Enterprise RAG Challenge 3

Создаем агента, входящего в ТОП с нуля с помощью AI-assistant development

7 место • 162 задачи • 360995 запусков агентов

Константин Крестников
Управляющий директор, Сбер
Лид команды GigaChain

Дисклеймер

- Мы будем пользоваться Cursor. Я его оплатил с личной карты и использовал на личном компьютере. Вопросы о том, как использовать его внутри Сбера сегодня обсуждать не будем.
- Вопросы можно задать в чате по ходу воркшопа, я отвечу на них в конце.
- Просьба не задавать вопросы, которые касаются внутренних процессов Сбера и GigaChat. Встреча открытая!

Agenda

1. Что такое ERC3
2. Архитектура агента
3. Настройка Cursor
4. Создание агента с нуля
5. Win & Fail
6. Пробуем GigaChat
7. Дополнительные ресурсы
8. Q&A

Демо: интерфейс ERC3, настройка Cursor, разработка агента, запуск на разных моделях

Что такое ERC3?

ERC3 - соревнование AI агентов, которые должны решать задачи в рамках организаций, приближенных к реальным.

- **Организация + HTTP API** — магазин, офис или крупная компания с набором эндпоинтов для взаимодействия
- **Текстовые задачи** — например, "Купить все GPU в интернет-магазине"
- **Агент на стороне пользователя** — отправляет решения по API
- **Свобода архитектуры** — любые модели, любой подход (можно даже вручную)
- **Ограниченное время** — в момент запуска выдаются новые задачи, агент должен успеть решить и отправить ответы - хардкод не работает

Площадка продолжает работать, зарегистрироваться можно тут: https://erc.timetoact-group.at/

Типы задач в ERC3

Бенчмарк	Задачи	Назначение
`demo`	4	Тест инфраструктуры
`store`	15	Онлайн-магазин
`erc3-dev`	16	Разработка агента для корпорации
`erc3-test`	24	Разработка агента для корпорации (расширенный тест)
`erc3-prod`	**103**	**Финал**

Какой у меня был план

(и я его придерживался)

- Потратить 2–4 часа
- Реализовать все с помощью AI Assistant coding
- Сделать реализацию на стеке GigaChain с возможностью проверить работу агента на GigaChat
- Запустить цикл самоулучшения
- Применить наши лучшие практики — think tool, планирование, reasoning в JSON полях

Что в итоге получилось

- Потратил в итоге ~10 часов
- Создан агент на базе GPT-5.1 + LangGraph ReAct Agent
- Агент может работать на базе GigaChat
- Удалось запустить цикл самоулучшения
- Агент занял седьмое место
- Стоимость финального прогона: **$3.62** за 103 задачи
-

Prize Leaderboard

Total submissions: 38 • Cutoff: 2025-12-09 13:40 CET (3 hours) • Evals hidden

#	TEAM	SESSION NAME	SCORE	COST	SUBMITTED	TASK
1	VZS9FL	▶ @aostrikov claude sequential evolution	0.718	34.21	2025-12-09 11:30	6m 38s
2	NLN7Dw	▶ Ilia Ris	0.621	0.56	2025-12-09 13:11	5m 43s
3	Kc7F2N	▶ Function Calling Agent (gpt-4.1) v17 removed find_employee	0.612	5.46	2025-12-09 10:34	38s
4	MMzXeM	▶ Simple Agent & deepseek-reasoner A. Ovsov.	0.602	0.63	2025-12-09 10:26	7m 47s
5	f1Uixf	▶ Langchain Tool Agent openai/gpt-4.1	0.544	16.29	2025-12-09 10:46	17s
6	K8khZ8	▶ CC SDK ERC3 Agent	!0.534	1.78	2025-12-09 12:58	4m 58s
7	xoDvsa	▶ @Krestnikov (Giga team)	0.515	3.62	2025-12-09 11:45	32s
8	Lcnxuy	▶ @andrey_aiweapps - ERC3 Challenge Agent	0.505	14.41	2025-12-09 10:35	1m 26s
9	MgSeuz	▶ NextStep SGR (google/gemini-2.5-flash) from ERC3 Samples +pipelined	0.505	2.80	2025-12-09 10:59	27s
10	mx78kt	▶ @dimaprodev agent	0.495	1.41	2025-12-09 11:40	24s

Пример задачи 1

Buy all GPUs

- В магазине продается компьютерное железо — его можно просматривать через список всех товаров
- Часть железа — GPU A100 (4 штуки) и H100 (3 штуки)
- Их нужно добавить в корзину
- При вызове Checkout возвращается ошибка — Status: 400, Error: insufficient inventory for product gpu-h100 during checkout: available 1, in basket 3, Code 400. Две H100 уже купили, осталась одна
- Нужно удалить из корзины две H100
- Снова делаем чекаут — на этот раз все ок, покупка совершена.

