

Программа для ЭВМ “ОптимальСити. Чат-платформа”

Руководство пользователя

Оглавление

1. Общее описание.....	4
2. Глоссарий.....	4
3. Работа с системой.....	7
3.1 Вход в систему.....	7
4. Раздел “Мои проекты”	8
4.1 Создание карточки сценария чат-бота.....	8
4.2 Создание сценария чат-бота.....	12
4.2.1 Меню общих настроек сценария.....	14
4.2.2 Управление сценариями и подсценариями через меню Общих настроек.....	17
4.2.3 Функциональный блок “Сказать”.....	18
4.2.4 Функциональный блок “Ожидание ввода”.....	18
4.2.5 Функциональный блок “Переменные”.....	19
4.2.6 Функциональный блок “Проверка”.....	21
4.2.7 Функции для проверки.....	24
~string_in – функция для проверки вхождения подстроки в строку..	24
~string_match – функция проверки на соответствие подстроки заданному шаблону.....	25
~array_get – функция проверки вхождения подстроки в строку элемента массива.....	25
~array_size – функция проверки на количество элементов.....	26
4.2.8 Функциональный блок “Функции”.....	26
~str_replace – функция производит замену в строке на подстроку....	27
~arr_explode – функция, которая помогает разбить строку на массив	

по регулярному выражению (regex).....	27
~array_explode – функция, которая помогает разбить строку на массив по регулярному выражению синтаксиса Lua.....	28
~array_join – функция, которая помогает объединить массив в строку с разделителем.....	29
~string_digits – функция, которая определяет количество цифр в строке или переменной.....	30
~string_len – функция, которая определяет длину строки.....	31
~array_set – функция, которая присваивает переменной индекс в массиве.....	31
~array_get – функция, которая получает значение элемента массива.....	32
~array_size – функция, которая показывает длину массива.....	33
4.2.9 Функциональный блок “Запрос”.....	33
4.2.10 Функциональный блок “Email”.....	40
4.2.11 Функциональный блок “Текущее время”.....	42
4.2.12 Функциональный блок “Подсценарий”.....	44
4.2.13 Соединения: копирование, вставка, переходы.....	46
5. NLU.....	50

1. Общее описание

Веб-приложение “ОптималСити. Чат-платформа” – визуальный конструктор сценариев для работы чат-ботов. Low-code подход, лежащий в основе системы, упрощает создание чат-ботов, позволяет проводить тонкую настройку сервисов.

В настоящем руководстве описаны функции системы, предназначенные для использования пользователем системы.

2. Глоссарий

Диалог-дизайнер	визуальный конструктор сценариев для работы чат-ботов
Функциональные (логические) блоки	набор элементов визуального конструктора, обладающих функциями
Регулярные выражения	регулярные выражения (их еще называют regex, или regeх) — это механизм для поиска и замены текста
Авторизация	предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки данных прав при попытке выполнения этих действий
JSON	JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript
API	программный интерфейс, то есть описание способов взаимодействия одной компьютерной

	программы с другими
Экспорт	это автоматический или полуавтоматический вывод наборов данных между различными программными приложениями
NLU	отдельный веб-сервер с ИИ-моделью классификации. Производит смысловой поиск по базе знаний и выдачу ответов с процентом уверенности

“ОптимальСити. Чат-платформа” позволяет:

Платформа для создания чат-ботов “ОптимальСити. Чат-платформа” предоставляет пользователю следующие функциональные возможности:

- создание сценариев в визуальном конструкторе для функционирования чат-ботов;
- отладка и тестирование чат-ботов;
- интеграция с телеграм;
- виджет.

Перечень основных функциональных характеристики системы “ОптимальСити. Чат-платформа” включает:

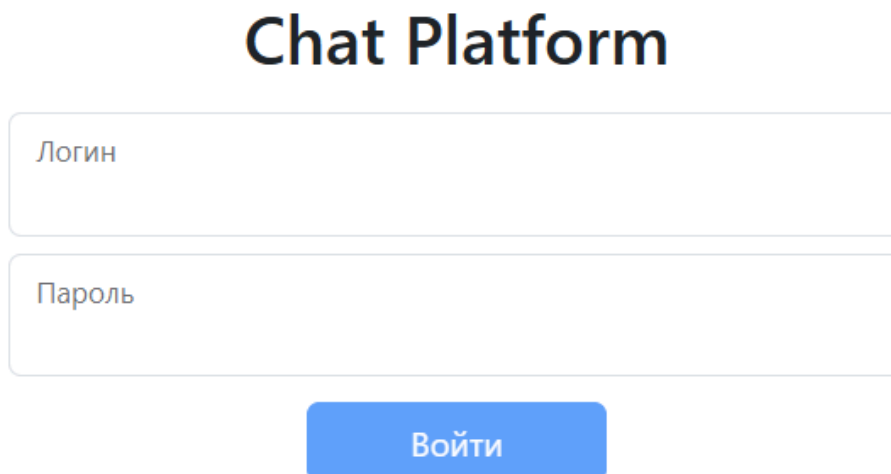
1. создание и редактирование сценариев для работы чат-ботов:
 - создание веток сценария чат-бота через редактор (диалог-дизайнер) с использованием функциональных блоков;
 - копирование и вставка функциональных блоков сценария и их соединений;
 - создание подсценариев;
 - распознавание текста, введённого пользователем, и дальнейшее направление по веткам сценария с помощью функциональных блоков;

- осуществление http-запросов с помощью функциональных блоков;
 - возможность интеграции с NLU.
2. сохранение и загрузки пользовательских сценариев в интерфейс конструктора сценариев:
 - сохранение пользовательских сценариев в формате JSON;
 - загрузка пользовательских сценариев в формате JSON.
 3. поиск по сохранённым пользовательским сценариям.
 4. управление учётными записями пользователей.

3. Работа с системой

3.1 Вход в систему

“ОптималСити. Чат-платформа” представляет собой веб-приложение. Вся работа с системой ведётся в веб-браузере. После ввода адреса сервера в адресную строку браузера отобразится окно авторизации (рис. 1).



The image shows a login interface for the 'Chat Platform'. At the top, the title 'Chat Platform' is displayed in a large, bold, black font. Below the title, there are two input fields: the first is labeled 'Логин' (Login) and the second is labeled 'Пароль' (Password). Both fields are empty and have a light gray border. Below these fields is a blue button with the text 'Войти' (Login) in white. The entire form is centered on a white background.

рис. 1

Пользователи могут быть зарегистрированы в системе по принципу “Внутренней регистрации”.

Введите логин и пароль и нажмите на кнопку **“Войти”**. В случае, если вы забыли логин или пароль, обратитесь к Администратору.

После успешного входа в систему пользователю отобразится рабочий стол. В левой части рабочего стола отображается панель с основными разделами Платформы:

- “Мои проекты”.

4. Раздел “Мои проекты”

Раздел “Мои проекты” (рис. 2) предоставляет пользователю следующие функциональные возможности:

- выбор проекта;
- поиск по проектам;
- просмотр детальной информации о проекте.

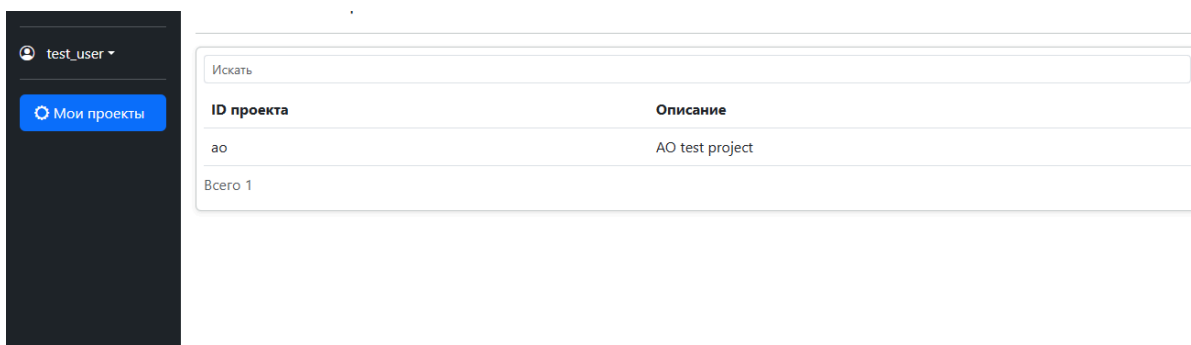


рис. 2

4.1 Создание карточки сценария чат-бота

Для того чтобы создать сценарий чат-бота необходимо нажать на кнопку “Создать сценарий” во вкладке выбранного проекта (рис. 3).

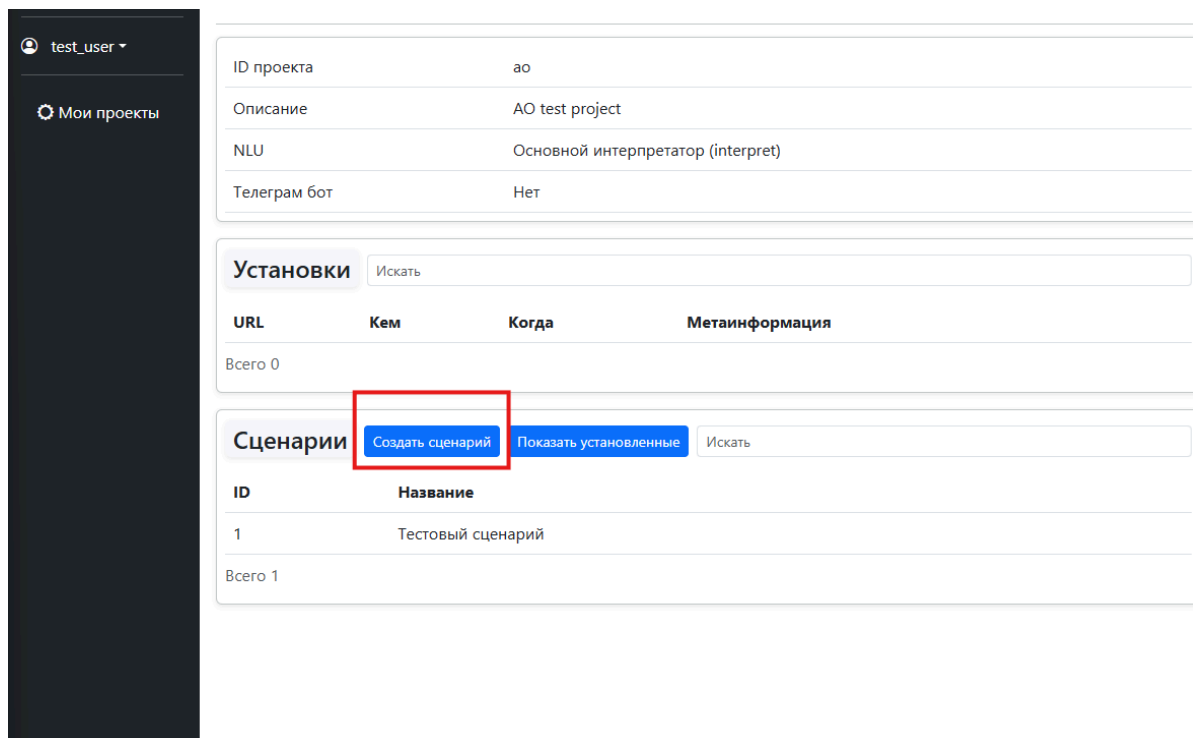


рис. 3

После нажатия на кнопку **“Создать сценарий”** открывается карточка нового сценария. Новому сценарию автоматически присваивается название **“Новый”**.

На экране вы можете увидеть редактируемое поле для названия сценария, а также редактируемое поле для комментариев (рис. 4). Для создания сценария введите название своего сценария и комментарий (по желанию). Затем нажмите на кнопку **“Ок”**. При нажатии на кнопку **“Отменить”** карточка сценария не сохранится.

Если вы хотите загрузить готовый сценарий в формате JSON, нажмите на кнопку **“Загрузить json”**. В открывшемся окне выберите файл, и подтвердите загрузку.

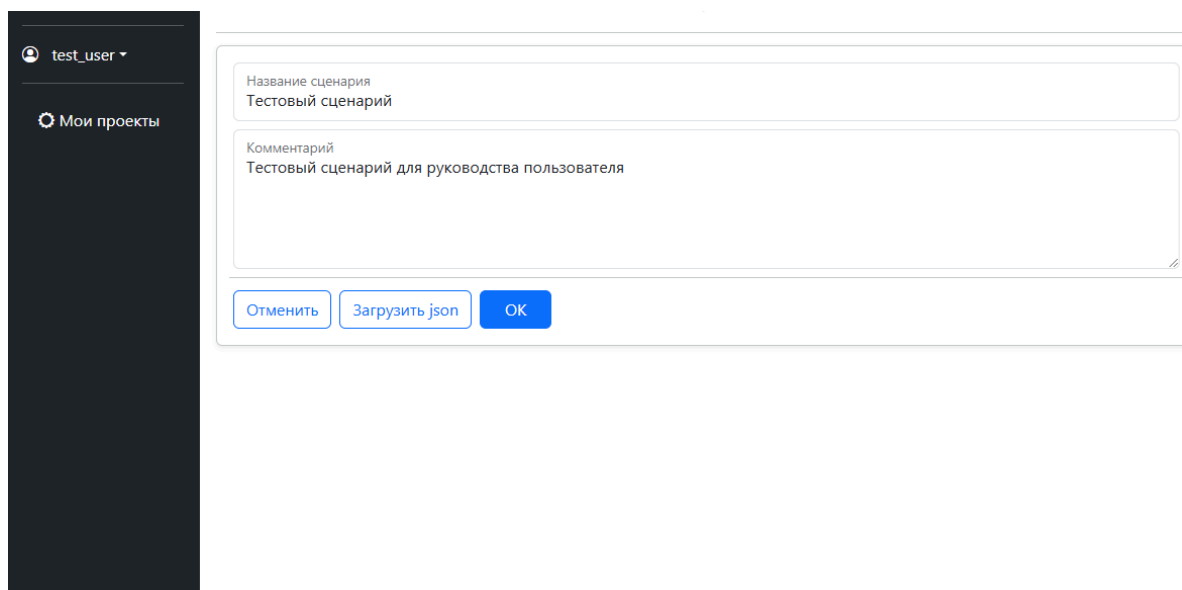


рис. 4

После нажатия на кнопку **“Ок”** карточка сценария сохранится и редактируемые поля станут серыми. Редактировать поля **“Название”** и **“Комментарий”** можно, нажав кнопку **“Редактировать”** (рис. 5).

