

● On-Premise AI · Исследование кода

AI Insider

AI-система, которая знает о вашем коде больше, чем ваша команда.

Внутри вашего периметра. Под полным контролем.



100% on-premise · Без облака

120K+

ИССЛЕДОВАНИЙ

200+

РЕПОЗИТОРИЕВ

20

ЯЗЫКОВ В
ОБУЧЕНИИ

<1ч

РАЗВЁРТЫВАНИЕ

Почему знание кода теряется

01 —

Высокий bus factor

Ключевые разработчики уходят — знания теряются. Новые сотрудники месяцами разбираются в legacy.

02 —

Legacy без документации

Критически важный код работает годами, но никто не помнит почему. Каждое изменение — лотерея.

03 —

Облачный AI — запрещён

ИБ, ФСТЭК, Ф3-152, GDPR запрещают отправлять исходный код во внешние сервисы.

04 —

Документация дорожает

Поддерживать документацию в актуальном состоянии — дорого. С ростом кодовой базы стоимость только растёт.

Не просто чатбот. Специалист по вашему коду.



Глубокое исследование

Навигирует по файлам, читает исходники, строит полную картину зависимостей. Не просто отвечает — исследует.



Предобученная модель

120К+ реальных исследований. Понимает паттерны, архитектуры и идиомы. Понимает любые языки программирования.



Полностью on-premise

Работает на вашем оборудовании. Данные не уходят наружу. Ф3-152, GDPR, ФСТЭК.



Для всей команды

Разработчики, аналитики, тестировщики, менеджеры — на естественном языке.



AI Insider

Как реализована авторизация?



Exploring workspace...



Reading auth files...



Analyzing JWT logic...



Готово • уверенность: high

4 шага от вопроса до ответа

01

Задайте вопрос

На естественном языке: «Где авторизация?», «Как работают платежи?», «Какие зависимости у модуля X?»

02

AI исследует код

Навигирует по файлам, читает исходный код, анализирует зависимости — как опытный разработчик.

03

Получите ответ

Структурированный ответ с точными ссылками на файлы и строки кода. С оценкой уверенности.

04

Проверьте источники

Блок источников — конкретные файлы и строки, на которых основан ответ. Полная прозрачность.

Пример: Apache Superset

ШАГИ ИССЛЕДОВАНИЯ



Exploring workspace structure...



Reading cache config files...



Analyzing Redis/Memcached...



Answer ready · confidence: high

ИСТОЧНИКИ

`superset/config.py:1101-1140`

CACHE_DEFAULT_TIMEOUT, CACHE_CONFIG

`docs/configuration/cache.mdx:1-243`

Redis, Memcached backends, TTL

`superset/common/query_context.py:101-119`

TTL chain: custom → chart → db

Apache Superset — В ДЕЙСТВИИ

01 — ВОПРОС



Как Superset кеширует результаты запросов — какие backend (Redis, Memcached) и TTL стратегии используются?

02 — AI ИССЛЕДУЕТ КОД



Let me start by exploring the workspace structure to find Superset-related files.



Searching for cache configuration files in the Superset source code.



Reading the main Superset configuration file to find cache backend settings.



Reading the cache manager to understand how backends are initialized and used.



Reading the cache documentation to understand Redis/Memcached backends and TTL strategies.



Reading the rest of the cache documentation for more details on backends and TTL strategies.



Reading the async queries documentation to understand RESULTS_BACKEND configuration.



Checking the Docker development configuration for practical Redis cache examples.



Reading the cache utilities to understand chart-level TTL override strategies.



Checking how chart/dataset/database-level TTL overrides are chained in query processing.



Checking the query context get_cache_timeout to understand the chart/dataset/database override chain.



I have gathered all the information needed to provide a comprehensive answer.

03 — РАЗВЁРНУТЫЙ ОТВЕТ

Ответ high

Кэширование запросов в Superset

Superset использует библиотеку `Flask-Caching` для кэширования. Архитектура кэша состоит из нескольких независимых конфигураций.

