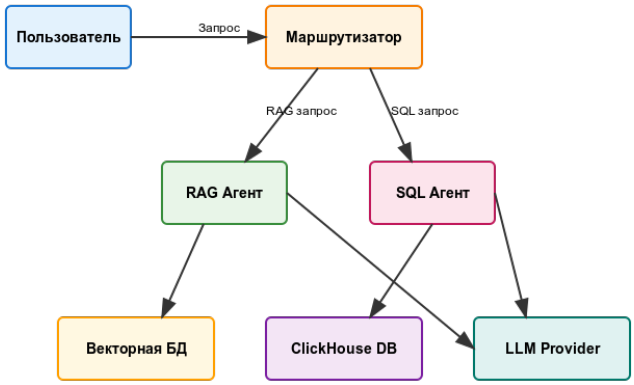
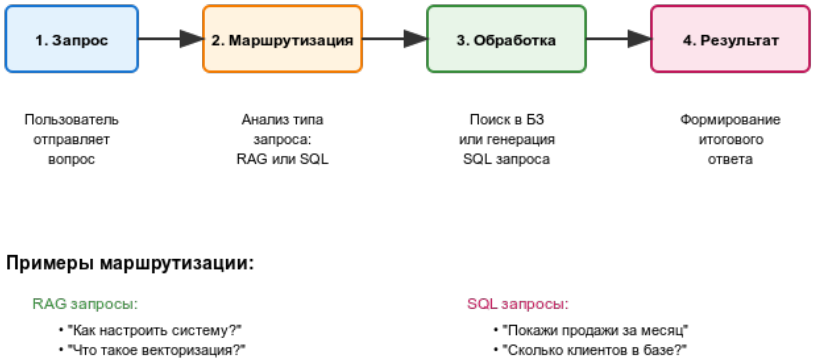
# Ядро системы - Маршрутизатор и RAG

## Создание основы мультиагентной системы

## Архитектура системы

В основе нашей мультиагентной системы лежит **Маршрутизатор** - это "диспетчер", который анализирует вопрос пользователя и решает, какой именно агент должен его обработать. По умолчанию система использует RAG для поиска информации в базе знаний.






Каждый агент выполняет свою специализированную задачу:

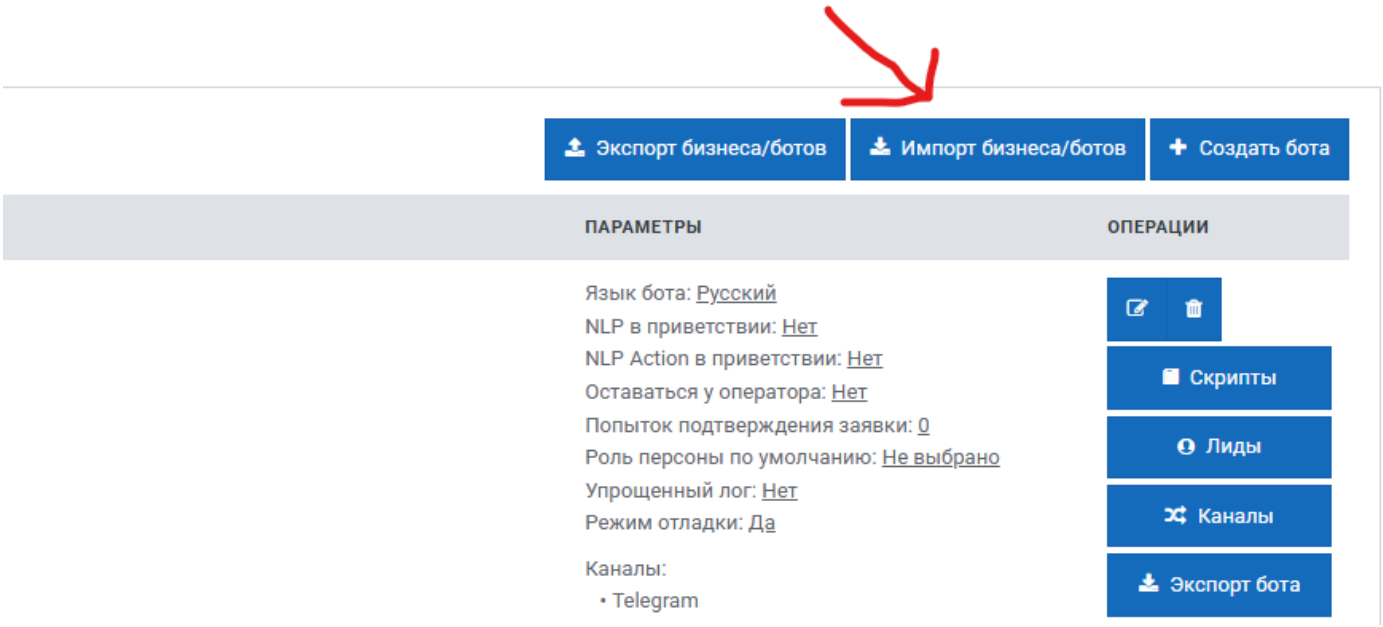
1. **CompanyInfo** - отвечает на вопросы о компании
2. **SQL-агент** - работает с базами данных

