

Case: Модификация кода с помощью CoPilot AI - Расширение доступа к результатам голосования

Этот кейс демонстрирует процесс модификации кода в Low-Code приложении Metabot для голосования с использованием ChatGPT4 в качестве AI-ассистента CoPilot.

Исходная задача заключается в расширении функционала приложения: ранее результаты голосования могли просматривать только владельцы и старшие партнеры, но теперь доступ к результатам голосования предоставляется всем участникам проекта, которые принимали участие в голосовании.

В кейсе подробно рассматривается процесс модификации кода, включая дословную формулировку запросов к CoPilot, анализ предложенных им решений, корректировку и интеграцию изменений в существующий код.

В конце автор делится субъективными ощущениями об эффективности использования CoPilot в процессе разработки и утверждает, что с его помощью удалось сэкономить значительное количество времени.

В кейсе представлен только фрагмент работы оператора с CoPilot, фокусируясь на модификации конкретного кода. Важно отметить, что взаимодействие с CoPilot началось значительно раньше, и в процессе этого взаимодействия CoPilot был обучен пониманию модели данных приложения, особенностям проекта и другим важным деталям. Это предварительное обучение CoPilot позволило ему эффективно ассистировать в процессе модификации кода, обладая необходимым контекстом и пониманием структуры данных и ключевых элементов приложения.

Prompt 1

Я хочу изменить логику работы куска кода, который отвечает за проверку, имеет ли человек доступ к результатам голосования или нет. Я хочу сделать не как сейчас, проверяем статус, является ли участник владельцем проекта или старшим партнером. Если он участвовал или

был приглашен к голосованию, то есть если у него есть запись, хотя бы одна запись, если у него есть запись в этом голосовании, в таблице votes есть запись для этого голосования, поправь, пожалуйста, код:

```
var current_member_id = lead.getAttr("member_id");
var entered_voting_id = lead.getAttr("voting_id");

// Поиск голосования и его статуса
var voting = table.find("votings", ["id", "status_id", "project_id",
                                     "object_type", "object_id", "description",
                                     "start_date", "end_date", "url"],
                        [{"id", "=", entered_voting_id}][0];

if (!voting) {
    memory.setAttr("voting_result", "VOTING_NOT_FOUND");
    return;
}

// Добавление дополнительной информации о голосовании
var project_name = table.find("projects", ["name"], [{"id", "=",
voting.project_id}][0]?.name;
var subject_type_name = table.find("subject_types", ["name"], [{"code", "=",
voting.object_type}][0]?.name;
memory.setAttr("project_name", project_name);
memory.setAttr("subject_type_name", subject_type_name);
memory.setAttr("subject_id", voting.object_id || "Не указано");
memory.setAttr("description", voting.description);
memory.setAttr("start_date", voting.start_date);
memory.setAttr("end_date", voting.end_date);
memory.setAttr("url", voting.url);

var voting_status_id = voting.status_id;
var voting_status = table.find("voting_statuses", ["name"], [{"id", "=",
voting_status_id}][0]?.name;
memory.setAttr("voting_status", voting_status);

/// Получение всех статусов пользователя в проекте
var user_statuses_records = table.find("project_member_statuses", ["status_id", [
    ["project_id", "=", voting.project_id],
    ["member_id", "=", current_member_id]
]]);
```

```

var user_statuses = user_statuses_records.map(function(record) { return record.status_id; });

// ID статусов, дающих доступ к результатам голосования
var access_status_ids = table.find("member_statuses", ["id"], [
  ["code", "IN", ["owner", "project_owner", "senior_partner"]]
]).map(function(record) { return record.id; }); // Преобразование результатов в массив ID

var hasAccess = user_statuses.some(function(status_id) {
  return access_status_ids.includes(status_id);
});

if (!hasAccess) {
  memory.setAttr("voting_result", "ACCESS_DENIED");
  return;
}

// Формирование таблицы результатов голосования
var votes = table.find("votes", ["member_id", "vote_type_id", "comment"], [{"voting_id", "=",
entered_voting_id}]);
var votesTable = "# | Участник | Голос | Комментарий\n" +
  "-----\n"

let n = 1
votes.forEach( vote => {
  var member_name = table.find("members", ["first_name", "last_name"], [{"id", "=",
vote.member_id}])[0];
  var vote_type = table.find("vote_types", ["name"], [{"id", "=",
vote.vote_type_id}])[0]?.name;
  var formatted_vote_type = {
    "За": "    За",
    "Против": "    Против",
    "Воздержался": "    Воздержался",
    "Ожидается": "    Ожидается"
  }[vote_type] || "    Ожидается";

  votesTable += n++ + " | " + member_name.first_name + " " + member_name.last_name + " | " +
    formatted_vote_type + " | " +
    (vote.comment || "") + "\n";
});