Поддерживаемые Backend'ы

1. Redis (рекомендован для production)

Два варианта:

- `RedisCache` — стандартный Redis:

```
CACHE_CONFIG = {
    'CACHE_TYPE': 'RedisCache',
    'CACHE_DEFAULT_TIMEOUT': 300,
    'CACHE_KEY_PREFIX': 'superset_',
    'CACHE_REDIS_HOST': 'localhost',
    'CACHE_REDIS_PORT': 6379,
    'CACHE_REDIS_DB': 1,
}
```

- `RedisSentinelCache` — Redis с Sentinel для HA (высокой доступности)

Для `RESULTS_BACKEND` (SQL Lab async) используется объект ``cachelib``:

```
from flask_caching.backends.rediscache import RedisCache
RESULTS_BACKEND = RedisCache(host='localhost', port=6379, key_prefix='superset_res
```

`SIGNAL_CACHE_CONFIG` требует *исключительно Redis или Valkey* — используются Redis-специфичные функции (`pub/sub`, ``SET NX EX``) для distributed locking и event notifications.

2. Memcached

Поддерживается через `Flask-Caching`. Рекомендуется библиотека ``pylibmc`` (не ``python-memcached``), так как последняя не обрабатывает бинарные данные корректно). Также поддерживается в `RESULTS_BACKEND`.

3. SupersetMetastoreCache (встроенный)

Хранит данные в таблице key-value метаданных Superset. Используется по умолчанию для `FILTER_STATE` и `EXPLORE_FORM_DATA`:

```
DATA_CACHE_CONFIG = {
    "CACHE_TYPE": "SupersetMetastoreCache",
    "CACHE_KEY_PREFIX": "superset_results",
    "CACHE_DEFAULT_TIMEOUT": 86400,
}
```

4. NullCache

По умолчанию для `CACHE_CONFIG` и `DATA_CACHE_CONFIG` — кэширование отключено.

5. Прочие

- `FileSystemCache` — для разработки (dev docker config)
- `S3Cache` — для `THUMBNAIL_CACHE_CONFIG` и `RESULTS_BACKEND`
- `SimpleCache` (in-memory)

Конфигурации кэша

Конфигурация	Назначение	TTL по умолчанию
<code>`CACHE_CONFIG`</code>	Метаданные Superset	NullCache (нет TTL)
<code>`DATA_CACHE_CONFIG`</code>	Данные чартов / запросов к датасетам	NullCache (нет TTL)
<code>`FILTER_STATE_CACHE_CONFIG`</code>	Состояние фильтров дашборда	90 дней
<code>`EXPLORE_FORM_DATA_CACHE_CONFIG`</code>	Данные формы Explore	7 дней
<code>`THUMBNAIL_CACHE_CONFIG`</code>	Скриншоты/превью	7 дней
<code>`RESULTS_BACKEND`</code>	Результаты async SQL Lab	без TTL (объект cachelib)
<code>`SIGNAL_CACHE_CONFIG`</code>	Distributed locks + pub/sub	None (опционально)

TTL Стратегии

Глобальный TTL

```
CACHE_DEFAULT_TIMEOUT = int(timedelta(days=1).total_seconds()) # 86400 сек
```

Приоритетная цепочка TTL для данных чартов

Реализована в ``superset/common/query_context.py`` и ``query_context_processor.py``:

04 — ИСТОЧНИКИ

Источники

superset/config.py:1101–1140

Определения всех основных конфигураций кэша: `CACHE_DEFAULT_TIMEOUT` (1 день), `CACHE_CONFIG`, `DATA_CACHE_CONFIG`, `FILTER_STATE_CACHE_CONFIG` (90 дней), `EXPLORE_FORM_DATA_CACHE_CONFIG` (7 дней), `RESULTS_BACKEND`, `SIGNAL_CACHE_CONFIG`

docs/docs/configuration/cache.mdx:1–243

Документация по всем backend'ам (Redis, Memcached, SupersetMetastoreCache), примеры конфигурации, TTL стратегии

docs/docs/configuration/async-queries-celery.mdx:55–77

`RESULTS_BACKEND` с примерами Redis и S3, использование `MessagePack` для сериализации

superset/common/query_context.py:101–119

Приоритетная цепочка TTL: `custom_timeout` → `chart` → `datasource` → `database`

superset/common/query_context_processor.py:385–394

Полная цепочка TTL: `query_context` → `DATA_CACHE_CONFIG` → `CACHE_DEFAULT_TIMEOUT`

superset/utils/cache.py:56–100

`set_and_log_cache`: обработка `CACHE_DISABLED_TIMEOUT` (-1) для отключения кэширования

docker/pythonpath_dev/superset_config.py:64–91

Практический пример конфигурации Redis: `RedisCache` с TTL 300 сек, `FileSystemCache` для `RESULTS_BACKEND` в dev

Уверенность: Все утверждения подтверждены непосредственным чтением исходного кода и документации. Прочитаны все ключевые файлы: `config.py`, `cache.py`, `query_context.py`, `query_context_processor.py`, `cache.mdx`, `async-queries-celery.mdx` и `docker` конфиг.