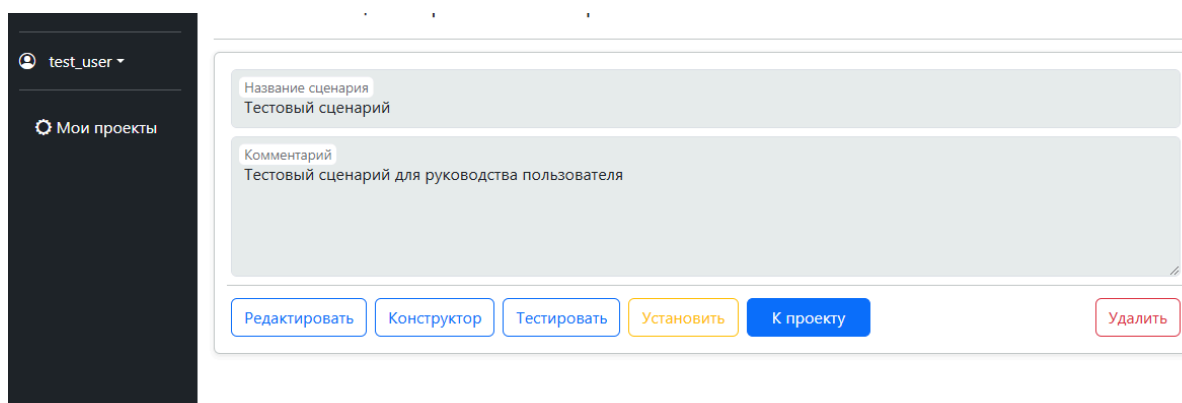


рис. 5

Кнопка **“К проекту”** возвращает на страницу карточки проекта.

Кнопка **“Конструктор”** открывает страницу с диалог-дизайнером для создания или редактирования сценария чат-бота (подробнее см. п. 3.2).

Кнопка **“Тестировать”** позволяет пользователю установить сценарий в тестовом режиме и проверить работоспособность чат-бота. После нажатия на кнопку открывается новая вкладка с чат-ботом, установленным в тестовом режиме (рис. 6).

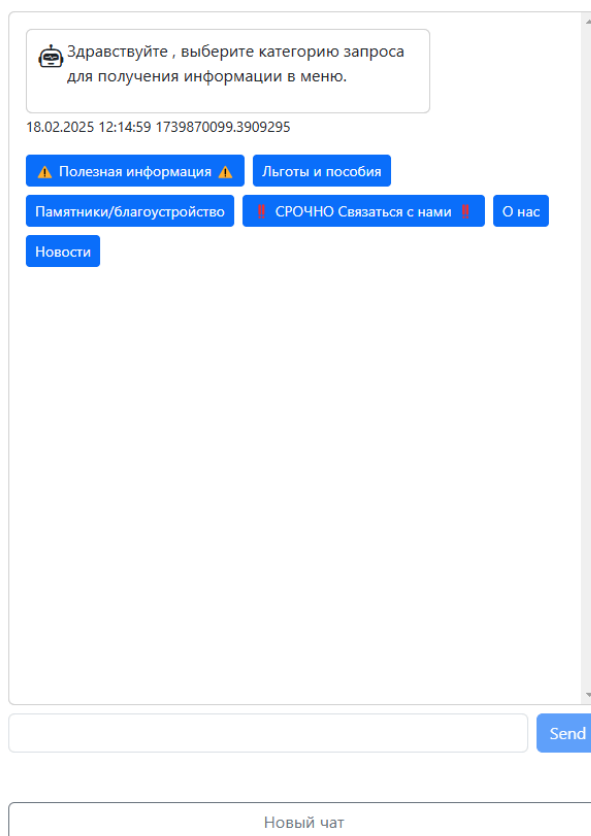


рис. 6

Кнопка **“Установить”** позволяет пользователю установить сценарий в продакшн.

Кнопка **“Удалить”** позволяет пользователю удалить сценарий.

4.2 Создание сценария чат-бота

В данном разделе подробно рассмотрено создание сценария чат-бота, а также функциональные возможности диалог-дизайнера.

После нажатия на кнопку **“Конструктор”** пользователю открывается рабочий стол диалог-дизайнера.

В левой части рабочего стола можно видеть набор функциональных блоков (рис. 7):

- “Сказать”;
- “Ожидание ввода”;
- “Пауза”;
- “Переменные”;
- “Проверка”
- “Функции”;
- “Перенаправить”;
- “Запрос”;
- “Email”;
- “Текущее время”;
- “Подсценарий”.

@Ответ (основной) < Тестовый сценарий

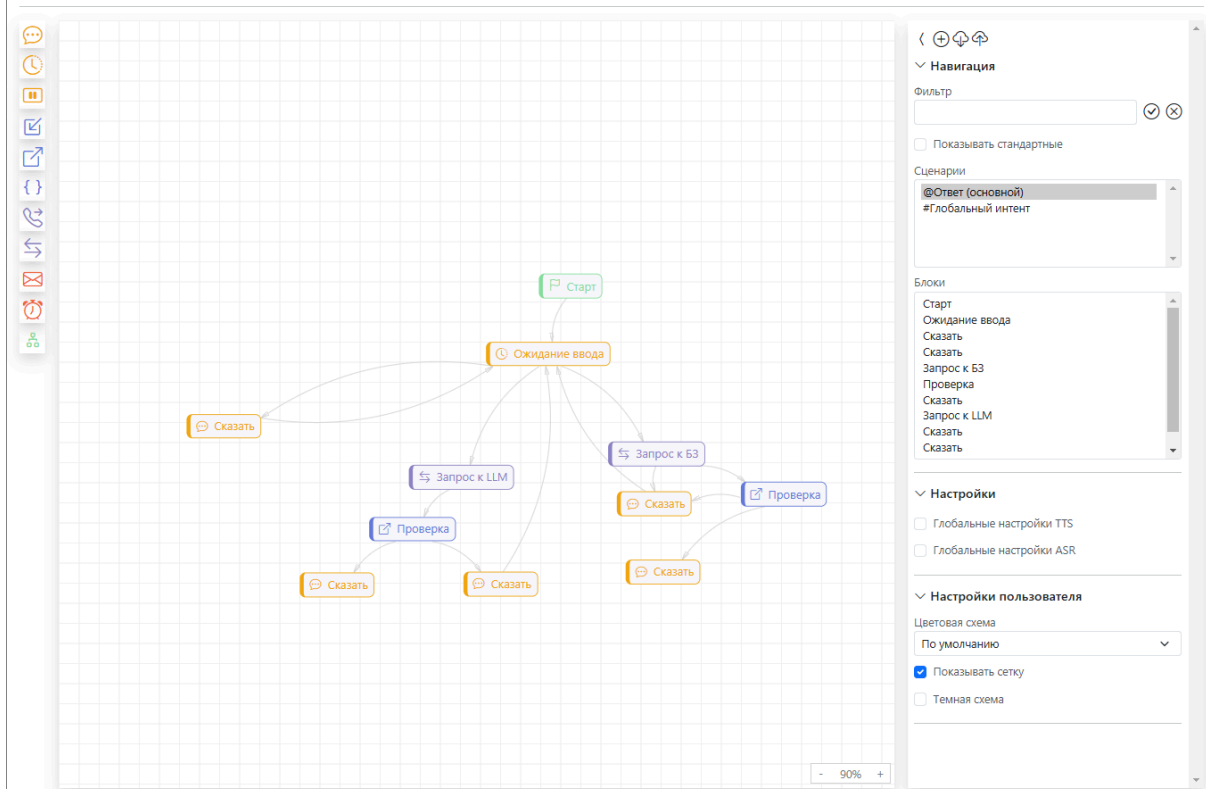


рис. 7

Блоки можно перетаскивать на поле при помощи мыши.

В правой части рабочего стола видны Общие настройки сценария или настройки блока в случае, если пользователь выбирает конкретный блок.

4.2.1 Меню общих настроек сценария

При нажатии на пустое рабочее пространство открывается меню общих настроек (рис. 8):

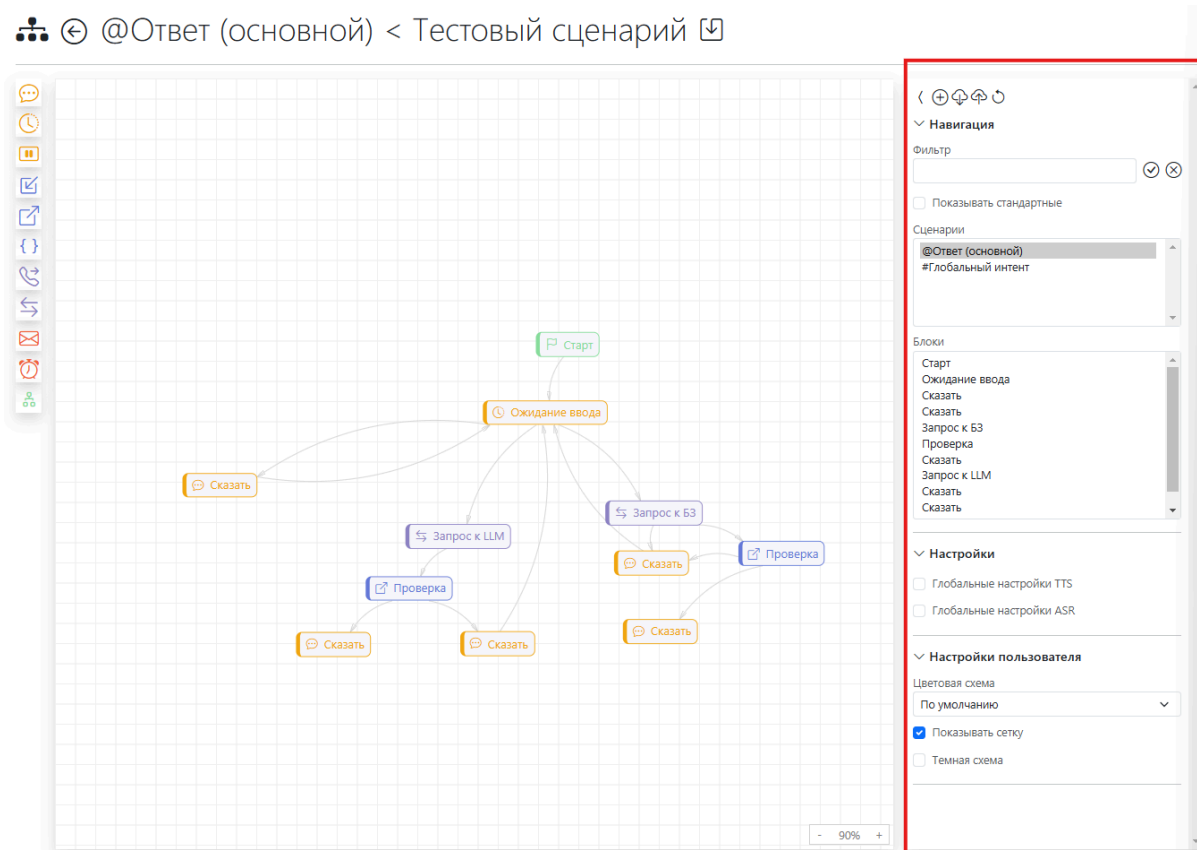


рис. 8

- В верхней части меню расположена панель действий со сценарием (рис. 8). Пользователь может создать новый подсценарий, загрузить сценарий, экспортировать сценарий или отменить последнее изменение.
- Также пользователь может осуществлять поиск по функциональным блокам, воспользовавшись строкой “Фильтр” в разделе “Навигация” (рис. 9).

🏠 ⏪ @Ответ (основной) < Тестовый сценарий 📄

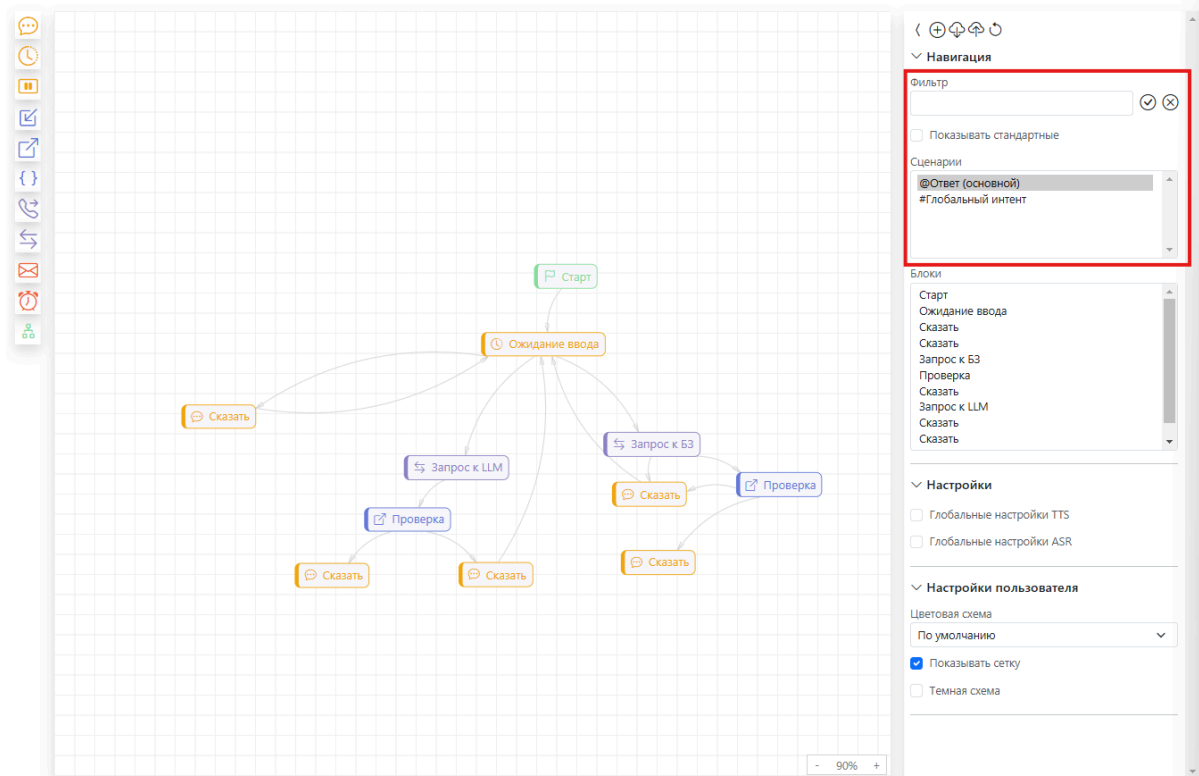


рис. 9

- В разделе **“Настройки пользователя”** можно выбрать цветовую схему (цветная или монохромная), указать видимость сетки и формат темы (тёмная или светлая) (рис. 10).