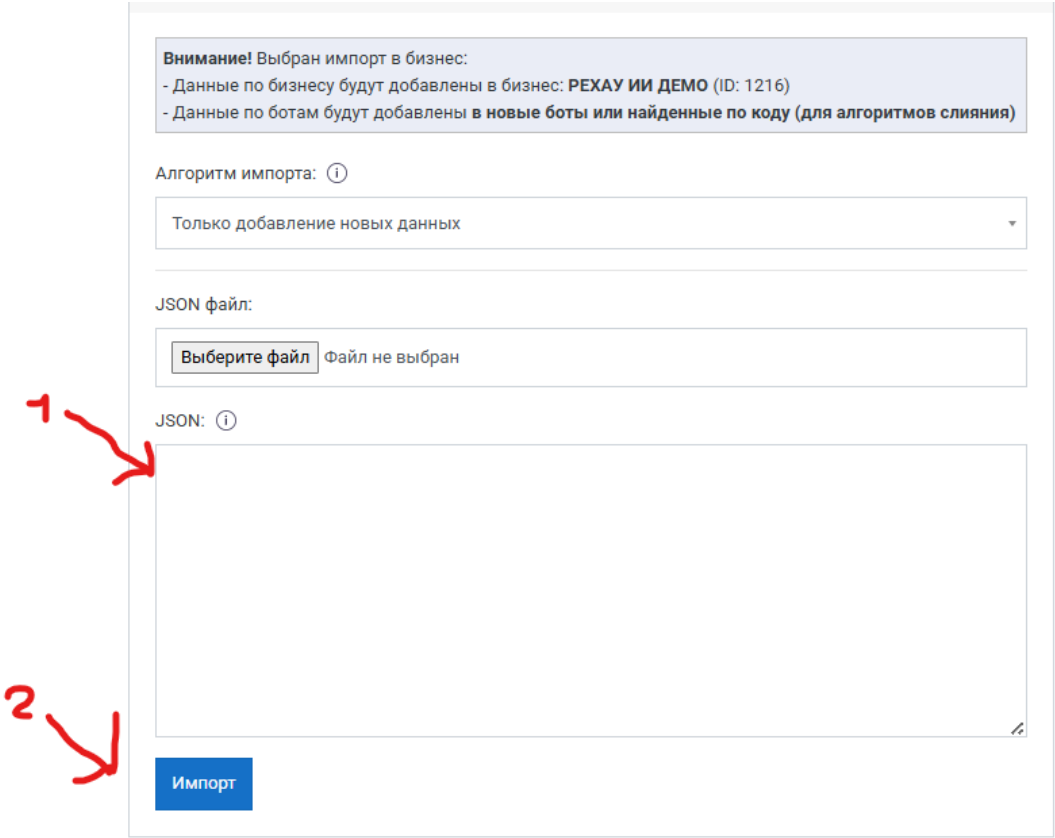
# Часть 1: Установка и настройка основного агента

## Шаг 1: Импорт готового решения

Мы подготовили готовую конфигурацию, которую можно быстро развернуть:

1. Перейдите по предоставленной ссылке для скачивания конфигурации
2. Скопируйте JSON-код конфигурации
3. В интерфейсе Метабот откройте раздел "Импорт бизнеса/ботов"
4. Вставьте скопированный JSON и нажмите "Импорт"




## Шаг 2: Проверка установленных компонентов

После успешного импорта в вашем боте должны появиться следующие элементы:

**1. Системный раздел MRAG** Это ядро системы, которое обрабатывает запросы и управляет агентами.

2. MRAG(Системный) (4)

| | | |
|---|---|---|
| Обработка ошибок | Standard | Code: RAG:ErrorFallback<br>Root Script: Yes |
| Flow 1: RAG Интент и КВ поиск | Standard | Code: RAG:DetectIntent_and_FindChunks<br>Root Script: Yes |
| Flow 2: Ответ пользователю | Standard | Code: RAG:UserReply<br>Root Script: Yes |
| Flow 0: Маршрутизатор | Standard | Code: MA:Router<br>Root Script: Yes |

**2. Базовые таблицы данных:**

- `gpt_knowledge_base` - хранилище знаний для RAG
- `gpt_prompts` - коллекция промптов для разных агентов

| | | | | | |
|---|---|---|---|---|---|
| 417 | Да | Да | Да | gpt_knowledge_base | База знаний |

**Структура таблицы (JSON словарь):**

| АКТИВНО | СИХР. СТРУКТУРЫ | НАИМЕНОВАНИЕ | ЗАГОЛОВОК | ТИП | ОБЯЗАТЕЛЬНОЕ | ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ | РАЗМЕР | Т |
|---|---|---|---|---|---|---|---|---|
| Да | Да | id | ID | AUTOINCREMENT | Да | NULL | | |
| Да | Да | context | Домен | TEXT | Да | | | |
| Да | Да | content | Контент | TEXTAREA | | | | |
| Да | Да | embeddings | embeddings | VECTOR | | NULL | 1536 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1613 | Да | Да | Да | gpt_knowledge_base | GPT - База знаний (ТЕСТ) | 2025-08-01 12:02:12 | 2025-08-06 12:41:26 | Структура / Данные / Обновить все поля в БД |

Структура таблицы:

| АКТИВНО | СИХР. СТРУКТУРЫ | НАИМЕНОВАНИЕ | ЗАГОЛОВОК | ТИП | СВЯЗЫВАЕМАЯ ТАБЛИЦА | ОБЯЗАТЕЛЬНОЕ | ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ |
|---|---|---|---|---|---|---|---|
| Да | Да | id | ID | AUTOINCREMENT | | Да | NULL |
| Да | Да | context | Домен | TEXT | | | |
| Да | Да | content | Контент | TEXTAREA | | | |
| Да | Да | embeddings | embeddings | VECTOR | | | NULL |

**3. Плагин управления "AgentsParams"** Центр управления всеми настройками агентов.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 309 | Да | Стандарт | SaaS | SaaS | AgentsParams | AgentsParams | 2025-06-27 13:36:18 | 2025-06-27 13:36:18 | Скрипты |

# Шаг 3: Настройка конфигурации агентов

Мы создали систему конфигураций, которая позволяет управлять агентами без глубокого понимания кода. Все настройки вынесены в понятные конфигурационные файлы.

> **Совет для работы:** В конфигурации есть множество параметров. Используйте поиск по документации, чтобы быстро найти описание нужного параметра и понять, как он работает.

## Основные поля конфигурации:

```
let activeAgent = lead.getAttr("activeAgent") // Устанавливается при вызове скрипта
let agentCFG = {} // Объект с настройками текущего агента

if (activeAgent === "MainFlow") {
  agentCFG = {
    common: {
      title: "Основной Flow с маршрутизатором", // Понятное название агента
      agentName: "MainFlow", // Техническое имя для системы
      promptTable: "gpt_prompts", // Таблица с промптами
      userQueryAttibName: "user_query", // Атрибут для вопроса пользователя
      historyMaxLength: 4, // Сколько сообщений помнить в истории
      exitScript: "KB:FollowUp", // Что выполнить после завершения диалога
    },

    detectRoute: { // Настройки маршрутизатора
      provider: "OpenAI",
      model: "gpt-4o",
      modelParams: {
        "temperature": 1
      },
      prompt: "$route_prompt", // Промпт для определения маршрута
      routerTools: [{
        tools: "RAG_Tools", // Набор доступных инструментов
        route: "RAG:DetectIntent_and_FindChunks", // Скрипт для RAG
      }],
      errorScript: "RAG:ErrorFallback", // Обработка ошибок
    },

    detectIntent: { // Детектор намерений пользователя
      provider: "OpenAI",
      model: "gpt-4o",
      modelParams: {
        "temperature": 1
```

```
      },
      prompt: "$rag_intent_prompt", // Промпт для анализа намерений
      errorScript: "RAG:ErrorFallback",
      kbName: "defKnowBase", // База знаний
      kbDomain: "main", // Домен знаний
    },

    userReply: { // Генератор ответов пользователю
      provider: "OpenAI",
      model: "gpt-4o",
      modelParams: {
        "temperature": 1
      },
      errorScript: "RAG:ErrorFallback",
      useHistory: 1, // Использовать контекст диалога
      addUserQuery: 1, // Добавлять вопрос в промпт
      sendBotAnwser: 1, // Отправлять ответ пользователю
      systemPrompts: { // Структура промптов
        start: [["$start_prompt", "$rag_prompt"]], // Начальные промпты
        final: ["$final_prompt"] // Финальные промпты
        // Между start и final автоматически вставляется история диалога
      },
    }
  }
} else if (activeAgent === "newAgent") {
  // Здесь описываются конфигурации других агентов
  // Поля могут отличаться в зависимости от специализации агента
} else {
  bot.sendMessage(`MainConfig:snippet - неизвестный activeAgent: ${activeAgent}`)
  bot.stop()
}
```