— ВОЗМОЖНОСТИ

Всё для enterprise



Две модели

4B — ноутбук.

14B — сервер с GPU.



За час

Docker или нативная
установка. Полная
поддержка.



Любые языки

Понимает все. Обучена
на 20 ключевых.



Изоляция

Без интернета.
Соответствие ФСТЭК,
Ф3-152.



Источники

Файлы и строки кода в
каждом ответе.



Безлимит

Все пользователи
компании без
ограничений.

Понимание, не генерация

Copilot и Cursor создают новый код. AI Insider объясняет существующий. Разные инструменты — разные задачи.

	AI Insider	Copilot	Cursor	Cursor
On-premise	✓	✗	✗	
Исследование кода	✓	✗	✗	
Предобученная модель	✓	✗	✗	
Ссылки на строки	✓	✗	✗	
Без интернета	✓	✗	✗	
Безлим. лицензии	✓	✗	✗	

Что получает команда после внедрения

 **+20–35%** скорость разработки

Разработчики мгновенно понимают возможности внутренних систем — меньше времени на исследование, больше на создание.

↗ [Google RCT, arxiv 2024](#)

 **в 1,5–2×** быстрее онбординг

Новые разработчики, аналитики и тестировщики выходят на продуктивность без месяцев погружения.

↗ [Cortex State of Dev Productivity 2024](#)

 **–40%** нецелевых трудозатрат

Поиск ответов, поддержка документации, описания — 63% команды тратит 30+ мин/день только на поиск.

↗ [Stack Overflow Dev Survey 2023](#)

 **Снижение bus factor**

Критическая экспертиза о системах перестаёт быть заложницей 1–2 ключевых людей. Знания о коде доступны всей команде в любой момент.



Качество анализа и постановки задач

Аналитики видят полную картину системы — более точные требования, меньше итераций и доработок на этапе разработки.



Качество тестирования и релизов

Тестировщики понимают что и как проверять на уровне кода — покрытие растёт, критические дефекты реже уходят в прод.

ОКУПАЕМОСТЬ

Расчёт для 50 человек

Разработчики, аналитики, тестировщики, DevOps.
Стоимость часа с накладными — 1 875 ₽.

СРЕДНЯЯ ЗРЕЛОСТЬ

736 000

₽ / мес экономии

при стандартных процессах и документации

НИЗКАЯ ЗРЕЛОСТЬ (LEGACY)

× 2,7
ROI

экономия превышает стоимость лицензии в 2,7× с
первого месяца

КОРПОРАТИВНАЯ ЛИЦЕНЗИЯ

~5 мес

окупаемость

при 277 778 ₽/мес и средней зрелости

ПОИСК ПО КОДУ

45 мин × 50 чел × −40% = 619 000 ₽/мес

ОНБОРДИНГ

8 новых/год × 3 → 2 мес = 67 000 ₽/мес

ДОКУМЕНТАЦИЯ

2 дня/мес × 50 чел × −40% = 50 000 ₽/мес

⚠ Расчёт основан на публичных исследованиях: Stack Overflow 2023, Cortex 2024, Google RCT arxiv 2024, IBM Research. Реальный эффект зависит от специфики команды.

— СТОИМОСТЬ

Прозрачные условия

КОРПОРАТИВНАЯ ЛИЦЕНЗИЯ

ВЫГОДНО

₽10 000 000

за 3 года · бессрочная

- ✓ Безлимитные лицензии
- ✓ Обе модели: 4В и 14В
- ✓ Поддержка 3 года
- ✓ Обновления платформы
- ✓ Помощь с развёртыванием
- ✓ Обучение команды

ПОМЕСЯЧНАЯ ОПЛАТА

₽400 000

в месяц · гибкий формат

- ✓ Безлимитные лицензии
- ✓ Обе модели: 4В и 14В
- ✓ Техническая поддержка
- ✓ Обновления платформы
- ✓ Помощь с развёртыванием
- ✓ Обучение команды



Корпоративная лицензия = **₽277 778/мес** вместо ₽400 000. Экономия 30% при горизонте 3 года.

— СЛЕДУЮЩИЙ ШАГ

ГОТОВЫ УВИДЕТЬ AI Insider в деле?

Проведём демонстрацию на ваших или open-source проектах — до принятия решения.

Запросить демо →

Открыть сайт

САЙТ

buff.systems/landings/ai-insider

ПРИМЕРЫ

5 000+ исследований