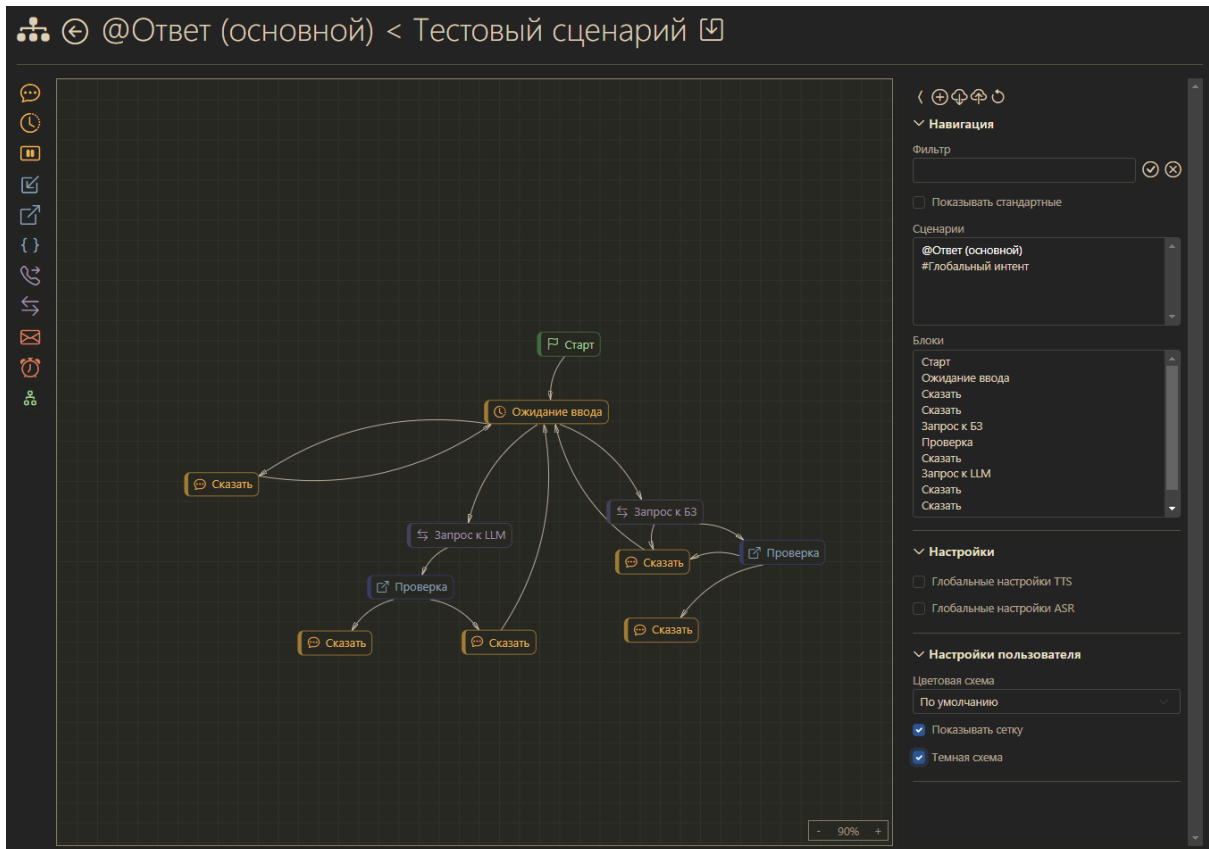


рис. 10

4.2.2 Управление сценариями и подсценариями через меню **Общих настроек**

Диалог-дизайнер позволяет пользователю управлять “стандартными” сценариями и созданными пользователем подсценариями через меню общих настроек (рис. 11).

☰ ⌕ @Ответ (основной) < Тестовый сценарий ↗

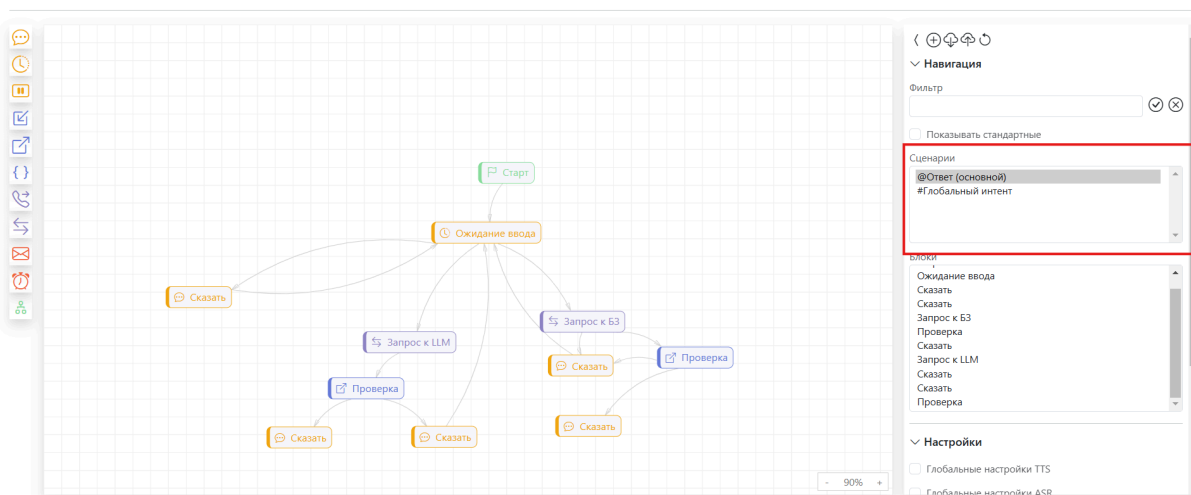


рис. 11

Для создания нового подсценария внутри сценария необходимо нажать на кнопку создания. Создастся новый подсценарий со стандартным названием (рис. 12).

☰ ⌕ Сценарий 1 < Тестовый сценарий ↗

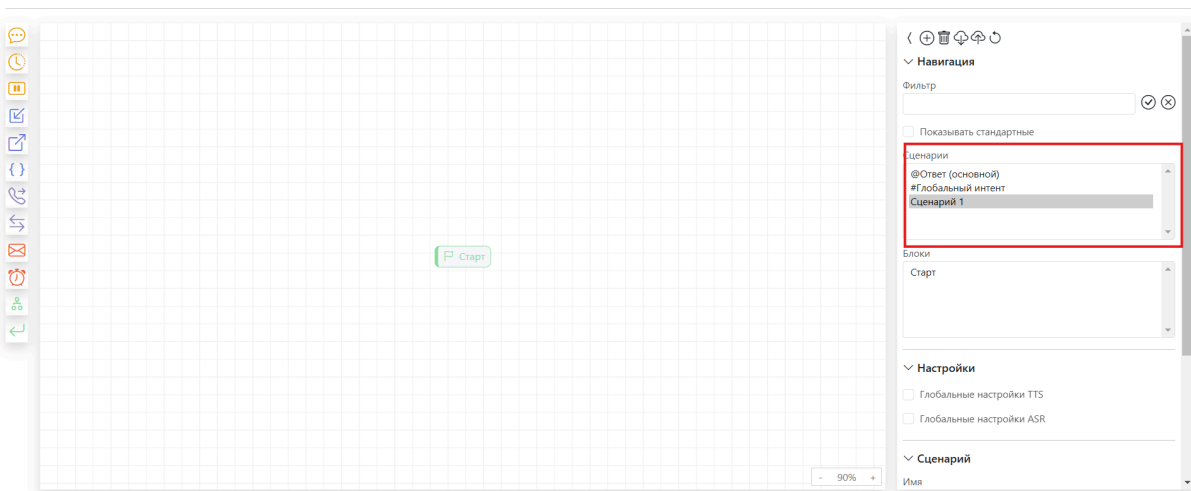


рис. 12

4.2.3 Функциональный блок “Сказать”

Блок “Сказать” предоставляет пользователю возможность вывести сообщения для чат-бота.

При добавлении на рабочее пространство и нажатии на блок “Сказать” открывается окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 13):

☰ @Ответ (основной) < Тестовый сценарий 📄

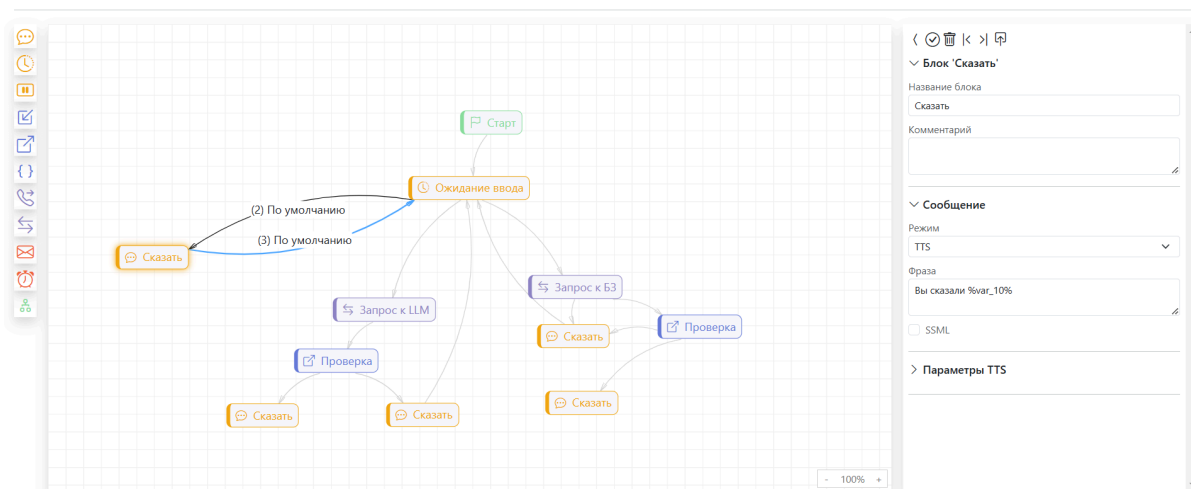


рис. 13

Важно! Если вам нужно, чтобы чат-бот передал данные из переменной, то переменную можно вызвать прямо в поле “Фраза”, заключив её в знак “%”.

4.2.4 Функциональный блок “Ожидание ввода”

Функциональный блок “Ожидание ввода” предоставляет пользователю возможность добавить кнопки в чат-бота.

При добавлении на рабочее пространство и нажатии на функциональный блок “Ожидание ввода” откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 14).

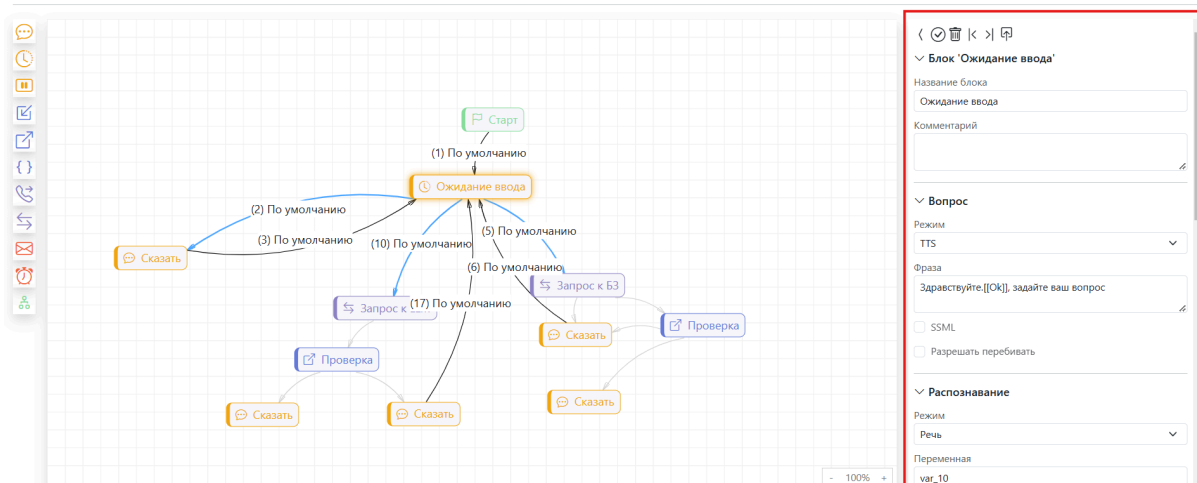


рис. 14

1. Раздел **“Блок “Ожидание ввода”** содержит:
 - Редактируемое поле **“Название блока”** – поле для изменения названия блока;
 - Редактируемое поле **“Комментарий”** – поле для комментария к блоку.
2. Раздел **“Вопрос”** содержит:
 - Редактируемое поле **“Фраза”** – в это поле можно ввести фразу для чат-бота, а также кнопку, заключив текст кнопки в квадратные скобки.

4.2.5 Функциональный блок **“Переменные”**

Функциональный блок **“Переменные”** предоставляет пользователю возможность добавления, сравнения, перезаписывания переменных.

При добавлении на рабочее пространство и нажатии на функциональный блок **“Переменные”** откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 15).

Важно! Для того чтобы вызвать переменные в других блоках необходимо заключить название переменной в знак “%”. Например, %var_10%.

Важно! Переменные, используемые в сценарии, необходимо объявить в начале сценария, используя блок “Переменные”. Например, если вы хотите, чтобы робот запрашивал оценку у пользователя и сохранял её, необходимо ввести переменную (например, grade_1), а затем записать её в соответствующее поле блока “Ожидание ввода”. В таком случае оценка запишется в переменную и сохранится. Если вы внесёте данную переменную в последующие блоки, то она перезапишется.

🏠 @Ответ (основной) < Руководство пользователя 📖

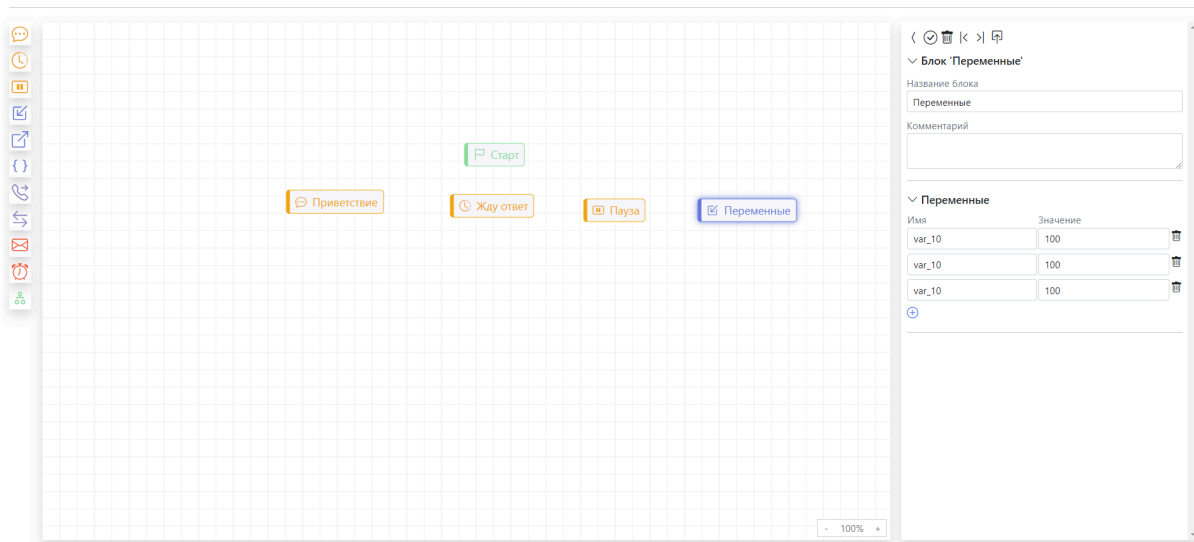


рис. 15

1. Раздел “**Блок “Переменные”**” содержит:
 - Редактируемое поле “Название блока” (поле для изменения названия блока);
 - Редактируемое поле “Комментарий” (поле для комментария к блоку).
2. Раздел “**Переменные**” содержит:
 - Кнопку добавления переменной (рис. 16)

Важно! При создании переменная получает стандартное название “var_10” и стандартное значение “100”. Название переменной и её значение можно менять.

