On-Premise AI · Глубокое 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 · Без внешних API

120K+	200+	20	<1ч
ИССЛЕДОВАНИЙ В ОБУЧЕНИИ	OPEN-SOURCE ПРОЕКТОВ	ЯЗЫКОВ В ОБУЧЕНИИ	ДО РАЗВЁРТЫВАНИЯ

ЗНАКОМЫЕ ПРОБЛЕМЫ

Почему знание кода теряется и дорожает

01 —

Высокий bus factor

Ключевые разработчики уходят — знания о коде теряются навсегда. Новые сотрудники месяцами разбираются в legacy вместо продуктивной работы.

02 —

Legacy без документации

Критически важный код работает десятилетиями, но никто не помнит — почему именно так. Каждое изменение — лотерея с непредсказуемыми последствиями.

03 —

Облачный AI — запрещён

Требования ИБ и регуляторов (ФСТЭК, ФСБ, ФЗ-152, GDPR) запрещают отправлять исходный код во внешние сервисы. Это интеллектуальная собственность.

04 —

Документация дорожает

Поддерживать документацию и инструкции в актуальном состоянии — дорого. И с ростом кодовой базы эта стоимость только увеличивается.

РЕШЕНИЕ

Не просто чатбот. Специалист по вашему коду.

Глубокое исследование

AI не просто отвечает — она навигирует по вашим файлам, читает исходники, строит полную картину зависимостей. Как опытный разработчик.

Предобученная модель

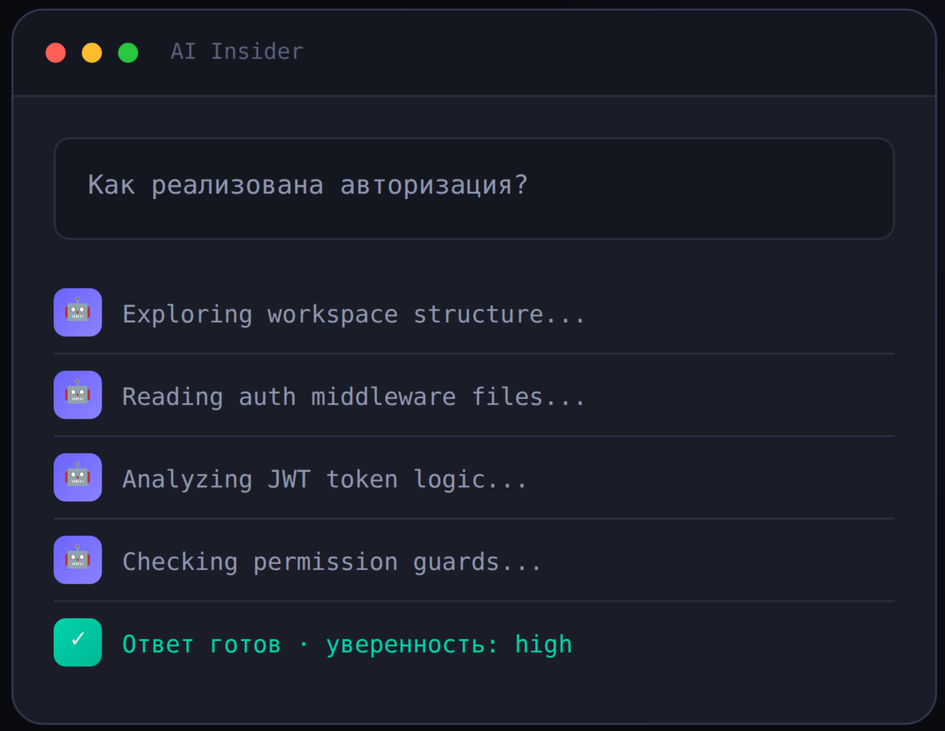
Обучена на 120K+ реальных исследованиях. Знает паттерны, архитектуры и идиомы. Понимает любые языки программирования.

Полностью on-premise

Работает на вашем оборудовании. Никакие данные не уходят наружу. Соответствие ФЗ-152, GDPR, ФСТЭК, ФСБ.

Для всей команды

Разработчики, аналитики, тестировщики, менеджеры — любой может задавать вопросы о коде на естественном языке.



КАК ЭТО РАБОТАЕТ

От вопроса до ответа за секунды

01

Задайте вопрос

На естественном языке: «Где реализована авторизация?», «Как работает обработка платежей?», «Какие зависимости у модуля X?»

02

AI исследует

Модель навигирует по файлам, ищет определения, читает исходный код, анализирует зависимости — как опытный разработчик.

03

Получите ответ

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

04

Источники

Каждый ответ содержит список конкретных файлов и строк, которые послужили основой. Всегда можно проверить.

Пример: Apache Superset — кэширование запросов

ШАГИ ИССЛЕДОВАНИЯ

🤖 Exploring workspace structure...

🤖 Reading cache config files...

🤖 Analyzing Redis/Memcached backends...

🤖 Checking TTL override chain...