```

```
memory.setAttr("voting_results_table", votesTable);
```

ChatGPT

Чтобы изменить логику проверки доступа к результатам голосования, мы должны убедиться, что у пользователя есть хотя бы одна запись в таблице `votes` для данного голосования. Вот обновленный код, который учитывает этот критерий:

```
var current_member_id = lead.getAttr("member_id");
var entered_voting_id = lead.getAttr("voting_id");

// Поиск голосования
var voting = table.find("votings", ["id", "status_id", "project_id", "object_type",
"object_id", "description", "start_date", "end_date", "url"], [{"id", "=",
entered_voting_id}])[0];

if (!voting) {
    memory.setAttr("voting_result", "VOTING_NOT_FOUND");
    return;
}

// Проверка, участвовал ли пользователь в голосовании
var userVote = table.find("votes", ["id"], [{"voting_id", "=", entered_voting_id},
["member_id", "=", current_member_id]]);

var hasAccess = userVote.length > 0;

if (!hasAccess) {
    memory.setAttr("voting_result", "ACCESS_DENIED");
    return;
}

// Формирование таблицы результатов голосования
var votes = table.find("votes", ["member_id", "vote_type_id", "comment"], [{"voting_id", "=",
entered_voting_id}]);
var votesTable = "# | Участник | Голос | Комментарий\n" +
    "-----\n"

let n = 1;
votes.forEach( vote => {
```

```

    var member_name = table.find("members", ["first_name", "last_name"], [{"id", "=",
vote.member_id}])[0];

    var vote_type = table.find("vote_types", ["name"], [{"id", "=",
vote.vote_type_id}])[0]?.name;

    var formatted_vote_type = {
        "За": "    За",
        "Против": "    Против",
        "Воздержался": "    Воздержался",
        "Ожидается": "    Ожидается"
    }[vote_type] || "    Ожидается";

    votesTable += n++ + " | " + member_name.first_name + " " + member_name.last_name + " | " +
        formatted_vote_type + " | " +
        (vote.comment || "") + "\n";

});

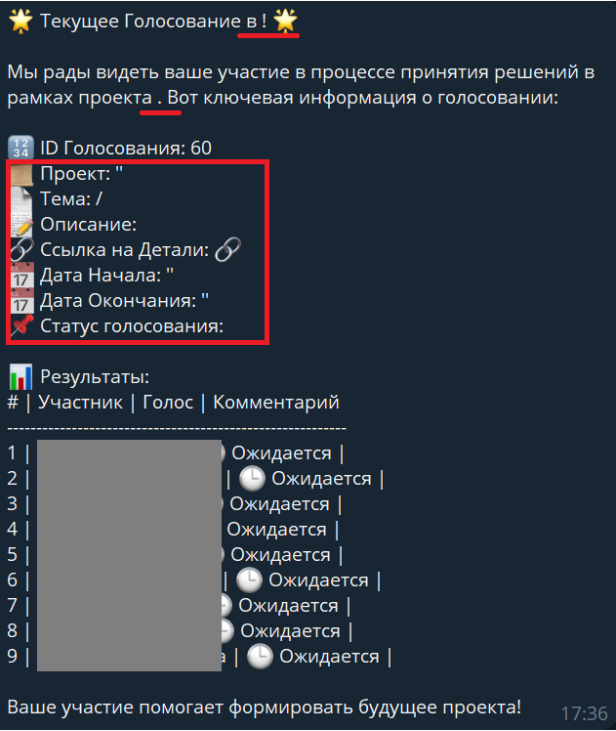
memory.setAttr("voting_results_table", votesTable);

```

В этой версии кода, проверка наличия доступа к результатам голосования основана на наличии записи участника в таблице `votes` для данного голосования. Если такая запись есть, пользователь имеет доступ к просмотру результатов голосования.