Посмотреть содержимое задачи можно тут — https://erc.timetoact-group.at/tasks/tsk-42p4e73LVV7ZrWcAWs551q

Пример задачи 2

Buy 3× Dog Food Premium with the most discount. Coupons: DOGSALE, DOGGY10, DOGGY25, WOOF15

- В магазине продаются товары для животных
- Среди товаров есть премиальная собачья еда
- Купон DOGSALE не работает
- Купон WOOF15 не даёт скидку
- Купоны DOGGY10 и DOGGY25 дают 10% и 25% соответственно, но не суммируются
- В других задачах логика работы купонов может быть другой — агенту нужно попробовать различные варианты и найти оптимальный.

Посмотреть содержимое задачи можно тут — https://erc.timetoact-group.at/tasks/tsk-42p4w4i1FWmNdQxgU7hNiae

Часть 1

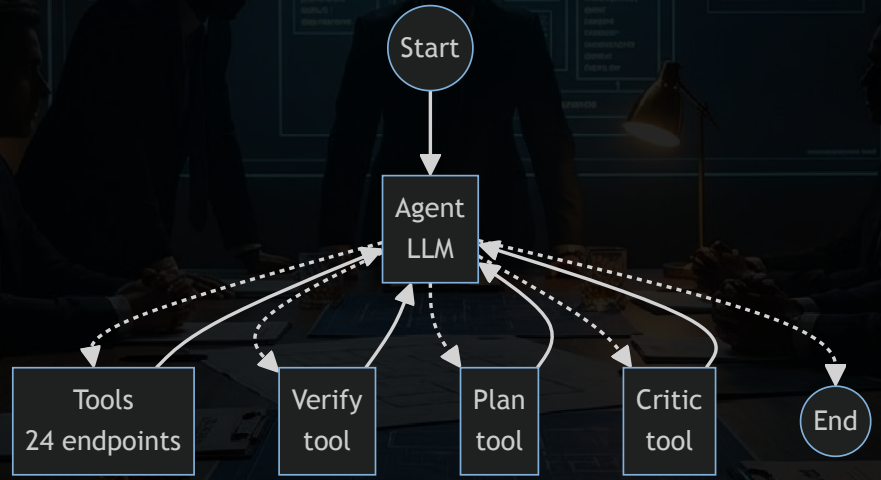
Интерфейс ERC3

... живое демо ...

Часть 2

Архитектура агента

Архитектура агента



Пример тула

Этот пример иллюстрирует функцию структурированной проверки ответа агента перед отправкой

```
def verify_function(
    outcome: str,
    employee_links: str,
    project_links: str,
    customer_links: str,
    made_modifications: bool,
    permissions_checked: bool,
    wiki_checked: bool,
    reasoning: str
) → str:
    """
    Структурированная верификация перед финальным ответом.
    Проверяет что агент явно продумал outcome, links и соблюдение правил.
    """
```

Часть 3

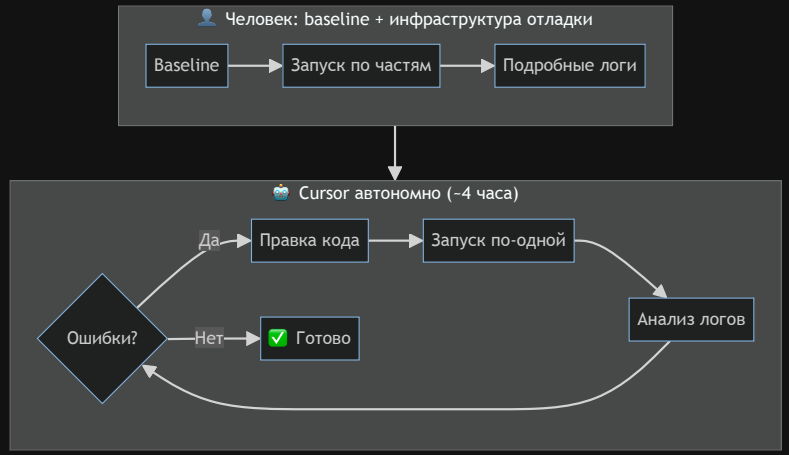
Настроим Cursor

... Живое демо ...