🔗 @Ответ (основной) < Руководство пользователя 📖

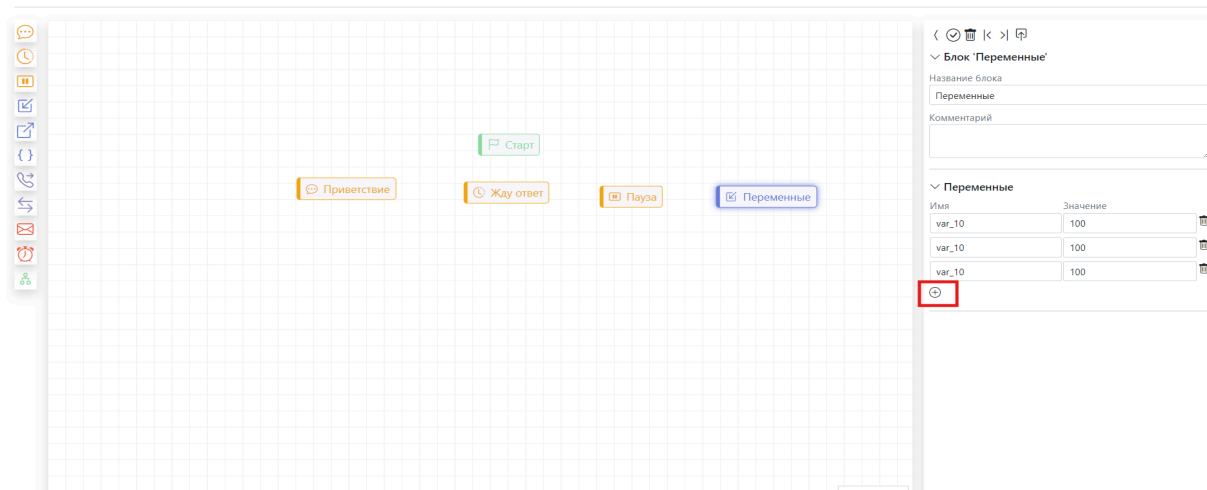


рис. 16

4.2.6 Функциональный блок “Проверка”

Для проверки полученных значений переменных в диалога пользователя с чат-ботом можно воспользоваться функциональным блоком “Проверка“. С его помощью можно задать определённые условия, выполнение которых будет определять дальнейшее действие чат-бота.

При добавлении на рабочее пространство и нажатии на функциональный блок “Переменные” откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 17).

@Ответ (основной) < Руководство пользователя

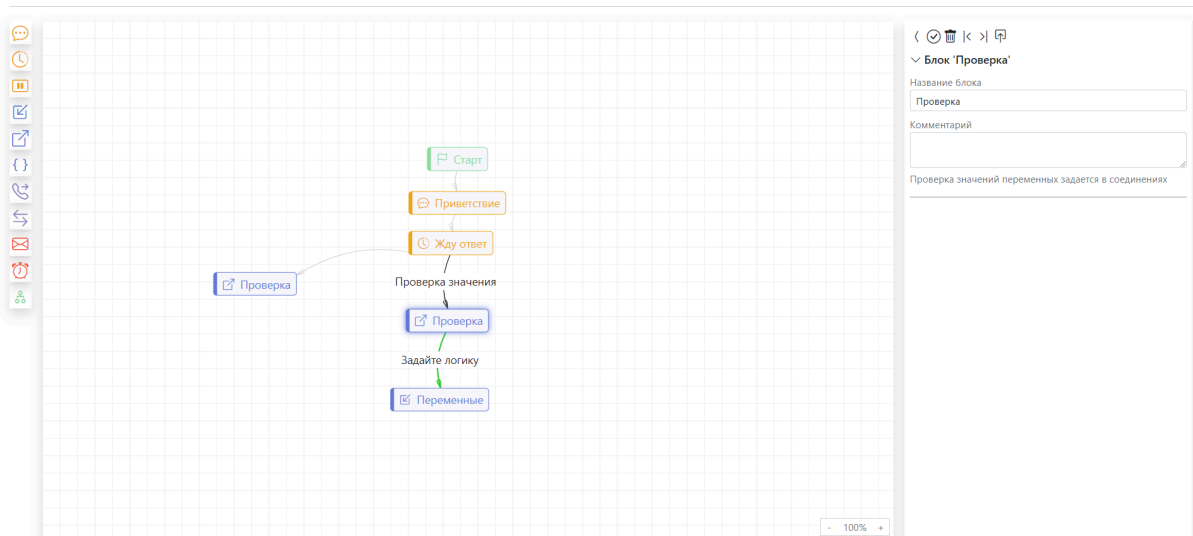


рис. 17

1. Раздел “Блок “Проверка” содержит:
 - Редактируемое поле “Название блока” (поле для изменения названия блока);
 - Редактируемое поле “Комментарий” (поле для комментария к блоку).

Важно! Проверка значений переменных задаётся в соединениях.

Блок “Проверка” можно соединить с другими блоками (рис. 18) и установить условие, выбранное из выпадающего списка (рис. 19).

@Ответ (основной) < Руководство пользователя

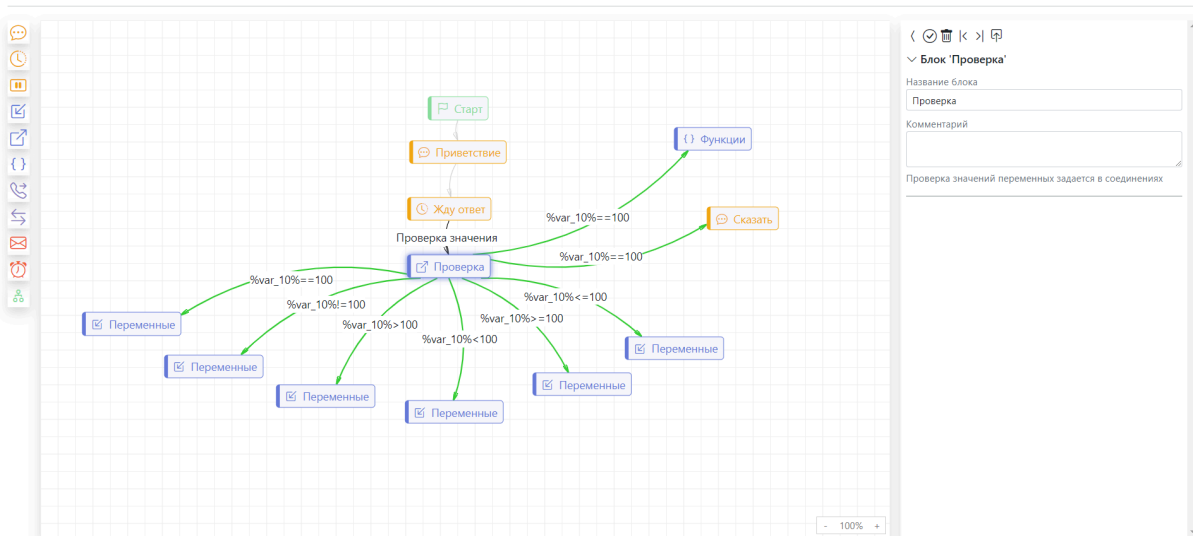


рис. 18

☰ @Ответ (основной) < Руководство пользователя 📄

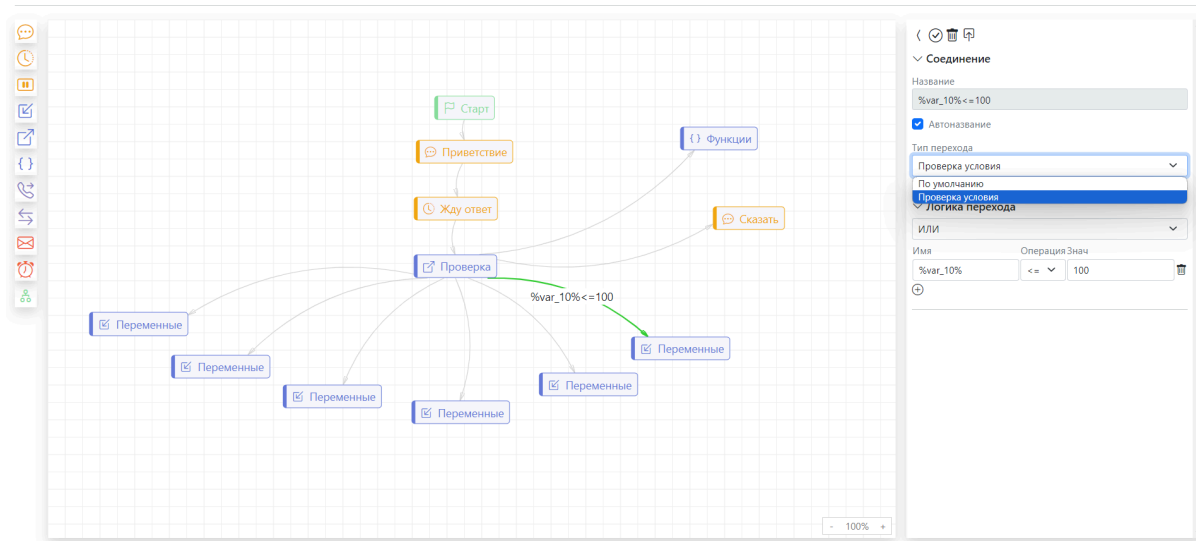


рис. 19

Затем необходимо выбрать логику перехода из выпадающего списка в разделе “Логика перехода”, вызвать переменную и выбрать подходящий оператор сравнения (рис. 20).

☰ @Ответ (основной) < Руководство пользователя 📄

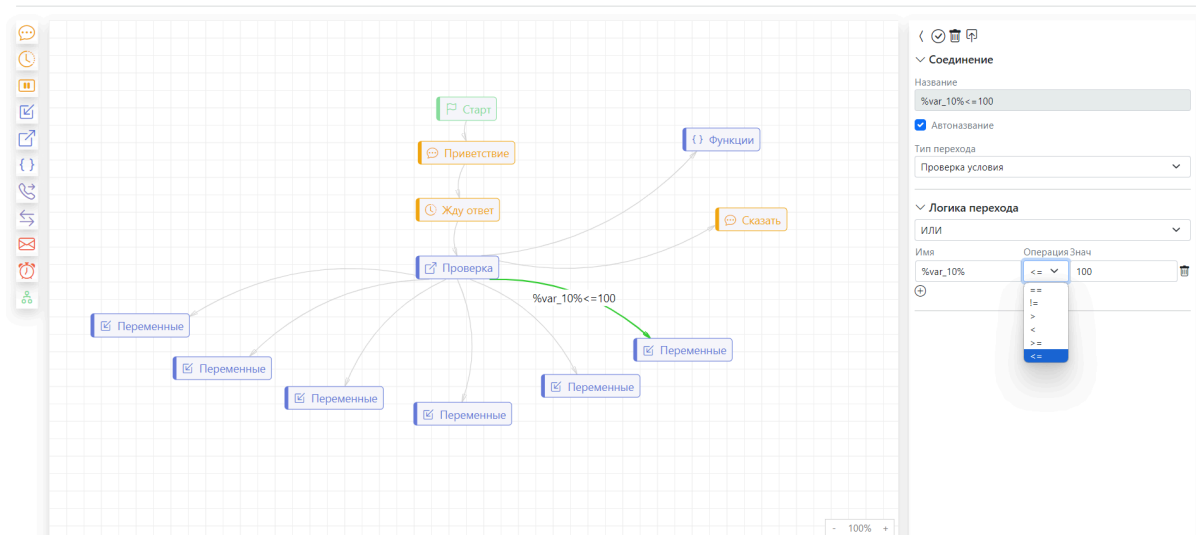


рис. 20

Важно! “ОптимальСити. Чат-платформа” поддерживает 6 стандартных типов операций:

- 1) Равно ==;
- 2) Неравно !=;
- 3) Больше >;

- 4) Меньше \leq ;
- 5) Больше или равно \geq ;
- 6) Меньше или равно \leq .

Если используются операторы $==$ или $!=$, то сравниваются строки. Если используются операторы $>$, $<$, \geq , \leq , то сравниваются только численные значения. При любых других значениях, включая пустые переменные и отсутствующие переменные, равенства/неравенства не выполняются.

4.2.7 Функции для проверки

`~string_in` – функция для проверки вхождения подстроки в строку

Левое значение	Тип условия	Правое значение
<code>~string_in(%var_1%)</code>	<code>==</code>	<code>%var_2%</code>

Данный функционал позволит обрабатывать случаи, когда пользователь использует такие фразы, как "да нет". Чат-бот может интерпретировать письменную речь пользователя как "Да", в то время как пользователем подразумевался ответ "Нет".

С помощью функции **`string_in`**, полученный ответ пользователя может быть проверен на предмет вхождения в строку подстроки "Нет".

И если находится вхождение – система может перенаправить пользователя в ветку, предусматривающую ответ "Нет".

~string_match – функция проверки на соответствие подстроки заданному шаблону

Левое значение	Тип условия	Правое значение
~string_match("<шаблон>")	==	%var_1%

Функция **~string_match** позволяет проверить, входит ли в заданную строку значение, соответствующее регулярному выражению

<шаблон> – строка-образец, задающая правило поиска в виде регулярного выражения.

var_1– переменная, которая содержит строку.

Также возможен такой вид функции:

Левое значение	Тип условия	Правое значение
~str_match("%var_4% % %var_3%")	==	%var_4% % %var_3%

%var_4% % %var_3% – регулярное выражение, в котором вызваны переменные, разделенные знаком "%".

"%var_4% % %var_3%" – шаблон предполагаемого получаемого результата, в котором переменные var_4 и var_3 разделены знаком "%".

Также шаблон выражения может заключаться в кавычки.

~array_get – функция проверки вхождения подстроки в строку элемента массива

Левое значение	Тип условия	Правое значение
<code>~array_get(<имя элемента массива>)</code>	<code>==</code>	<code>%var_1%</code>

Данная функция позволит проверить, входит ли подстрока в строку, являющуюся значением элемента массива.

<имя элемента массива> – указывается в виде `array.0` (имя элемента массива.индекс элемента)

var_1 – переменная содержит строку.

~array_size – функция проверки на количество элементов

Левое значение	Тип условия	Правое значение
<code>~array_size(array.%join_id%)</code>	<code>==</code>	<code>%var_1%</code>

array.%join_id% – имя массива, задаваемое в виде шаблона (`array.%join_id%`), элементы которого необходимо объединить в строку (данной переменной должно быть присвоено значение в отдельном блоке и заранее).

var_1 – переменная содержит численное значение.

4.2.8 Функциональный блок “Функции”

Данный блок позволяет использовать функции чат-платформы с уже готовыми паттернами.