# Шаг 4: Настройка таблицы промптов

Промпты - это инструкции для ИИ. Они определяют, как агент будет вести себя и отвечать на вопросы.

**Процедура настройки:**

1. Откройте таблицу `gpt_prompts` в интерфейсе Метабот
2. Создайте промпт для каждого агента с уникальным именем

3. Заполните три обязательных поля:
   - `agent_name` - имя агента (например: "MainFlow")
   - `name` - название промпта (например: "route_prompt")
   - `prompt` - содержание промпта с макропеременными

**Пример настройки промпта маршрутизатора в таблице:**

```
agent_name: MainFlow
name: route_prompt
prompt: Ты агент ИИ который отвечает на вопросы о компании {{@company_name}}.

У тебя есть инструменты в распоряжении:
<tools>
**RAG_Tools** - Используется, когда нужно ответить на вопросы об услугах
и продукции компании, об условиях эксплуатации продукции, об используемых
инструментах для монтажа продукции и т.д.
</tools>

Учитывая историю и запрос пользователя, сделай вывод о реальном намерении
пользователя. Отвечай на русском языке без лишних рассуждений и вопросов.

История чата:
"""{{$chat_history_str}}"""

Запрос пользователя:
"""{{$user_query}}"""

Пожалуйста, отвечай как можно точнее и всегда обязательно указывай инструмент (Tools).
Ответ всегда должен быть представлен в виде JSON-объекта, в котором выводятся
указанные ниже поля. JSON в самом конце! Без пояснений.

{
  "tools": "",  string: название выбранных вами инструментов
  "user_intent": "",  string: подробное намерение пользователя для системы RAG
  "request": ""  "полученный вами запрос"
}
```

## Система макросов в промптах

Для динамической подстановки данных используйте макросы:

**Данные из лида:** `{{$lead_attr}}` - любой атрибут лида
**Данные из бота:** `{{@bot_attr}}` - любой атрибут бота

**Системные макросы:**

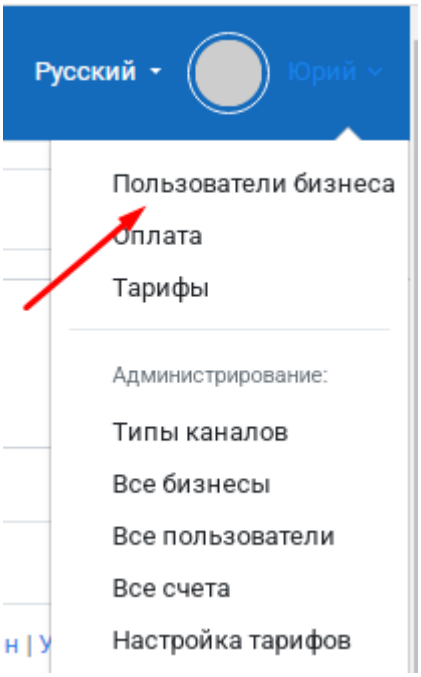
- `{{$chat_history_str}}` - история диалога в текстовом виде
- `{{$user_query}}` - текущий вопрос пользователя