Комментарий оператора

Код верный, без ошибок - я проверил. Но GPT потерял часть кода, связанную с выводом карточки голосования — видимо он забыл об этом нюансе, ведь со-дизайн этого блока мы делали давно, а также он потерял переносы в длинной строке номер 5, но это я сам поправлю потом.



Исходный код этой части диалогового сценария:

Send Text	<pre>return memory.getAttr("voting_result") == "VOTING_NOT_FOUND"</pre>	К сожалению, мы не смогли найти голосование с указанным ID. Пожалуйста, проверьте данные и попробуйте снова.
Send Text	<pre>return memory.getAttr("voting_result") == "ACCESS_DENIED"</pre>	Извините, у вас нет доступа к результатам этого голосования. Доступ предоставляется определенным категориям участников проекта.
Send Text	<pre>return memory.getAttr("voting_results_table")</pre>	<p>Текущее Голосование в {{ &\$project_name }}!</p> <p>Мы рады видеть ваше участие в процессе принятия решений в рамках проекта {{ &\$project_name }}. Вот ключевая информация о голосовании:</p> <p>ID Голосования: {{ \$voting_id }}</p> <p>Проект: '{{ &\$project_name }}'</p> <p>Тема: '{{ &\$subject_type_name }}/{{ &\$subject_id }}</p> <p>Описание: '{{ &\$description }}</p> <p>Ссылка на Детали: '{{ &\$url }}'</p> <p>Дата Начала: '{{ &\$start_date }}</p> <p>Дата Окончания: '{{ &\$end_date }}</p> <p>Статус голосования: '{{ &\$voting_status }}</p> <p>Результаты:</p> <p>{{ &\$voting_results_table }}</p> <p>Ваше участие помогает формировать будущее проекта!</p>

Копируем эту часть скрипта с платформы Metabot в GPT и делаем новый запрос.

Prompt 2

Ты потерял блок с дополнительной инфой, для вывода контента, вот контент сценария: Send Text

```
return memory.getAttr("voting_result") == "VOTING_NOT_FOUND"
```

К сожалению, мы не смогли найти голосование с указанным ID. Пожалуйста, проверьте данные и попробуйте снова.

Send Text

```
return memory.getAttr("voting_result") == "ACCESS_DENIED"
```

Извините, у вас нет доступа к результатам этого голосования. Доступ предоставляется

определенным категориям участников проекта.

Send Text

```
return memory.getAttr("voting_results_table")
```

Текущее Голосование в {{ &\$project_name }}!

Мы рады видеть ваше участие в процессе принятия решений в рамках проекта {{ &\$project_name }}. Вот ключевая информация о голосовании:

ID Голосования: {{ \$voting_id }}

Проект: '{{ &\$project_name }}'

Тема: {{ &\$subject_type_name }}/{{ &\$subject_id }}

Описание: {{ &\$description }}

Ссылка на Детали: {{ &\$url }}

Дата Начала: '{{ &\$start_date }}'

Дата Окончания: '{{ &\$end_date }}'

Статус голосования: {{ &\$voting_status }}

Результаты:

```
{{ &$voting_results_table }}
```

Ваше участие помогает формировать будущее проекта!