При добавлении на рабочее пространство и нажатии на функциональный блок “Функции” откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 21).

У блока есть 9 функций, рассмотрим их подробнее.

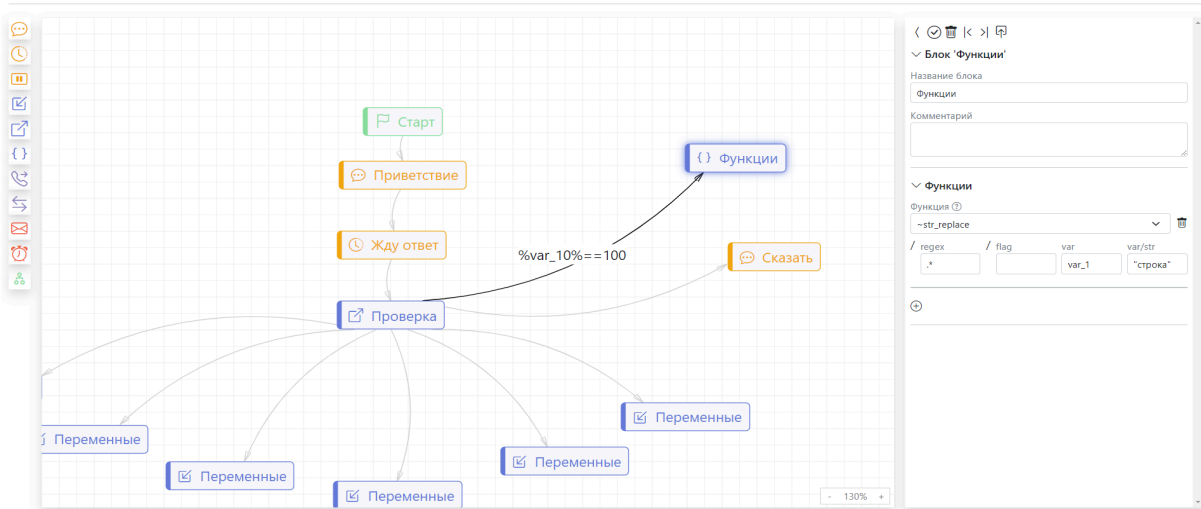


рис. 21

~str_replace – функция производит замену в строке на подстроку

- в поле “**regex**” указывается регулярное выражение (без кавычек и слешей);
- в поле “**flag**” указывается один или несколько флагов;
- в поле “**var**” указывается переменная, в которой осуществляется замена;
- в поле “**var/str**” указывается переменная или строка, которая будет вставлена в переменную, где происходит замена (рис. 29).

~arr_explode – функция, которая помогает разбить строку на массив по регулярному выражению (regex)

- в поле “**regex**” указывается регулярное выражение (без кавычек и слешей);
- в поле “**flag**” указывается один или несколько флагов;
- в поле “**var**” указывается переменная, в которой осуществляется замена;

- в поле “**explode_id**” указывается итератор, который собирает элементы массива (необходимо задать до функции) (рис. 22).

☰ @Ответ (основной) < Руководство пользователя 📄

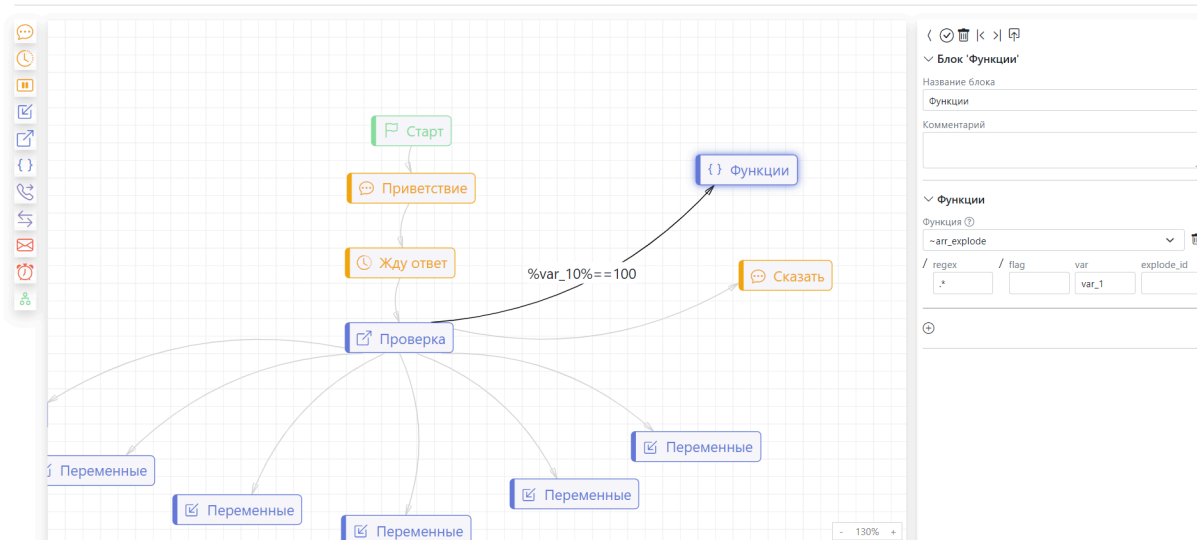


рис. 22

~array_explode – функция, которая помогает разбить строку на массив по регулярному выражению синтаксиса Lua

- в поле “**lua**” указывается регулярное выражение на языке Lua, записывается без кавычек и слешей;
- в поле “**var**” указывается переменная, в которой осуществляется замена;
- в поле “**explode_id**” указывается итератор, который собирает элементы массива (задается до функции) (рис. 23).

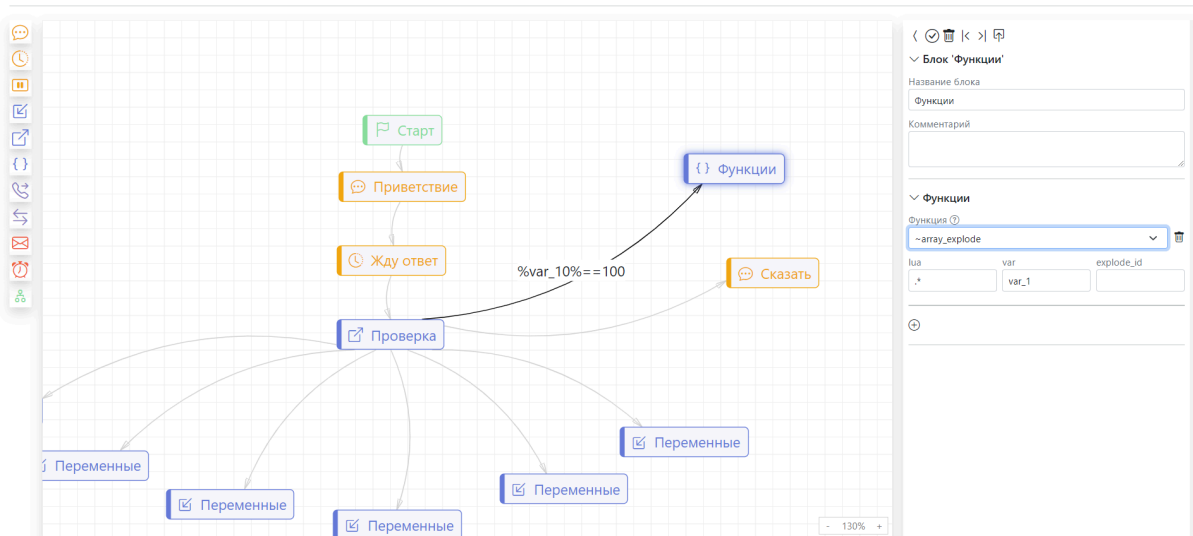


рис. 23

~array_join – функция, которая помогает объединить массив в строку с разделителем

- в поле “**sep**” указывается сепаратор (разделитель) – символ, которым будут разделены элементы массива в строке;
- в поле “**join_id**” указывается итератор, который собирает элементы массива в строку (важно задать переменную до функции)
- в поле “**var**” указывается переменная, куда запишется строка (рис. 24).

☰ @Ответ (основной) < Руководство пользователя 📄

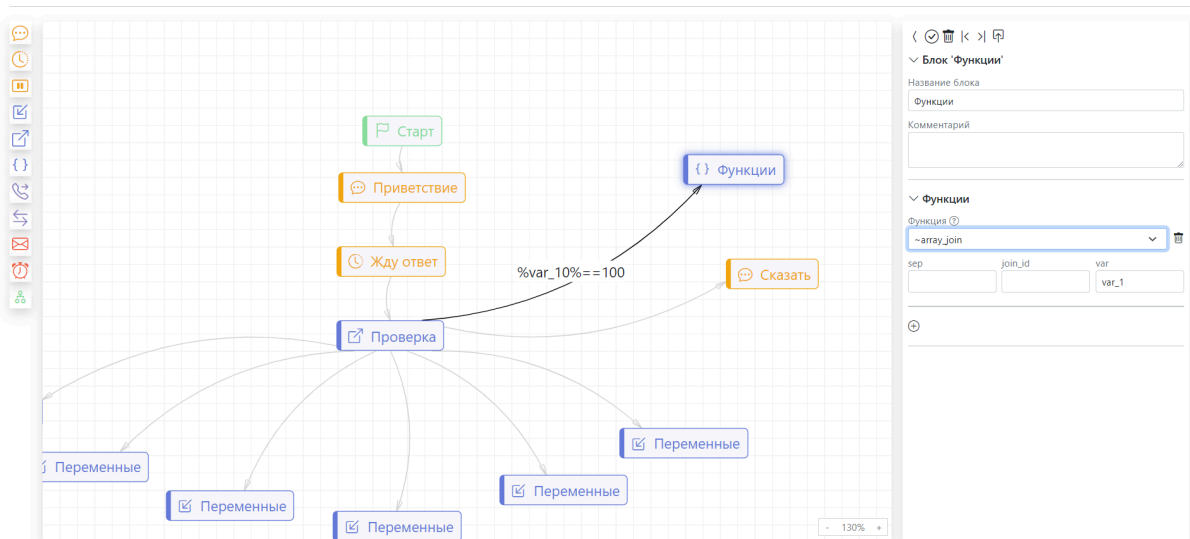


рис. 24

~string_digits – функция, которая определяет количество цифр в строке или переменной

- в поле “var/str” указывается переменная/строка, в которой будут искаться цифры;
- в поле “var” указывается переменная, куда запишется количество цифр (рис. 25).

☰ @Ответ (основной) < Руководство пользователя 📄

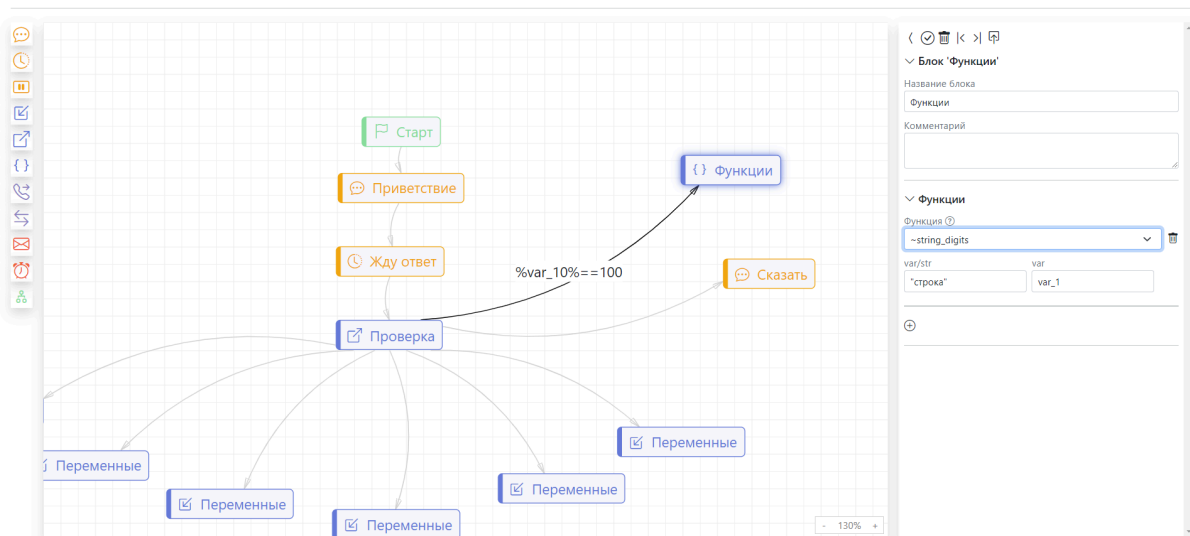


рис. 25

~string_len – функция, которая определяет длину строки

- в поле “var/str” указывается переменная или строка, длину которой нужно определить;
- в поле “var” указывается переменная, в которую запишется полученная длина (рис. 26).

🔗 @Ответ (основной) < Руководство пользователя 📄

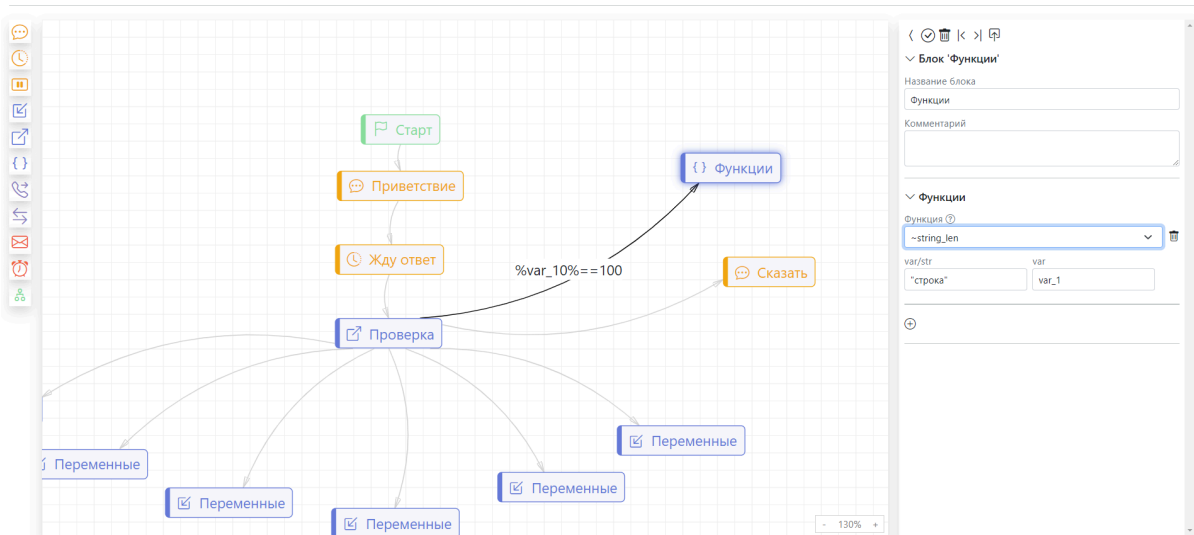


рис. 26

~array_set – функция, которая присваивает переменной индекс в массиве

- в поле “var/str” указывается переменная или строка;
- в поле “arr_el” указывается элемент массива, который задаётся пользователем (рис. 27).