Часть 4

Создание агента с нуля

Цикл самоулучшения



- **Человек:** создаёт baseline, добавляет запуск по частям (одна задача / до первой ошибки), настраивает логи
- **Cursor:** запускает задачи по одной → читает логи → правит код → повторяет до успеха

...Живое демо...

Часть 5

Win & Fail

Fail

- Проскочил этап планирования и исследования – сразу пошел решать
- Не догадался решать задачи параллельно
- Не стал делать предварительную выгрузку данных. Это делает агент и это повышает на него нагрузку
- Не предполагал, что в финале задач будет так много

Win

- Угадал с моделью — идеальный баланс времени и качества
- Удалось вовремя среагировать на изменение правил финала (1 → 3 часа)
- LangGraph спас ситуацию — встроенная защита от зацикливаний сработала там, где я не подумал
- Вышло недорого — $3.62 за весь финал

Часть 6

Пробуем GigaChat

... живое демо ...

Результаты

- Важно, что промпты затюнены под OpenAI. После небольшого тюнинга на GigaChat также достигается 100%.
- Стоимость прогона 15 задач на GPT-5.2: **$1.5**
- Итоговый финал на 103 задачи может быть использован как бенчмарк агентов/моделей

Часть 7

Дополнительные ресурсы

Промпты

1. Изучи как работает store_agent.py, обрати внимание на типизацию аргументов функций и их описание. Как они реализованы?
2. Переделай store_agent на langchain с использованием create_agent. Возьми пример реализации из @docs/agents.md . Сохрани точно сигнатуры функций, название и описание аргументов, описания функций, чтобы они были точно такими же как у исходного решения. Добавь функции plan и critic, которые в качестве аргументов принимают рассуждения и критику модели. В промпт добавь необходимость их вызвать в начале и в конце. Добавь обработку ошибок внутри функций. Если функция упала с ошибкой - модель должна это увидеть и иметь возможность исправить ситуацию. Лимит на количество рекурсий агента - 50.
3. Сделай так, чтобы задачи можно было запускать по-одной с помощью ключей в main.py, затем запускай их по-одной и добейся чтобы проходили все
4. Поменяй модель на GigaChat с использованием from langchain_gigachat import gigachat. Она уже установлена.

Подсказки

1. Клонируем репозиторий с примером: git clone git@github.com:trustbit/erc3-agents.git
2. Настраиваем Cursor: 2.1. Подключаем MCP docs-langchain 2.2. Добавляем документацию по ERC3
3. Инициализируем venv (python3.13 -m venv .venv && source .venv/bin/activate && pip install -r requirements.txt)
4. Добавляем библиотеки: pip install langchain langgraph langchain-openai python_dotenv langchain-gigachat-lc1==0.4.0b4
5. Просим переписать на langchain create_agent
6. Просим добавить возможность запускать задачи по одной через ключи.
7. Формируем AGENTS.md – агент для участия в соревновании в рамках ERC3. Он получает задачи и должен их решать. При разработке учитывай, что задачи в будущем будут заменены, поэтому не используй жесткую привязку к конкретным значениям, чтобы избежать оверфита
8. Запускаем непрерывный цикл улучшения

MCP сервер langchain-gigachat

Настройка в Cursor: `~/.cursor/mcp.json`

```
{
  "mcpServers": {
    "docs-langchain": {
      "url": "https://docs.langchain.com/mcp"
    }
  }
}
```

- MCP (Model Context Protocol) позволяет подключать внешние источники данных к AI-ассистенту
- Сервер `docs-langchain` предоставляет актуальную документацию LangChain прямо в контексте Cursor
- После настройки Cursor автоматически получает доступ к документации при работе с кодом

Часть 8

Финал

Спасибо за внимание!

Готов ответить на вопросы

@Robofuture
Telegram про ИИ

Константин Крестников
Управляющий директор, Сбер
Лид команды GigaChain

GigaChain
GitHub