ChatGPT

Понял вас, вот обновленный код с учетом блока дополнительной информации для вывода контента в сценарии:

```
var current_member_id = lead.getAttr("member_id");
var entered_voting_id = lead.getAttr("voting_id");

// Поиск голосования
var voting = table.find("votings", ["id", "status_id", "project_id", "object_type",
"object_id", "description", "start_date", "end_date", "url"], [{"id", "=",
entered_voting_id}])[0];

if (!voting) {
    memory.setAttr("voting_result", "VOTING_NOT_FOUND");
    return;
}

// Дополнительная информация о голосовании
var project_name = table.find("projects", ["name"], [{"id", "=",
voting.project_id}])[0]?.name;
```

```

var subject_type_name = table.find("subject_types", ["name"], [{"code", "=",
voting.object_type}])[0]?.name;

memory.setAttr("project_name", project_name);
memory.setAttr("subject_type_name", subject_type_name);
memory.setAttr("subject_id", voting.object_id || "Не указано");
memory.setAttr("description", voting.description);
memory.setAttr("start_date", voting.start_date);
memory.setAttr("end_date", voting.end_date);
memory.setAttr("url", voting.url);

var voting_status_id = voting.status_id;
var voting_status = table.find("voting_statuses", ["name"], [{"id", "=",
voting_status_id}])[0]?.name;
memory.setAttr("voting_status", voting_status);

// Проверка участия пользователя в голосовании
var userVote = table.find("votes", ["id"], [{"voting_id", "=", entered_voting_id},
["member_id", "=", current_member_id]]);

if (userVote.length === 0) {
    memory.setAttr("voting_result", "ACCESS_DENIED");
    return;
}

// Формирование таблицы результатов голосования
var votes = table.find("votes", ["member_id", "vote_type_id", "comment"], [{"voting_id", "=",
entered_voting_id}]);
var votesTable = "Участник | Голос | Комментарий\n" +
    "-----\n";

votes.forEach(vote => {
    var member_name = table.find("members", ["first_name", "last_name"], [{"id", "=",
vote.member_id}])[0];
    var vote_type = table.find("vote_types", ["name"], [{"id", "=",
vote.vote_type_id}])[0]?.name;
    var formatted_vote_type = {
        "За": "    За",
        "Против": "    Против",
        "Воздержался": "    Воздержался",

```



```

        "Ожидается": "    Ожидается"
    }[vote_type] || "    Ожидается";

    votesTable += member_name.first_name + " " + member_name.last_name + " | " +
        formatted_vote_type + " | " +
        (vote.comment || "Нет комментария") + "\n";
});

memory.setAttr("voting_results_table", votesTable);

```

Этот код учитывает наличие записи пользователя в голосовании и, если она есть, предоставляет доступ к результатам голосования. В случае отсутствия записи или отсутствия самого голосования, код устанавливает соответствующие атрибуты в памяти для вывода соответствующих сообщений в сценарии.

Комментарий разработчика

Отлично! Код работает и написан как я хотел. Но, эта гадина потеряла нумерацию строк!

Prompt 3

Ты потерял нумерацию строк. Исправь - с нею лучше понятно сколько людей участвуют в голосовании.

ChatGPT

Понимаю, давайте добавим нумерацию строк в таблицу результатов голосования. Вот обновленный код:

```

var current_member_id = lead.getAttr("member_id");
var entered_voting_id = lead.getAttr("voting_id");

// Поиск голосования
var voting = table.find("votings", ["id", "status_id", "project_id",
    "object_type", "object_id", "description",
    "start_date", "end_date", "url"],
    [{"id", "=", entered_voting_id}][0];

if (!voting) {
    memory.setAttr("voting_result", "VOTING_NOT_FOUND");
    return;
}

```

```

// Дополнительная информация о голосовании
var project_name = table.find("projects", ["name"], [{"id", "=",
voting.project_id}][0]?.name;
var subject_type_name = table.find("subject_types", ["name"], [{"code", "=",
voting.object_type}][0]?.name;

memory.setAttr("project_name", project_name);
memory.setAttr("subject_type_name", subject_type_name);
memory.setAttr("subject_id", voting.object_id || "Не указано");
memory.setAttr("description", voting.description);
memory.setAttr("start_date", voting.start_date);
memory.setAttr("end_date", voting.end_date);
memory.setAttr("url", voting.url);

var voting_status_id = voting.status_id;
var voting_status = table.find("voting_statuses", ["name"], [{"id", "=",
voting_status_id}][0]?.name;
memory.setAttr("voting_status", voting_status);

// Проверка участия пользователя в голосовании
var userVote = table.find("votes", ["id"], [{"voting_id", "=", entered_voting_id],
["member_id", "=", current_member_id]}]);

if (userVote.length === 0) {
    memory.setAttr("voting_result", "ACCESS_DENIED");
    return;
}

// Формирование таблицы результатов голосования
var votes = table.find("votes", ["member_id", "vote_type_id", "comment"], [{"voting_id", "=",
entered_voting_id}]);
var votesTable = "# | Участник | Голос | Комментарий\n" +
    "-----\n";

let n = 1;
votes.forEach(vote => {
    var member_name = table.find("members", ["first_name", "last_name"], [{"id", "=",
vote.member_id}][0];
    var vote_type = table.find("vote_types", ["name"], [{"id", "=",