☰ @Ответ (основной) < Руководство пользователя 📄

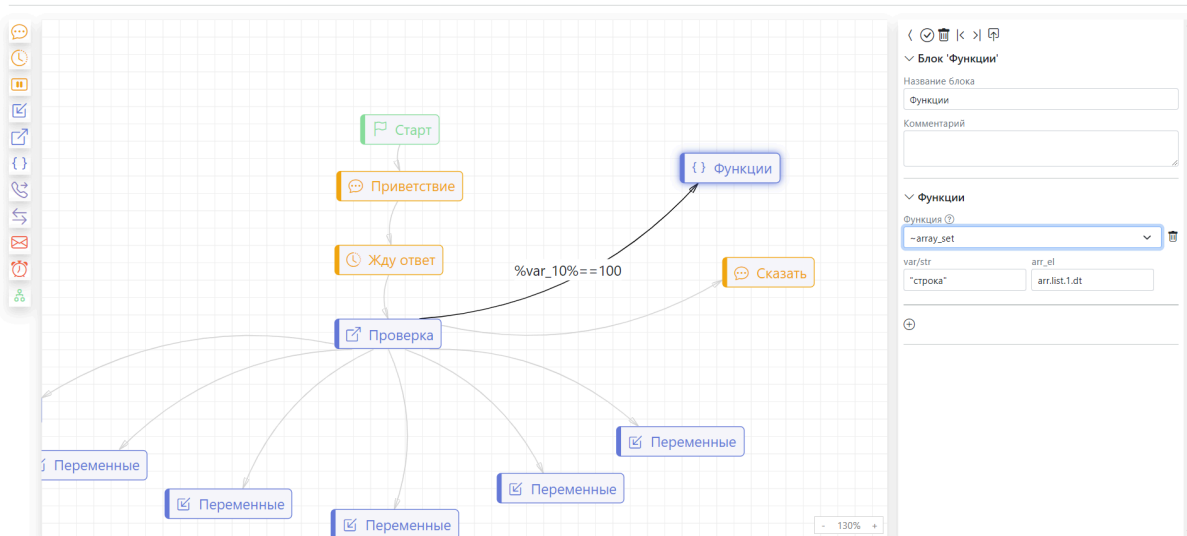


рис. 27

~array_get – функция, которая получает значение элемента массива

- в поле “arr_el” указывается элемент массива, который задаётся пользователем;
- в поле “var” указывается переменная, в которую запишется элемент массива (рис. 28).

☰ @Ответ (основной) < Руководство пользователя 📄

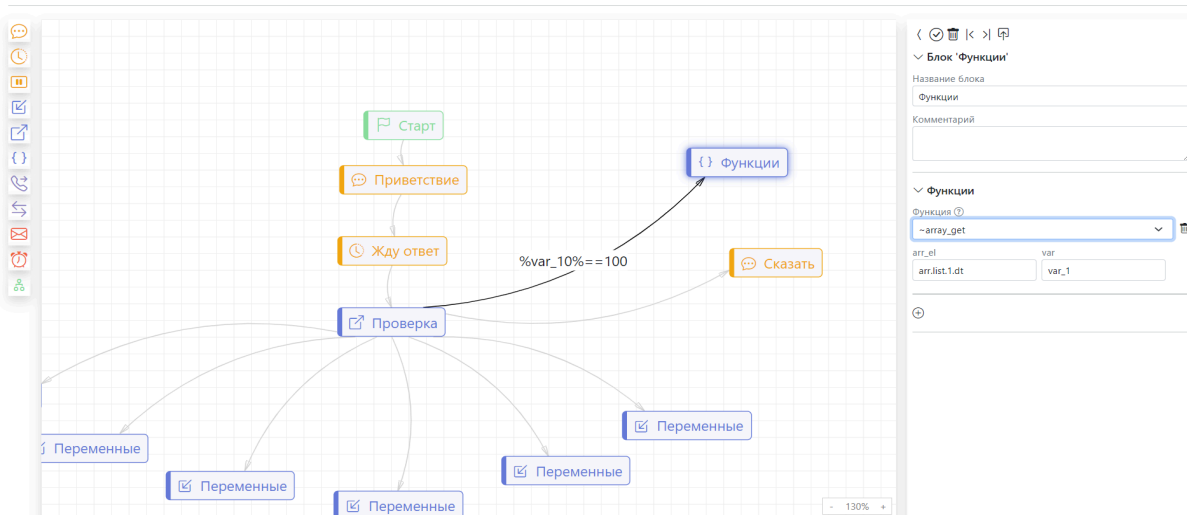


рис. 28

~array_size – функция, которая показывает длину массива

- в поле “arr” указывается имя массива;
- в поле “var” указывается переменная, в которую запишется длина массива (рис. 29).

☰ @Ответ (основной) < Руководство пользователя 📄

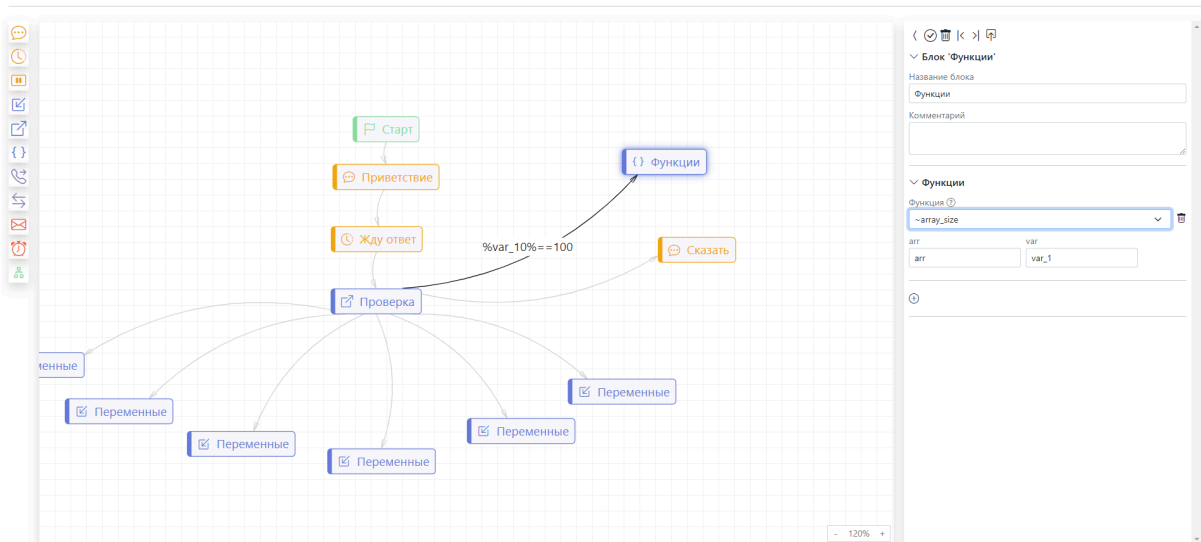


рис. 29

4.2.9 Функциональный блок “Запрос”

Данный блок позволяет отправлять HTTP-запрос.

При добавлении на рабочее пространство и нажатии на функциональный блок “Запрос” откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 30).

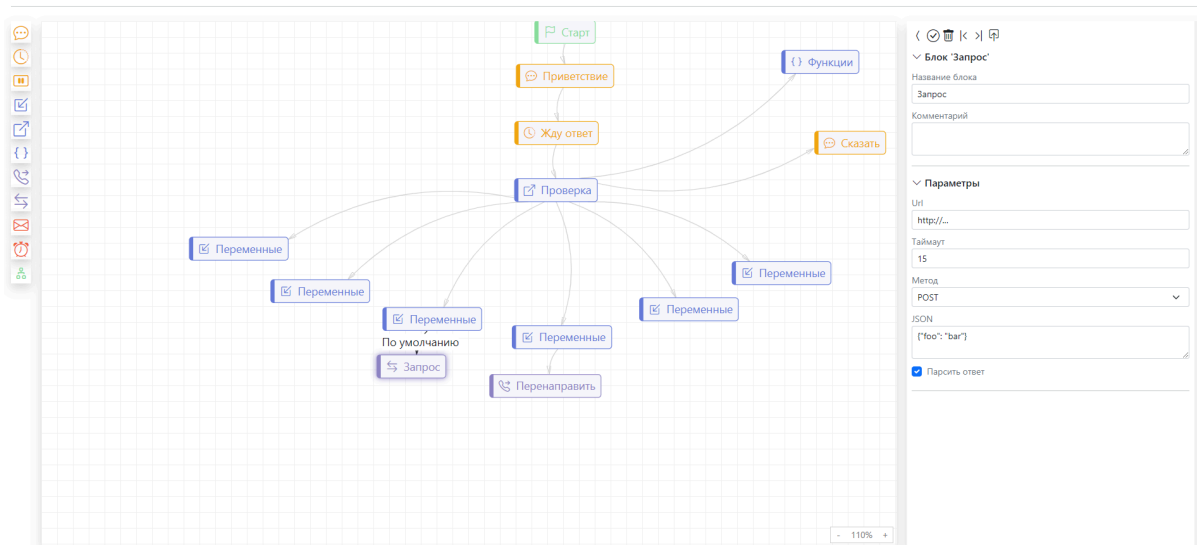


рис. 30

1. Раздел **“Блок “Запрос”** содержит:
 - Редактируемое поле **“Название блока”** – поле для изменения названия блока;
 - Редактируемое поле **“Комментарий”** – поле для комментария к блоку.
2. Раздел **“Параметры”** содержит:
 - **“URL”** – редактируемое поле для ввода адреса для запроса;
 - **“Таймаут”** – редактируемое поле для ввода значения таймаута запроса;
 - **“Метод”**, содержащий выпадающий список методов REST API:
 - GET — получение информации о данных или списка объектов;
 - DELETE — удаление данных;
 - POST — добавление или замена данных;
 - PUT — регулярное обновление данных.
3. **“JSON”** – редактируемое поле, в котором указываются передаваемые сервису параметры, как следует из названия, в формате .json.

Важно! Для того чтобы корректно настроить запрос, необходимо перед блоком “Запрос” добавить блок “Переменные” со следующими параметрами (рис. 31):

- Название переменной – HTTP_HEADERS;
- Значение переменной (пример) – {"Content-Type": "application/json", "Authorization": "Basic dm9pY2235ayUg34OlZiSWasgd0ludCEmMjE="}.

☰ @Ответ (основной) < Руководство пользователя 📖

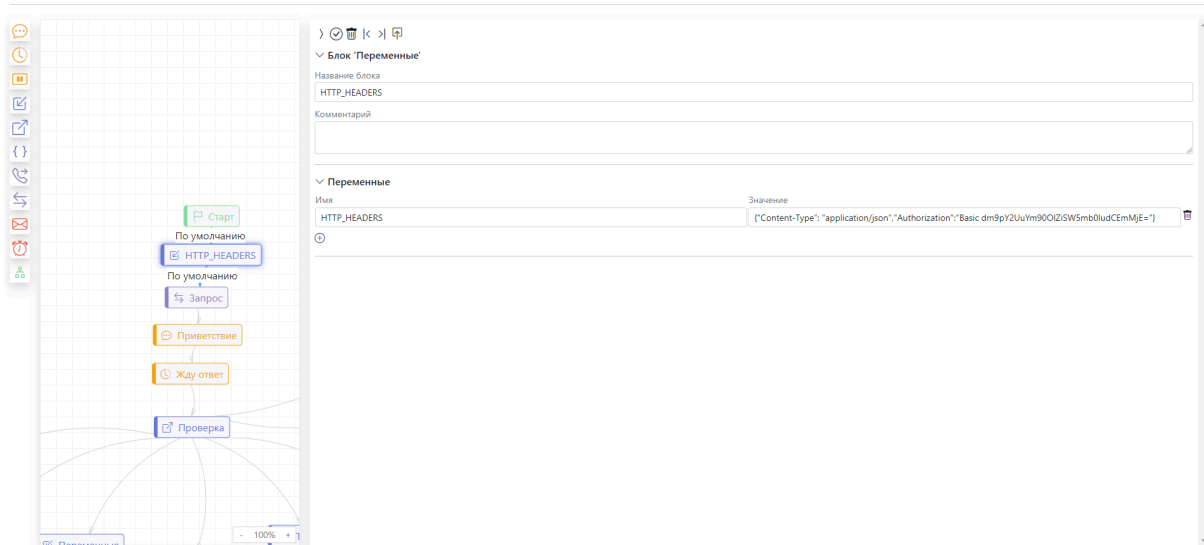


рис. 31

Заголовки HTTP позволяют клиенту и серверу отправлять дополнительную информацию с HTTP запросом или ответом. Такой дополнительной информацией может быть токен авторизации.

У заголовков в переменной HTTP_HEADERS наивысший приоритет и они заменяют исходные заголовки.

Когда передается несколько параметров, они записываются через запятую в JSON формате.

Важно! Спецификация HTTP не обязывает сервер понимать все методы. Обязателен только GET, а также не указывает серверу, что он должен делать при получении запроса с тем или иным методом, поэтому

при написании запроса, всегда стоит руководствоваться документацией, описывающей методы обращения.

4. Параметр “Парсить ответ”;

Если данное поле помечено, при получении ответа, он разбирается на переменные типа `response.<ключ массива>.<индекс элемента массива>.<ключ элемента массива>`.

Ниже представлена часть ответа на GET запрос к сервису `openweathermap` (рис. 32):

```
1 {
2   "coord": {
3     "lon": 37.6156,
4     "lat": 55.7522
5   },
6   "weather": [
7     {
8       "id": 500,
9       "main": "Rain",
10      "description": "light rain",
11      "icon": "10d"
12    }
13  ],
14  "base": "stations",
15  "main": {
16    "temp": 297.73,
17    "feels_like": 297.26,
18    "temp_min": 295.28,
19    "temp_max": 300.9,
20    "pressure": 1006,
21    "humidity": 39,
22    "sea_level": 1006,
23    "grnd_level": 989
24  },
25  "visibility": 10000,
26  "wind": {
27    "speed": 3.51,
28    "deg": 316,
29    "gust": 4.4
30  },
31  "rain": {
32    "1h": 1
33  },
34  "clouds": {
35    "all": 80
36  },
37  "dt": 1626710375,
38  "sys": {
39    "type": 2,
40    "id": 2000314,
41    "country": "RU",
42    "sunrise": 1626657123,
```

рис. 32

Ниже представлен пример того, как будет распарсен полученный ответ на платформе (рис. 33):

Переменная	Значение
response.coord.lon	37.6156
response.coord.lat	55.7522
response.weather.0.id	500
response.weather.0.main	Rain
response.weather.0.description	light
response.weather.0.icon	10d
response.base	stations
response.main.temp	295.39
response.main.feels_like	295.13
response.main.temp_min	294.25
response.main.temp_max	297.25
response.main.pressure	1007
response.main.humidity	56
response.main.sea_level	1007
response.main.grnd_level	990
response.visibility	10000
response.wind.speed	2
response.wind.deg	274
response.wind.gust	2.99
response.clouds.all	89
response.dt	1626673907
response.sys.type	2
response.sys.id	2000314
response.sys.country	RU
response.sys.sunrise	1626657123
response.sys.sunset	1626717553
response.timezone	10800
response.id	524901

рис. 33

Рассмотрим response.weather.0.id:

- **response** – часть имени переменной, добавляется системой ко всем переменным, сформированным в результате парсинга ответа;
- **weather** – ключ массива;
- **0** – индекс элемента массива;
- **id** – ключ элемента массива.

