



Инструкция по исправлению проблемы с отображением ответов в чате

Автор: MiniMax Agent

Дата: 30 июня 2025



Описание проблемы

Ответы от LLM через API поступают в терминал, но не отображаются в интерфейсе чата, хотя в логах видно, что данные доходят до чата.



Корень проблемы

Основная проблема: Несоответствие типов данных между `CrewAIClient` и `ChatWidget`.

Детали:

1. В `crewai_client.py` (строка 87):

```
python return data["response"] # ← Возвращает СТРОКУ
```

2. В `chat_widget.py` (строки 199-204):

```
python if process_result and "response" in process_result: # ←  
Ожидает СЛОВАРЬ response = process_result["response"] elif  
process_result and "error" in process_result: response = f"Ошибка  
CrewAI: {process_result["error"]}" else: response = "Неизвестный  
ответ от CrewAI." # ← Всегда выполняется!
```

3. Результат:

- `process_result` получает строку
- Условие `"response" in process_result` проверяет наличие подстроки "response" в тексте ответа
- Если слова "response" нет в ответе, выполняется `else` с сообщением "Неизвестный ответ от CrewAI."

Решение

Вариант 1: Изменить `crewai_client.py` (РЕКОМЕНДУЕТСЯ)

Заменить метод `process_request` в файле `crewai_client.py`:

```

def process_request(self, message, force_crewai=False):
    """
    Обработывает запрос через CrewAI API

    ИСПРАВЛЕНИЕ: Теперь возвращает структурированный объект
    """
    if not self.is_available():
        return {
            "response": f"CrewAI API сервер недоступен...",
            "error": "server_unavailable",
            "processed_with_crewai": False
        }

    try:
        response = requests.post(
            f"{self.base_url}/api/process",
            json={"message": message, "force_crewai": False},
            timeout=60
        )

        if response.status_code == 200:
            data = response.json()

            # ИСПРАВЛЕНИЕ: Возвращаем полный объект
            if "response" in data:
                return {
                    "response": data["response"],
                    "processed_with_crewai":
data.get("processed_with_crewai", False)
                }
            elif "error_message" in data:
                return {
                    "response": data["error_message"],
                    "error": data.get("error", "unknown_error"),
                    "processed_with_crewai":

```

```

data.get("processed_with_crewai", False)
        }
    else:
        return {
            "response": "Неизвестный формат ответа от
сервера",
            "error": "invalid_response_format"
        }
    else:
        return {
            "response": f"Ошибка HTTP {response.status_code}",
            "error": f"http_error_{response.status_code}"
        }
except requests.RequestException as e:
    return {
        "response": f"Ошибка связи: {str(e)}",
        "error": "connection_error"
    }

```

Вариант 2: Изменить `chat_widget.py`

Заменить логику обработки ответа в методе `send_message`:

```

# В функции process_in_background()
try:
    process_result = self.crew_ai_client.process_request(text)
    logger.info(f"Получен результат: {process_result}")

    # ИСПРАВЛЕНИЕ: Обрабатываем и строку, и словарь
    if isinstance(process_result, dict):
        if "response" in process_result:
            response = process_result["response"]
        elif "error" in process_result:
            response = f"Ошибка: {process_result['error']}"
        else:
            response = "Неизвестный формат ответа"
    elif isinstance(process_result, str):
        response = process_result # Прямое использование строки
    else:
        response = "Неподдерживаемый тип ответа"

except Exception as e:
    response = f"Ошибка обработки: {str(e)}"

```



Дополнительные улучшения

1. Улучшенная замена сообщения ожидания

```
def _update_assistant_response(self, waiting_id, response,
error_occurred=False):
    """Заменяет сообщение ожидания на реальный ответ"""
    try:
        current_html = self.history.toHtml()
        waiting_span = f"<span id='{waiting_id}'>⌚ Обрабатываю
запрос...</span>"

        if waiting_span in current_html:
            # Заменяем сообщение ожидания
            updated_html = current_html.replace(waiting_span,
response)
            self.history.setHtml(updated_html)
        else:
            # Fallback: добавляем новое сообщение
            self.append_message("Ассистент", response)
    except Exception as e:
        logger.error(f"Ошибка обновления: {e}")
        self.append_message("Ассистент", response)

    self.send_btn.setEnabled(True)
```

2. Методы отладки

```
def test_crewai_connection(self):
    """Тестирует соединение с CrewAI API"""
    if not self.crew_ai_client:
        self.append_message("Система", "❌ CrewAI клиент не инициализирован")
        return False

    if not self.crew_ai_client.is_available():
        self.append_message("Система", "❌ CrewAI API сервер недоступен")
        return False

    try:
        test_result = self.crew_ai_client.process_request("Тест соединения")
        self.append_message("Система", f"✅ Тест успешен: {test_result}")
        return True
    except Exception as e:
        self.append_message("Система", f"❌ Ошибка теста: {e}")
        return False
```

Пошаговая инструкция

1. **Сделайте резервную копию** оригинальных файлов
2. **Замените** `crewai_client.py` на исправленную версию
3. **Обновите методы в** `chat_widget.py`:
 - `send_message()`
 - `_update_assistant_response()`
4. **Добавьте метод тестирования** `test_crewai_connection()`

5. **Перезапустите приложение**

6. **Протестируйте** отправку сообщения

Тестирование

```
# В консоли Python или через метод в chat_widget.py
chat_widget.test_crewai_connection()
```





Логирование для отладки

Добавьте подробное логирование:

```
logger.info(f"Тип результата: {type(process_result)}")
logger.info(f"Содержимое результата: {process_result}")
logger.info(f"Проверка 'response' in result: {'response' in
str(process_result)}")
```

Ожидаемый результат

После исправления:

1.  Ответы LLM корректно отображаются в интерфейсе чата
2.  Сообщения ожидания заменяются реальными ответами
3.  Ошибки обрабатываются и отображаются корректно
4.  Логи показывают правильную обработку данных

Возможные дополнительные проблемы

1. **Кодировка текста** - убедитесь, что все файлы в UTF-8
2. **Потоки Qt** - используйте `QTimer.singleShot(0, update_ui)` для обновления UI

3. **HTML-экранирование** - используйте `html.escape()` для безопасного отображения

Эта инструкция должна полностью решить проблему с отображением ответов в чате!