```

```

vote.vote_type_id]][0]?.name;
    var formatted_vote_type = {
        "За": "    За",
        "Против": "    Против",
        "Воздержался": "    Воздержался",
        "Ожидается": "    Ожидается"
    }[vote_type] || "    Ожидается";

    votesTable += n++ + " | " + member_name.first_name + " " + member_name.last_name + " | " +
        formatted_vote_type + " | " +
        (vote.comment || "Нет комментария") + "\n";
});

memory.setAttr("voting_results_table", votesTable);

```

Результат

🌟 Текущее Голосование в Metabot & Co.! 🌟

Мы рады видеть ваше участие в процессе принятия решений в рамках проекта Metabot & Co.. Вот ключевая информация о голосовании:

12
34

ID Голосования: 60

📁

Проект: 'Metabot & Co.'

📄

Тема: Другое/Не указано

📝

Описание: Цель

🔗

Ссылка на Детали: 🔗

📅

Дата Начала: "

📅

Дата Окончания: "

🔴

Статус голосования: Открыто

📊

Результаты:

#	Участник	Голос	Комментарий
1		Ожидается	Нет комментария
2		🕒 Ожидается	Нет комментария
3		Ожидается	Нет комментария
4		Ожидается	Нет комментария
5		Ожидается	Нет комментария
6		🕒 Ожидается	Нет комментария
7		Ожидается	Нет комментария
8		Ожидается	Нет комментария
9		🕒 Ожидается	Нет комментария

Ваше участие помогает формировать будущее проекта!
17:41

Даты по прежнему не показываются, но это не ошибка. В версии приложения, представленной в этом кейсе, этот функционал еще не был реализован.

Отзыв разработчика о работе с CoPilot на платформе Metabot:

ChatGPT успешно справился! Теперь чат-бот присылал карточку со всей информацией, как я и задумывал.

Я не измерял специально, но по моим субъективным ощущениям, **он сэконобил мне, как минимум, пол часа времени!** А если бы я допустил какие-то ошибки в коде и занялся бы их вылавливанием, что иногда случается, то еще больше.

Единственная правка, которую я сделал, это пятую строку кода я разбил на отдельные строки, чтобы она вмещалась в экран, но это ерунда.

Главное, что он смог, как умелый разработчик сделать реинжиниринг нужного мне куска алгоритма. Мне кажется эта штука изменит подход к разработке. Если раньше мы все делали своими руками, то теперь у меня в распоряжении идеальный раб=) Я продумываю логику и результат, затем говорю, что хочу получить, и вуаля - он дает мне это через секунду.

Мне остается только очень хорошо и системно продумывать продукт, а он сделает всю черновую работу по его созданию, и, как мы убедились на этом примере, сможет его модифицировать.

А еще мне нравится программисткий Low-Code подход на платформе Metabot. Он хорошо подходит для разработки в паре с ИИ. На других конструкторах ботов, особенно с визуальными редакторами, мне приходится извращаться, чтобы сформировать промпт, а в Metabot я просто добавляю к промпту содержимое экрана из Metabot со всеми сообщениями и кодом, и нейронка понимает мою задачу.

Выводы и рекомендации

Ключевое осознание, которое произошло в ходе применения ChatGPT в разработке кода, заключается в том, что ИИ вполне заменяет Junior-программиста. И даже его превосходит - ИИ знает больше и умеет больше чем юниор! И ошибается реже и учится быстрее. А главное, что он не устает и всегда доступен, мгновенно решая задачи. В то время как юниоры уходят на несколько часов, потом приносят какую-то кривую штуку с ошибками и все повторяется по кругу.