Важно! Код ответа (состояния) HTTP автоматически помещается в переменную http_status_code и показывает, был ли успешно выполнен определённый HTTP запрос. Коды сгруппированы в 5 классов:

Класс	Значение класса
1xx	Информационные
2xx	Успешные
3xx	Перенаправления
4xx	Клиентские ошибки
5xx	Серверные ошибки

Примеры часто используемых кодов ошибок:

200	Во время исполнения файла не произошло никаких ошибок
301	Файл к которому вы обращаетесь перемещён, либо система сайта перенаправляет вас на другой раздел сайта.

404	Файл не найден, файл не существует.
403	В данной папке нет индексного файла или вам запрещён к нему доступ. Вы пытаетесь зайти в пустую папку, либо доступ к данной папке каким-либо образом ограничен
500	Критическая ошибка сервера. Сервер не может выполнить код, так как не может его обработать. Ошибка в коде либо в настройках сервера.
502	Сервер не может обработать запрос. Сервер отказывает в обработке запроса, так как у Вас идёт превышение лимитов, либо на текущий момент не работает web-сервер.

4.2.10 Функциональный блок “Email”

Данный блок позволяет отправлять электронное письмо одному или нескольким получателям.

При добавлении на рабочее пространство и нажатии на функциональный блок “Email” откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 34).

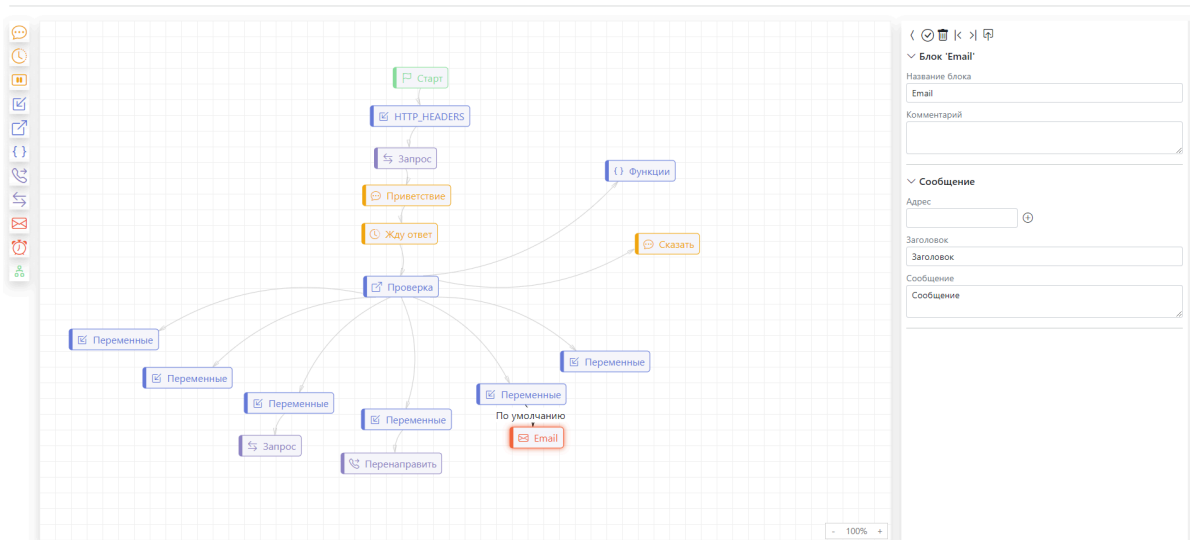


рис. 34

1. Раздел **“Блок “Email”** содержит:
 - Редактируемое поле **“Название блока”** – поле для изменения названия блока;
 - Редактируемое поле **“Комментарий”** – поле для комментария к блоку.
2. Раздел **“Сообщение”** содержит:
 - **“Адрес”** – редактируемое поле для ввода адреса для отправки электронного письма (рис. 35);
 - **“Заголовок”** – редактируемое поле для ввода названия заголовка;
 - **“Сообщение”** – редактируемое поля для ввода текста сообщения:

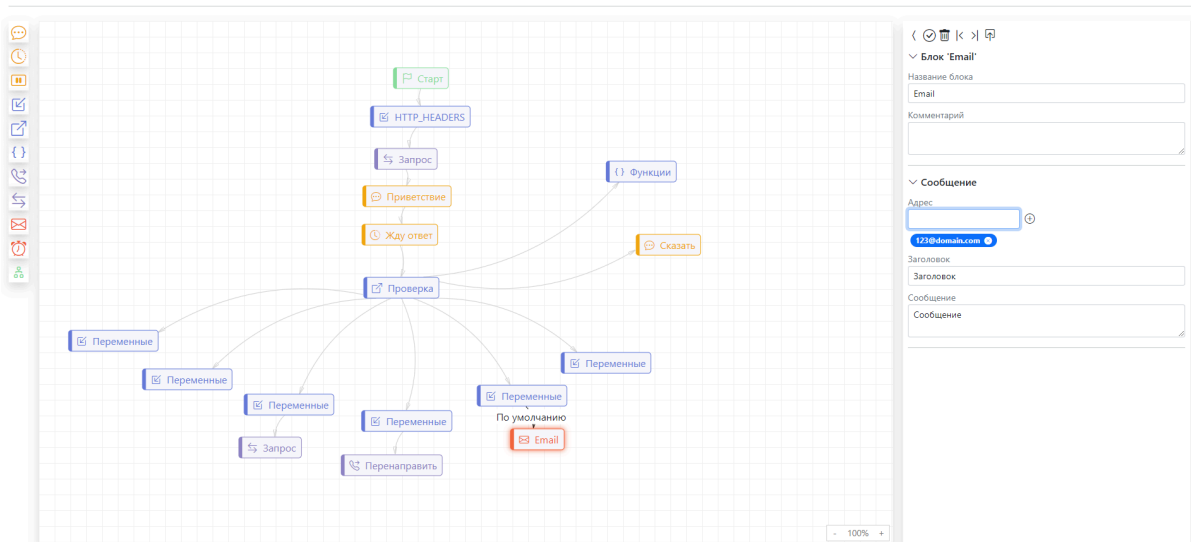


рис. 35

4.2.11 Функциональный блок “Текущее время”

Данный блок применяется для разветвления сценария в зависимости от системного времени сервера, исполняющего сценарий. С помощью данного блока можно озвучивать пользователю разные сообщения в рабочее и нерабочее время.

Важно! Подробные настройки блока возможны только в соединениях. При добавлении на рабочее пространство и нажатии на соединение блока “Текущее время” с другим блоком откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 36).

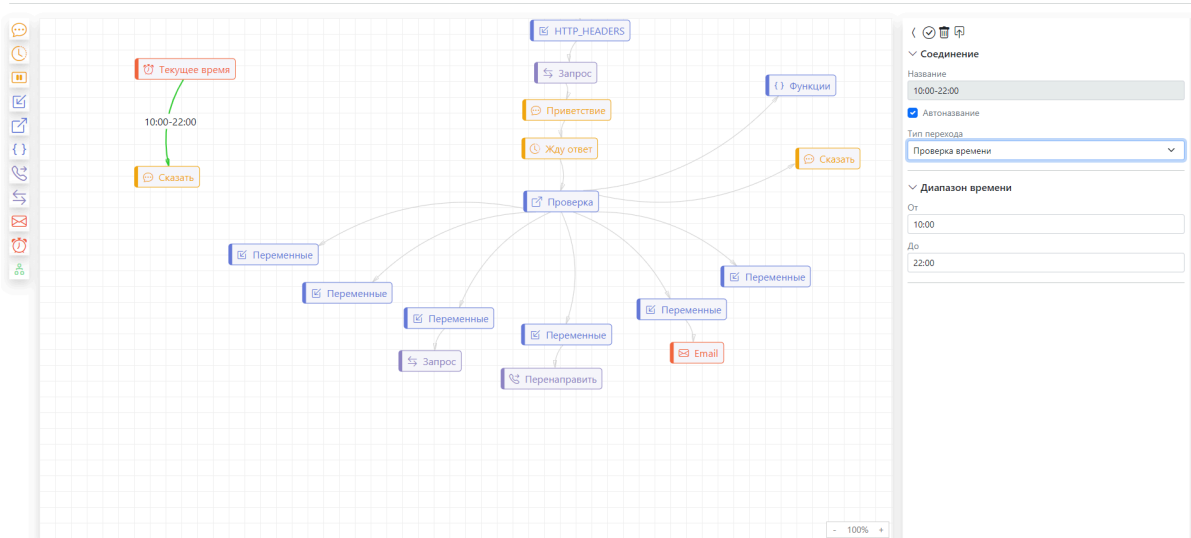


рис. 36

При нажатии на блок открывается окно настроек блока, в котором присутствуют редактируемые поля (рис. 37).

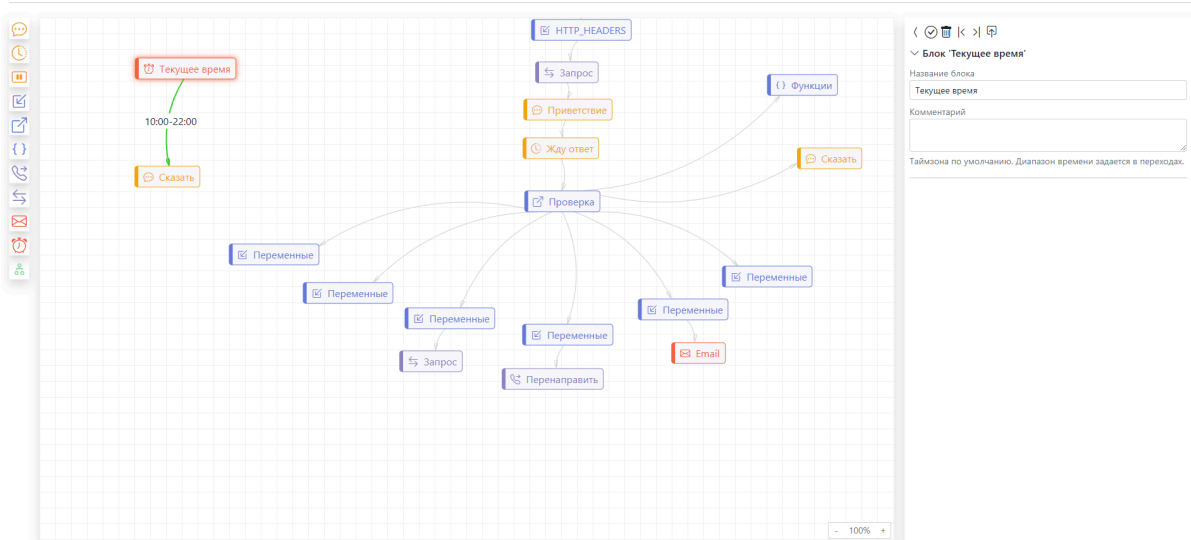


рис. 37

1. Раздел “Блок “Текущее время” содержит:
 - Редактируемое поле “Название блока” – поле для изменения названия блока;
 - Редактируемое поле “Комментарий” – поле для комментария к блоку (рис. 46).

2. Раздел “Соединение” содержит:
 - поле с названием и настройку “Автоназвание”;
 - выпадающий список “Тип перехода”.
3. Раздел “Диапазон времени” содержит:
 - редактируемые поля “От” и “До”.

Важно! Время нужно задавать в формате **ЧЧ:ММ**, иначе условие не сработает. Таким образом, правильная запись времени – 09:32. Неправильная запись времени 9:32, 9.32 и т.д.

4.2.12 Функциональный блок “Подсценарий”

Данный блок применяется для перехода между подсценариями внутри одного сценария.

При добавлении на рабочее пространство и нажатии на функциональный блок “Подсценарий” откроется окно настроек блока, в котором присутствуют разделы и редактируемые поля (рис. 38).

Важно! Для того чтобы назначить сценарий на функциональный блок, необходимо сначала создать или загрузить сценарий.

☰ @Ответ (основной) < Руководство пользователя 📖

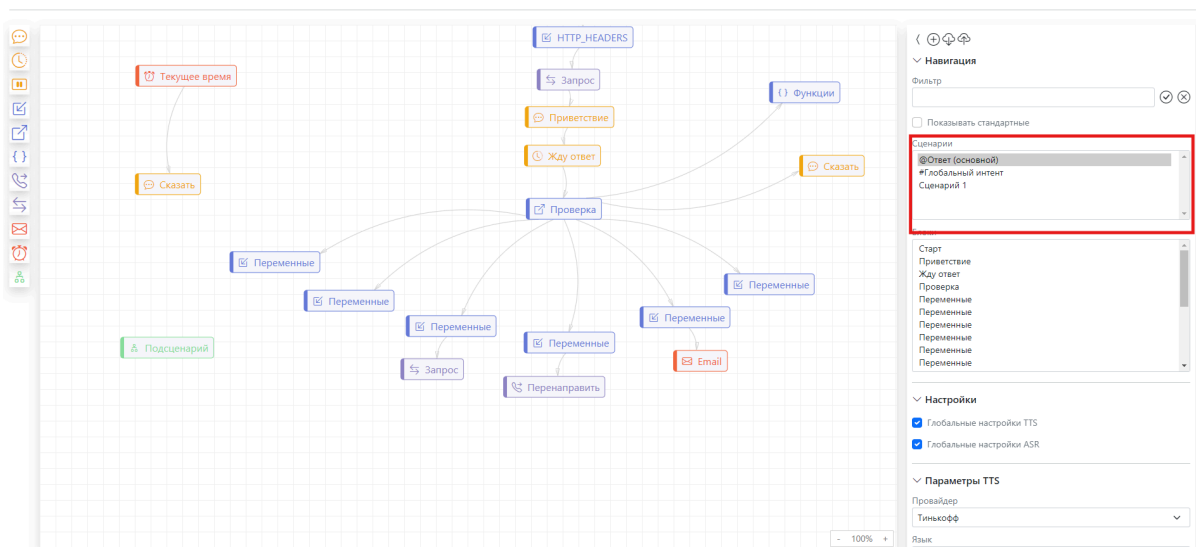


рис. 38

После того как сценарий отобразится в списке, нажмите на блок “Подсценарий”, вы увидите меню настроек функционального блока. Выберите необходимый вам сценарий из вариантов выпадающего списка подраздела “Сценарий” (рис. 39).