# Часть 2: Настройка API-доступа

## Шаг 5: Создание API-пользователя

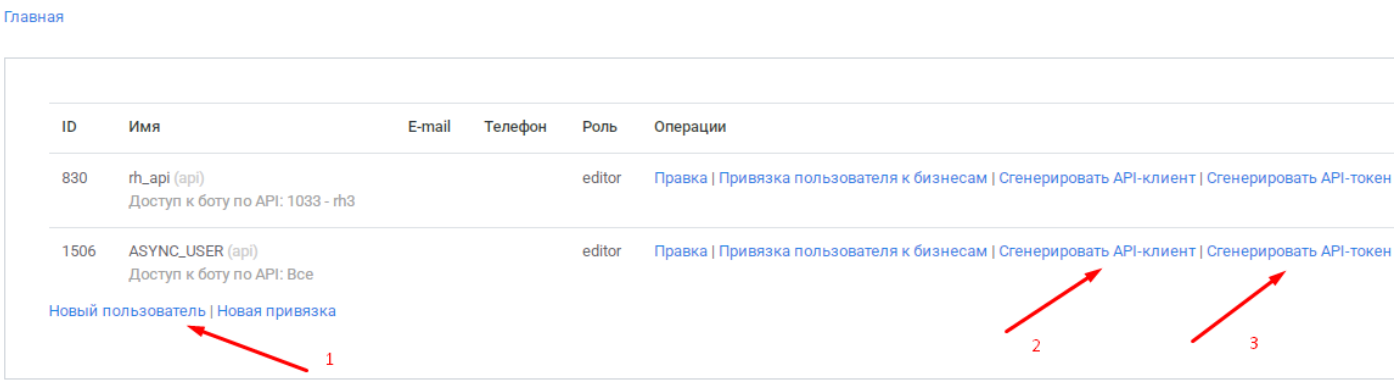
Для работы мультиагентной системы необходимо настроить API-доступ:

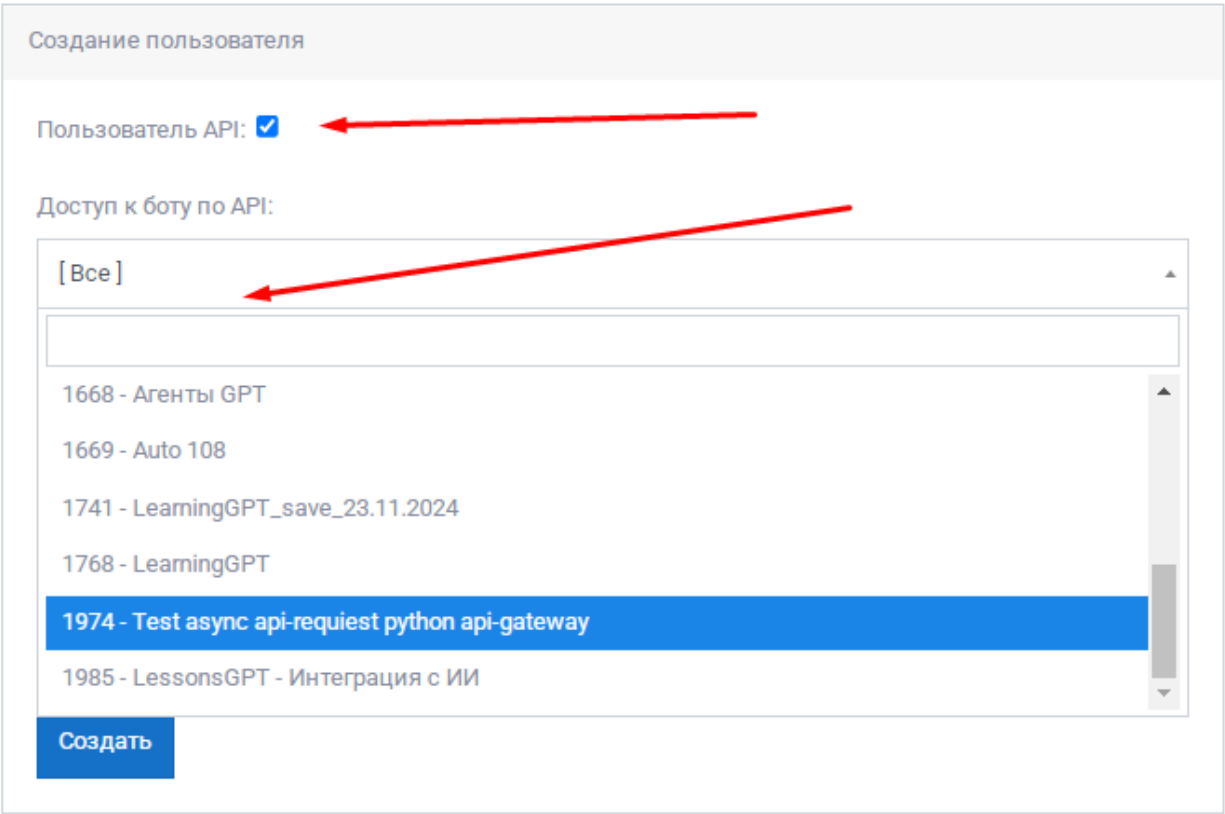
1. В настройках бизнеса создайте нового пользователя
2. Установите галочку "Пользователь API"
3. Предоставьте доступ к нужным ботам (всем или выбранным)
4. Назначьте соответствующую роль доступа