Проблема юниоров в том, что они не знают весь процесс и у них нет опыта. У GPT тоже нет знания вашего процесса, но зато есть опыт всего человечества, у него есть все знания,

чтобы выполнить поставленную задачу, причем сделать это почти мгновенно, при условии, что вы можете поставить задачу и задать весь нужный контекст, чтобы он решил его так как вы этого хотите!

Итак, без сомнения, ИИ изменит подход к разработке, выступив в роли junior-программиста, эффективно справляющегося с рутинными задачами более опытного коллеги-разработчика/человека-оператора ИИ. Главная ценность GPT раскрывается, когда за его "рулем" находится опытный разработчик. Такой оператор ИИ хорошо осведомлен о своих целях и может четко и ясно передать эти задачи машине. ИИ забирает на себя выполнение стандартных операций, освобождая время разработчика для более глубокой, творческой и системной работы.

Практические рекомендации

1. Понимание Роли GPT/ИИ:

- Рассматривайте ИИ как инструмент в руках скульптора, где вы - скульптор, а ИИ - ваше средство для ваяния кода по вашему замыслу.
- Воспринимайте ИИ не как замену разработчика, а как junior-разработчика для работы в паре, готового мгновенно и беспрекословно выполнить любую задачу.
- Это инструмент, который может исполнять задачи по вашему указанию, но не способен самостоятельно создавать или развивать сложные концепции.
- Воспринимайте ИИ как дополнение к вашим навыкам, а не как полную замену. Он может забрать на себя выполнение рутинной работы, позволяя вам сосредоточиться на более творческих и стратегических аспектах проекта.
- Используйте ИИ для выполнения повторяющихся и времязатратных операций, чтобы сосредоточиться на более важных аспектах разработки. Например, реинжиниринг куска кода, добавление комментариев, написание документации.

2. Комбинирование Low-Code и ИИ:

- Использование Low-Code вместе с ИИ позволяет многократно повышать продуктивность, снижая время, необходимое для разработки и реализации сложных функций.

3. Четкое Задание Задач:

- Удостоверьтесь, что ваши инструкции для ИИ четкие и точные. Это уменьшит количество ошибок и повысит качество выполнения задач.

4. Необходимость Навыков Программирования:

- Для эффективного использования ИИ необходимы базовые знания в области программирования, создания архитектуры и системного мышления. Это позволит вам точно формулировать задачи и корректировать результаты работы ИИ.

5. Принятие Активной Роли в Процессе:

- Помните, что ИИ не заменяет разработчика, а действует как его помощник. Будьте готовы вмешаться и вносить коррективы, когда это необходимо.
- Вы должны активно участвовать в процессе работы с ИИ, особенно когда речь идет о сложных и нестандартных задачах.

6. Использование Средств Сравнения Кода:

- Применяйте программы типа [WinMerge](#) для сравнения сгенерированного GPT кода с текущим. Это позволит вам быстро выявлять различия и упростит процесс дебаггинга.

7. Тестирование и Проверка Кода:

- Тщательно тестируйте сгенерированный код. Всегда проверяйте его на предмет ошибок и совместимости с вашими текущими системами.
- Всегда тщательно тестируйте код, полученный от ИИ, прежде чем использовать его в реальных проектах. Проверяйте на ошибки и убедитесь, что он соответствует вашим требованиям.

8. Обработка Ошибок:

- Экономьте время на отладке и выявлении ошибок. Если в процессе работы возникают ошибки, используйте возможности ИИ для их быстрого исправления. Это может включать в себя отправку информации об ошибках и соответствующем контексте для эффективного решения проблемы.

Заключение

Использование ИИ в разработке - это не просто автоматизация, это новый уровень сотрудничества между человеком и машиной, где ключевую роль играет способность разработчика правильно направлять ИИ.

Это возможность значительно ускорить разработку, передавая ИИ рутинные задачи и фокусируясь на более сложных и творческих аспектах проекта.

ИИ становится незаменимым инструментом в арсенале современного разработчика, позволяя ему максимально раскрыть свой творческий и интеллектуальный потенциал.

Удачи в освоении Low-Code в паре с ИИ!

Версия #4

Artem Garashko создал 28 January 2024 14:49:40

Artem Garashko обновил 5 February 2024 09:23:36