q09	27,131	benchmark_pg_02	q09	12,503	-53.92
16	benchmark_pg_01	q10	0,815	benchmark_pg_02	q10	0,025	-96.93
17	benchmark_pg_01	q11	264,585	benchmark_pg_02	q11	88,731	-66.46
18	benchmark_pg_01	q12	83,662	benchmark_pg_02	q12	66,349	-20.69
19	benchmark_pg_01	q13	27,899	benchmark_pg_02	q13	34,522	23.74
20	benchmark_pg_01	q14	0,011	benchmark_pg_02	q14	0,01	-9.09
21	benchmark_pg_01	q15	2 189,005	benchmark_pg_02	q15	2 111,416	-3.54
22	benchmark_pg_01	q16	2 871,329	benchmark_pg_02	q16	2 902,767	1.09
23	benchmark_pg_01	q17	0,66	benchmark_pg_02	q17	0,204	-69.09
24	benchmark_pg_01	q18	2 558,942	benchmark_pg_02	q18	2 183,507	-14.67
25	benchmark_pg_01	q19	47 818,243	benchmark_pg_02	q19	47 203,905	-1.28
26	benchmark_pg_01	q20	1 427,958	benchmark_pg_02	q20	1 455,418	1.92
27	benchmark_pg_01	q21	0,646	benchmark_pg_02	q21	0,069	-89.32

Приложение на Django



Сравнение замеров производительности



Что получилось

Первая версия системы DBBS разработана полностью:

1. Разработана БД для хранения запросов, групп, результатов выполнения замеров производительности
2. Создана утилита для работы с системой DBBS через командную строку
3. На фреймворке Django разработано приложение для администрирования DBBS
4. Обеспечено представление замеров производительности
5. Выполнены примеры замеров производительности

Выводы

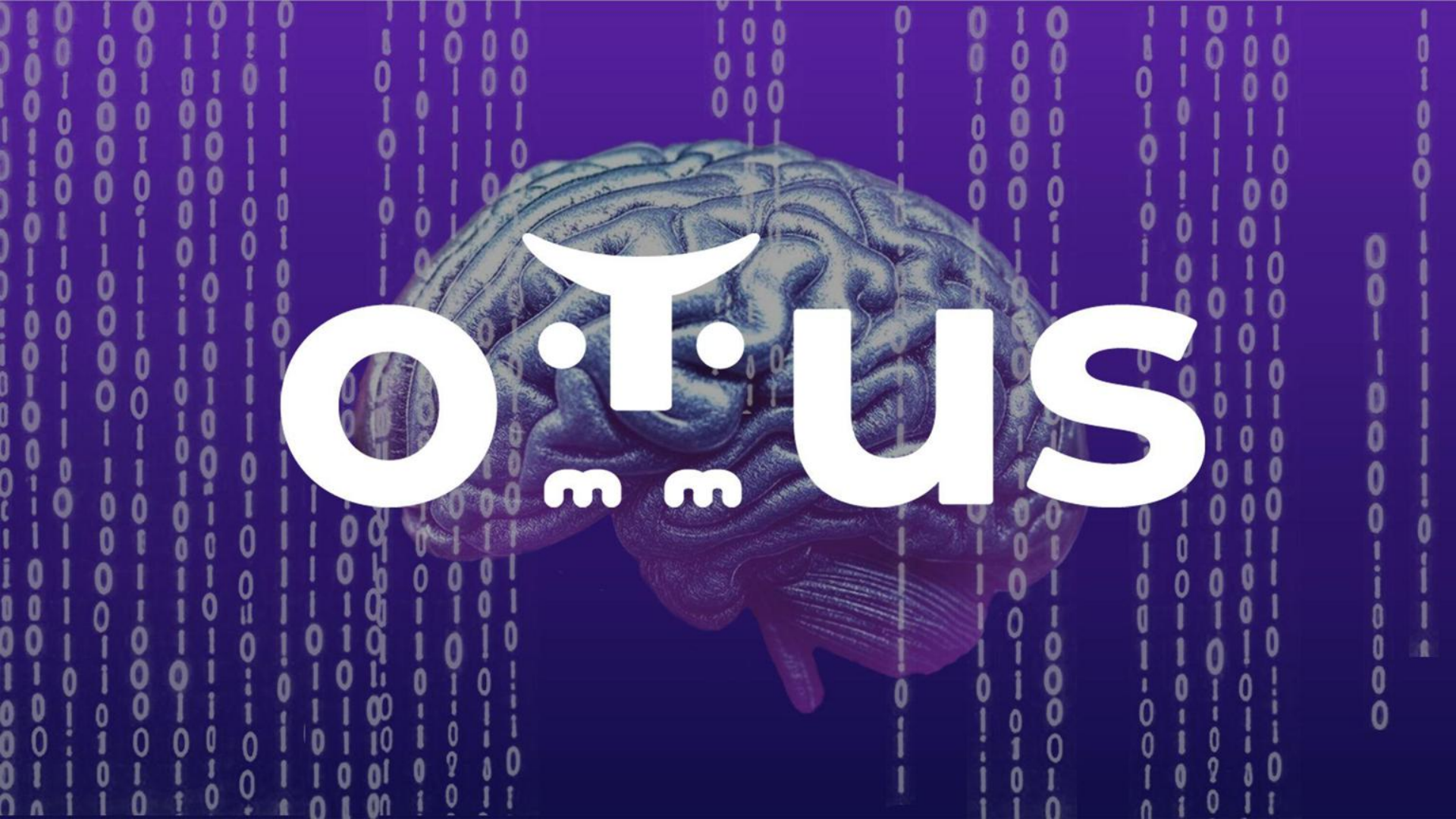
1. Поставленные цели достигнуты
2. Разработанная система будет использована в работе
3. Возможно дальнейшее совершенствование разработанной системы

Вопросы и рекомендации

\+ если есть вопросы

— если вопросов нет

Спасибо за внимание!


OTUS