**Генерация токена доступа:**

1. Создайте API-клиента для пользователя
2. Сгенерируйте токен доступа
3. **Важно:** Сохраните токен - он потребуется для подключения к API-шлюзу
4. **Обязательно:** Используйте токен в режиме 3




## Шаг 6: Регистрация в API-шлюзе

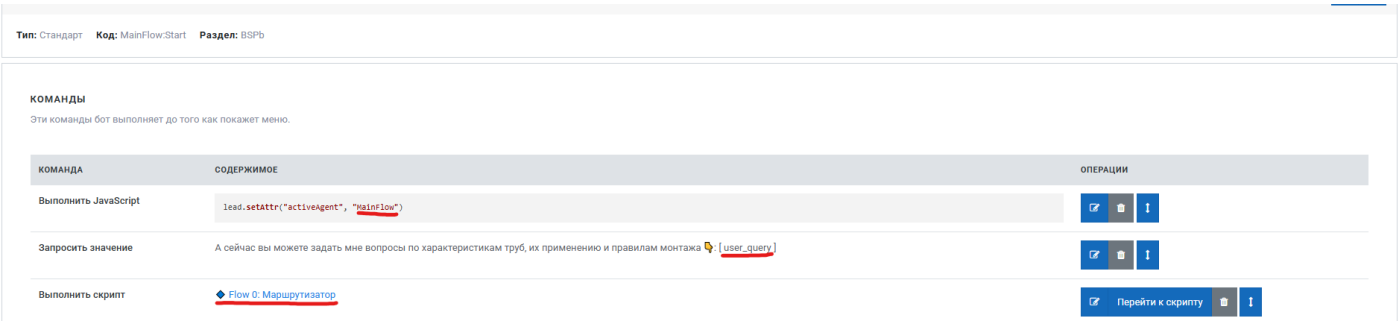
API-шлюз обеспечивает асинхронное взаимодействие между компонентами системы.

1. Обратитесь в поддержку. Сейчас регистрация производится специалистами Метабот.
2. Сообщите им параметры регистрации:
   - `bot_id` - идентификатор вашего бота
   - `token` - токен API, полученный на предыдущем шаге
   - `domain` - доменное имя сервера, где размещен бот. Например `https://app.metabot24.com`

# Часть 3: Запуск и тестирование

# Шаг 7: Создание пользовательского интерфейса

Создайте скрипт, который позволит пользователям взаимодействовать с системой:




**Ключевые параметры запуска:**

1. **MainFlow** - выбор основного агента для обработки запроса
2. **user_query** - атрибут для сохранения вопроса пользователя
3. **Flow 0** - запуск скрипта маршрутизатора

# Результат работы системы

При правильной настройке ваша мультиагентная система будет:

- **Понимать контекст** диалога и поддерживать связную беседу
- **Автоматически определять** наиболее подходящий агент для каждого вопроса
- **Использовать базу знаний** для предоставления точных ответов
- **Обращаться к специализированным агентам** при необходимости (SQL-агент, CompanyInfo и др.)

Система готова к работе и может быть расширена дополнительными агентами в зависимости от ваших потребностей.

---

Версия #2
Павел Борисов создал 5 September 2025 13:11:37
Artem Garashko обновил 12 November 2025 14:03:47