✅ Answer ready · confidence: high

ИСТОЧНИКИ

`superset/config.py:1101–1140`

CACHE_DEFAULT_TIMEOUT, CACHE_CONFIG, DATA_CACHE_CONFIG

`docs/configuration/cache.mdx:1–243`

Redis, Memcached backends, TTL стратегии

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

TTL chain: custom → chart → datasource → db

РЕАЛЬНЫЙ ПРИМЕР

Apache Superset — исследование кэширования

01 — ВОПРОС ПОЛЬЗОВАТЕЛЯ

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
`CACHE_CONFIG`	Метаданные Superset	NullCache (нет TTL)
`DATA_CACHE_CONFIG`	Данные чартов / запросов к датасетам	NullCache (нет TTL)
`FILTER_STATE_CACHE_CONFIG`	Состояние фильтров дашборда	90 дней
`EXPLORE_FORM_DATA_CACHE_CONFIG`	Данные формы Explore	7 дней
`THUMBNAIL_CACHE_CONFIG`	Скриншоты/превью	7 дней
`RESULTS_BACKEND`	Результаты async SQL Lab	без TTL (объект cachelib)
`SIGNAL_CACHE_CONFIG`	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: КОНКРЕТНЫЕ ФАЙЛЫ И СТРОКИ КОДА

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.

КЛЮЧЕВЫЕ ВОЗМОЖНОСТИ

Всё, что нужно enterprise-команде

Две модели под задачу

4B (Qwen 3.5) — работает на рабочем ноутбуке разработчика. 8GB RAM, CPU.

14B (Qwen Dense) — для серверного развёртывания. Мощнее и точнее. 16GB+ VRAM.

Быстрое развёртывание

Docker-контейнер или нативная установка. Подключение к репозиторию и начало работы — менее чем за час. Полная поддержка при развёртывании.

Любые языки

Понимает любые языки программирования. Специально обучена на 20 ключевых: TypeScript, Python, Go, Rust, Java, C#, Kotlin, Swift, PHP и другие.

Изолированный контур

Работает без интернета. Без зависимостей от внешних API. Соответствие ФЗ-152, GDPR, требованиям ФСТЭК и ФСБ.

Источники и уверенность

Каждый ответ — с оценкой уверенности и точными ссылками на файлы и строки кода. Всегда понятно, на чём основан ответ.

Безлимитные лицензии

Одна лицензия — вся компания. Разработчики, аналитики, тестировщики, менеджеры. Без ограничений по количеству пользователей.

СРАВНЕНИЕ

Инструмент понимания, а не генерации

*Copilot, Cursor и Cody фокусируются на **генерации нового кода**. AI Insider решает другую задачу — **понимание существующего**. Это не конкуренция, это другой инструмент для другой проблемы.*

Возможность	AI Insider	GitHub Copilot	Cursor	Cody
On-premise развёртывание	✓	✗	✗	✗
Глубокое исследование кода	✓	✗	✗	✗
Предобученная модель	✓	✗	✗	✗
Ссылки на файлы и строки	✓	✗	✗	✗
Работает без интернета	✓	✗	✗	✗
Безлимитные лицензии	✓	✗	✗	✗
Понимает любые языки	✓	✓	✓	✓

ЭФФЕКТ ДЛЯ БИЗНЕСА

Что получает команда после внедрения

⚡

+20–35% скорости 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 в кодовую базу.

↗ Cortex State of Dev Productivity 2024

📉

−40% нецелевых трудозатрат

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

↗ Stack Overflow Dev Survey 2023

🛡️

Снижение bus factor

Критическая экспертиза о системах перестаёт быть заложницей 1–2 ключевых людей. Знания о коде становятся 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 команды 50 человек

Разработчики, аналитики, тестировщики, DevOps. Средняя стоимость часа с накладными расходами — 1 875 ₽.

СРЕДНЯЯ ЗРЕЛОСТЬ

736 000

₽ / мес

экономия при стандартных процессах и документации

НИЗКАЯ ЗРЕЛОСТЬ (LEGACY)

× 2,7

ROI

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

КОРПОРАТИВНАЯ ЛИЦЕНЗИЯ

~5 мес

окупаемость

при 277 778 ₽/мес и средней зрелости процессов

ПОИСК ПО КОДУ

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

ОНБОРДИНГ

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

ДОКУМЕНТАЦИЯ

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

⚠ Расчёт основан на публичных исследованиях: Stack Overflow Dev Survey 2023, Cortex 2024, Google RCT arxiv 2024, IBM Research. Реальный эффект зависит от специфики команды, кодовой базы и зрелости процессов.

СТОИМОСТЬ

Прозрачные условия.
Без ограничения лицензий.

КОРПОРАТИВНАЯ ЛИЦЕНЗИЯ

ВЫГОДНО

₽10 000 000

за 3 года · бессрочная лицензия

- ✓ Безлимитные лицензии пользователей
- ✓ Обе модели: 4B и 14B
- ✓ 20 языков программирования
- ✓ Техническая поддержка 3 года
- ✓ Обновления модели и платформы
- ✓ Помощь с развёртыванием
- ✓ Обучение команды

ПОМЕСЯЧНАЯ ОПЛАТА

₽400 000

в месяц · гибкий формат

- ✓ Безлимитные лицензии пользователей
- ✓ Обе модели: 4B и 14B
- ✓ 20 языков программирования
- ✓ Техническая поддержка
- ✓ Обновления модели и платформы
- ✓ Помощь с развёртыванием
- ✓ Обучение команды

💡 *Корпоративная лицензия на 3 года — это **₽277 778 в месяц** против ₽400 000. Экономия 30% при горизонте планирования 3 года.*

СЛЕДУЮЩИЙ ШАГ

Готовы увидеть AI Insider в деле?

Мы проведём демонстрацию на ваших или open-source проектах. Вы увидите, как AI Insider исследует именно ваш стек — до принятия решения.

Запросить демо →

Смотреть лендинг

САЙТ

buff.systems/landings/ai-insider

ПРИМЕРЫ ИССЛЕДОВАНИЙ

5 000+ реальных исследований