☰ @Ответ (основной) < Руководство пользователя 📖

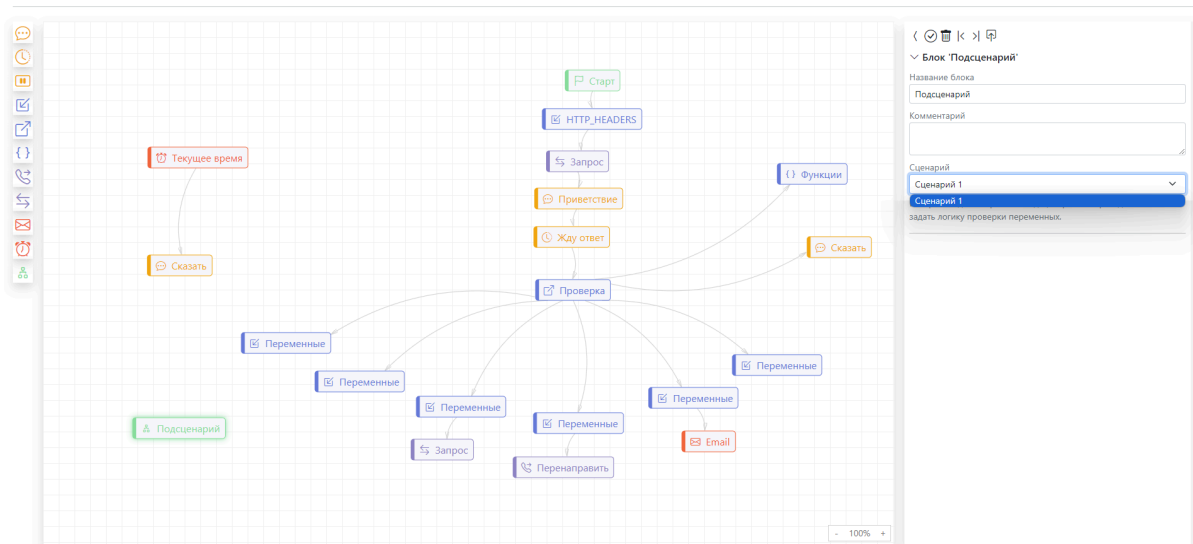


рис. 39

Важно! В соединениях можно задавать логику перехода к подсценарию (рис. 40).

☰ @Ответ (основной) < Руководство пользователя 📖

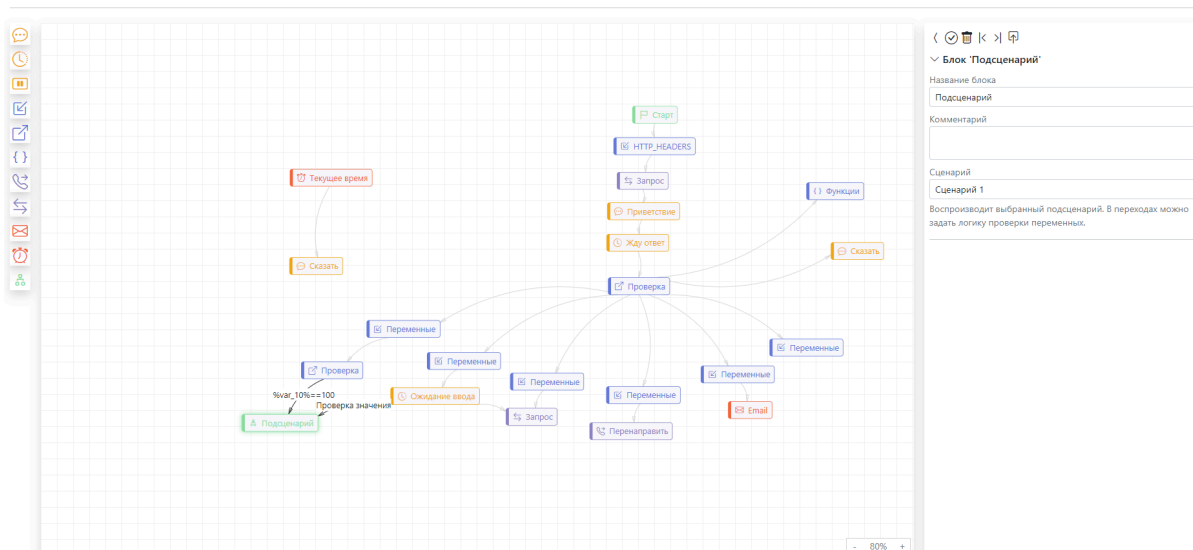


рис. 40

4.2.13 Соединения: копирование, вставка, переходы

Соединения в диалог-дизайнере сценариев можно копировать и вставлять.

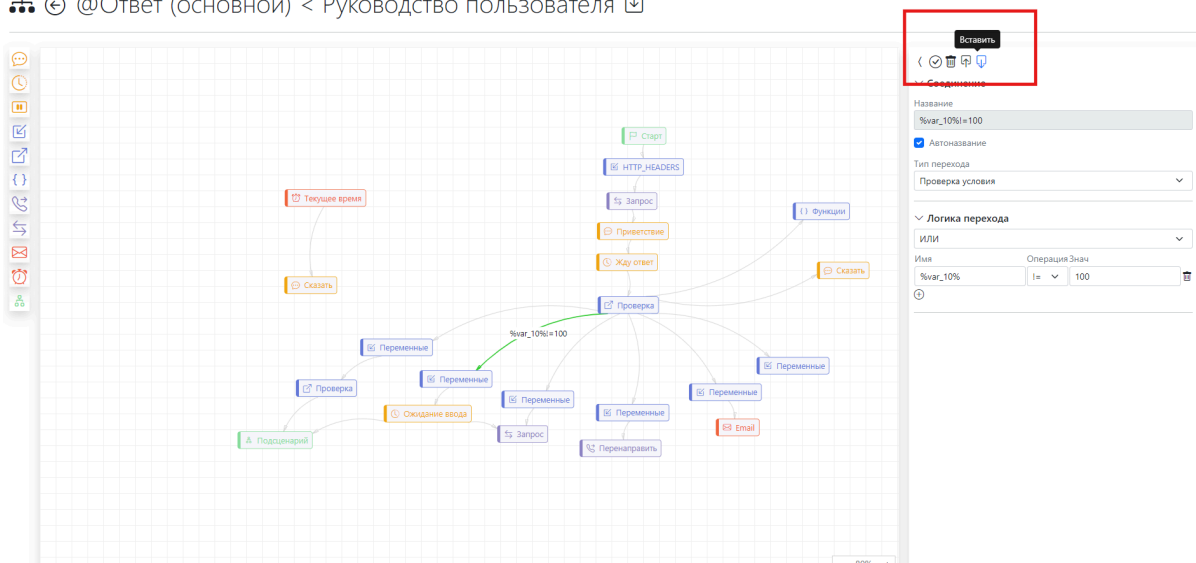
Для того чтобы скопировать соединение необходимо выбрать соединение, затем нажать на кнопку “Копировать” (рис. 47).

Важно! В системе реализована функция приоритетности соединений, то есть соединение, проведённое первым, будет иметь наивысший приоритет. Особенности функционирования соединений стоит иметь в виду при создании разветвлённых сценариев работа с несколькими ответвлениями от одного блока.



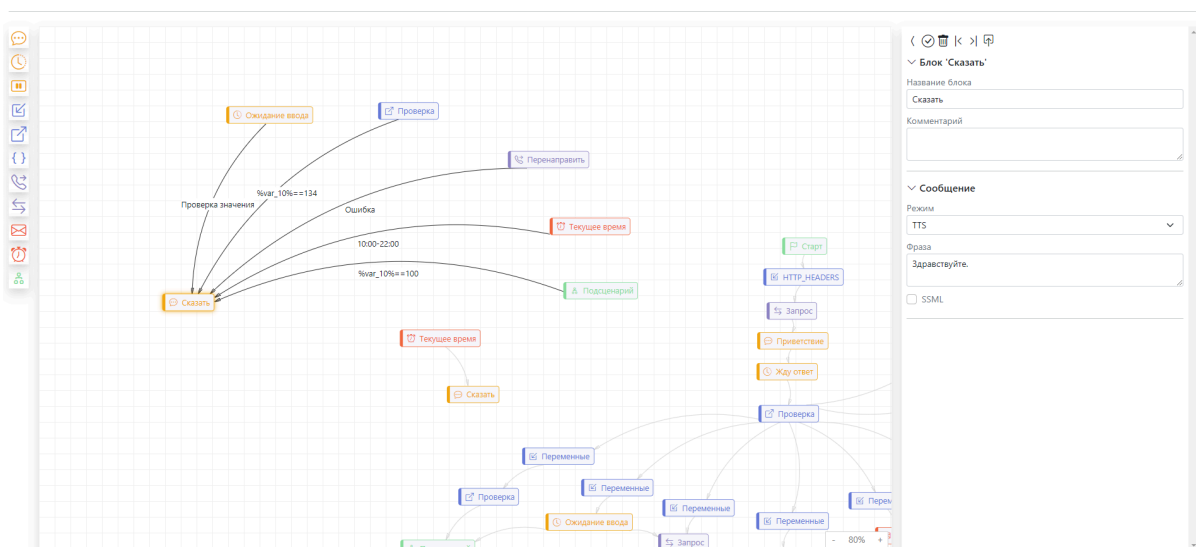
рис. 41

Для того чтобы вставить скопированное соединение необходимо сначала выбрать соединение, куда будет вставлено скопированное соединение. Затем необходимо нажать на кнопку “Вставить” (рис. 42).


рис. 42

Настраиваемые переходы доступны в соединениях от функциональных блоков:

- “Ожидание ввода”;
- “Проверка”;
- “Текущее время”;
- “Подсценарий” (рис. 43).


рис. 43

Переход от функционального блока **“Ожидание ввода”** содержит:

- Раздел **“Соединение”**, в котором указывается название блока и комментарий к нему, а также тип перехода: **“По умолчанию”** или **“Проверка значения”**;
- Раздел **“Значения”**, в котором выбирается **“Вид проверки”** и указывается **“Значение”** (рис. 44).

⌵ @Ответ (основной) < Руководство пользователя

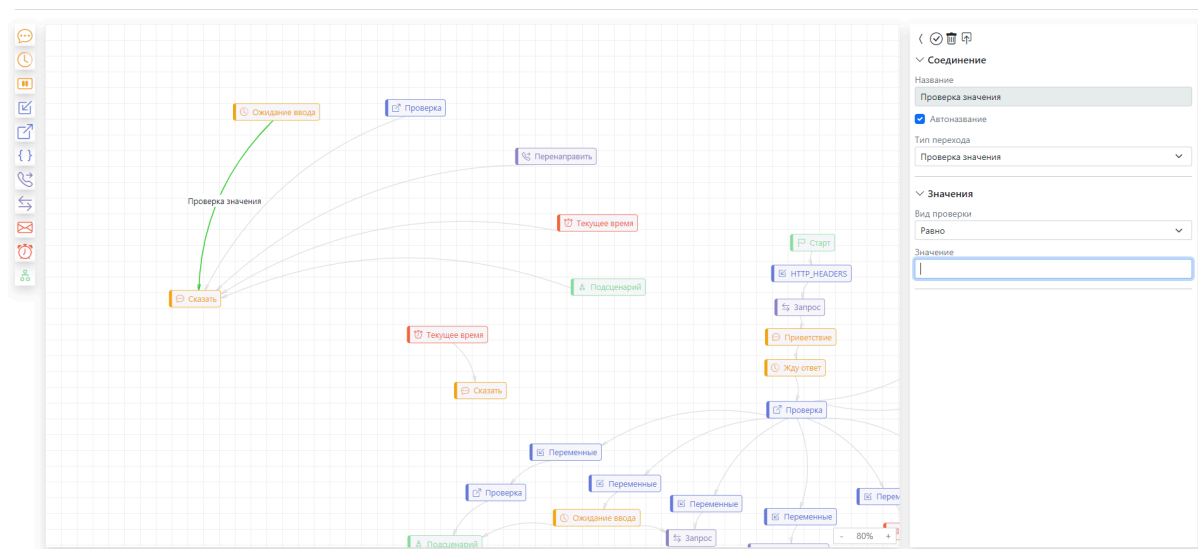


рис. 44

Соединение между блоком **“Ожидание ввода”** и другими функциональными блоками подразумевает 2 варианта: **“По умолчанию”** и **“Проверка значения”**. Соединение предлагает 4 вида проверки значения (**“Равно”**, **“Входит”**, **“Не входит”**, **“Рег. выражение”**). Таким образом, распознанная речь пользователя в блоке **“Ожидание ввода”** будет проверяться по заданным условиям и далее направляться по определённой ветке сценария.

Соединение между блоком **“Проверка”** и другими функциональными блоками подразумевает 2 варианта: **“По умолчанию”** и **“Проверка значения”**. Соединение предлагает 4 вида проверки значения (**“ИЛИ (режим совместимости)”**, **“ИЛИ”**, **“И”**, **“Сложное выражение”**).

Таким образом, заданная переменная будет проверяться по заданным условиям и далее направляться по определённой ветке сценария.

5. NLU

ChatBotNlu	
POST	<code>/api/create_model</code> Train Embedding
POST	<code>/api/add_new_question</code> Add New Question
POST	<code>/api/add_new_question_b</code> Add New Question B
PUT	<code>/api/update_model</code> Update Embedding
PUT	<code>/api/update_model_b</code> Update Embedding
POST	<code>/api/recognize</code> Read Note
DELETE	<code>/api/delete_model</code> Delete Model
GET	<code>/api/get_models</code> Get Models
POST	<code>/api/export_xlsx_created</code> Export Xlsx Created
POST	<code>/api/append_new_words</code> Update Word Dict
POST	<code>/api/get_words</code> Check Word
POST	<code>/api/upload_files</code> Root

рис. 45

Для загрузки Базы Знаний для робота необходимо:

1. Подготовить XLS файл с вопросами и ответами (AnswerText – то, что придёт в ответ, DefaultQuestionText — основной вопрос тематики, AdditionalQuestionText — вариации основного вопроса, FullQuestionText – служебные пометки, комментарии).
2. Загрузить датасет для обучения языковой модели. Подготовить запрос к `create_model`: использовать подготовленный файл; дать название создаваемой модели `model_name`; указать тип модели

bert/roberta; указать название датасета со словами-исключениями в `swords_file` (если имеется). Выполнить запрос к `create_model`.

3. Протестировать модель. Подготовить запрос к `recognize`: указать название своей созданной модели; указать тип модели (либо `roberta`, либо `string` в случае BERT); в поле `text` ввести тестовый вопрос. Выполнить запрос к `recognize`, изучить тело ответа.

Для обращения к созданной модели следует выполнить POST-запрос к эндпойнту **recognize**

<URL>/api/recognize

Формат тела запроса:

```
{  
  "text": "string", - сюда вставляем задаваемый вопрос  
  "model_name": "string", - здесь указываем имя интересующей нас модели  
  "model_type": "string", - здесь указываем тип интересующей нас модели  
  "swords_file_name": "string" - название датасета слов-исключений (если  
имеется)  
}
```