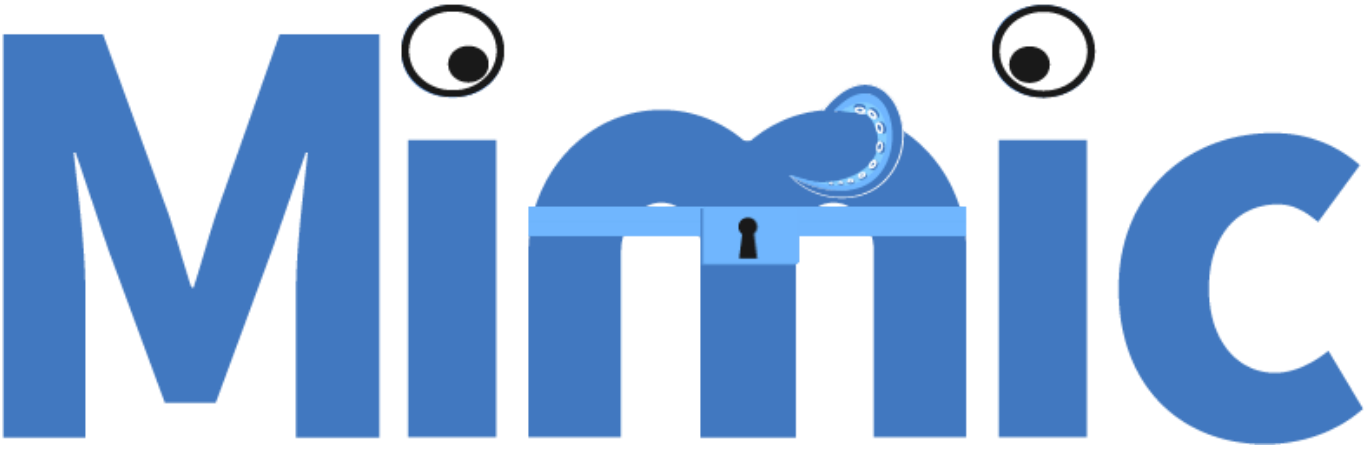
Пользовательская документация Mimic



> **Mimic** - это сервис для разработки и тестирования программного обеспечения, который взаимодействует с внешними API - он позволяет пользователю создавать имитацию или симуляцию поведения реального API (создать Mock), чтобы проверить работу своего приложения

Mock API может быть полезным в следующих случаях:

- Реальное API еще не готово или недоступно для тестирования
- Проверка ситуаций, которые сложно воспроизводимы на реальном сервисе
- Изолированное тестирование отдельных компонентов приложения

Использование моков API упрощает и ускоряет разработку, улучшает надежность тестов и обеспечивает более эффективное тестирование приложений.

Для каждого проекта можно создать свой набор маршрутов (endpoint/method). В рамках маршрута действуют настроенные правила, согласно которым будет выдаваться указанный в правиле ответ.

Срабатывание правил идёт последовательно. Для успешного срабатывания правила входящий запрос должен соответствовать всем условиям фильтра из правила. Фильтры срабатывают через условие "И".

> Например: Если в правиле два фильтра Заголовок Host со значением hdtech и в параметрах запроса фильтр с id значение 33, то для срабатывания правила во входящем запросе должен быть указан и заголовок Host=hdtech и параметр id=33.

Авторизация

В системе **Mimic** реализованы два варианта авторизации: **1. Основной способ авторизации - через систему Mimic**

- Пользователи могут пройти авторизацию, используя учетные данные, созданные администратором системы Mimic. Этот способ подходит **для всех пользователей**, включая тех, кто не является штатными сотрудниками компании.

2. Дополнительный способ авторизации - через интеграцию с KeyCloak

- Данный вариант доступен только в том случае, если в вашей компании **настроена интеграция с KeyCloak**. Все учетные записи штатных сотрудников созданы в KeyCloak, что позволяет им входить в систему **Mimic** с доступом на **просмотр**. Использование KeyCloak обеспечивает возможность единого входа (SSO), что упрощает процесс аутентификации и поднимает уровень безопасности.

Роли в системе

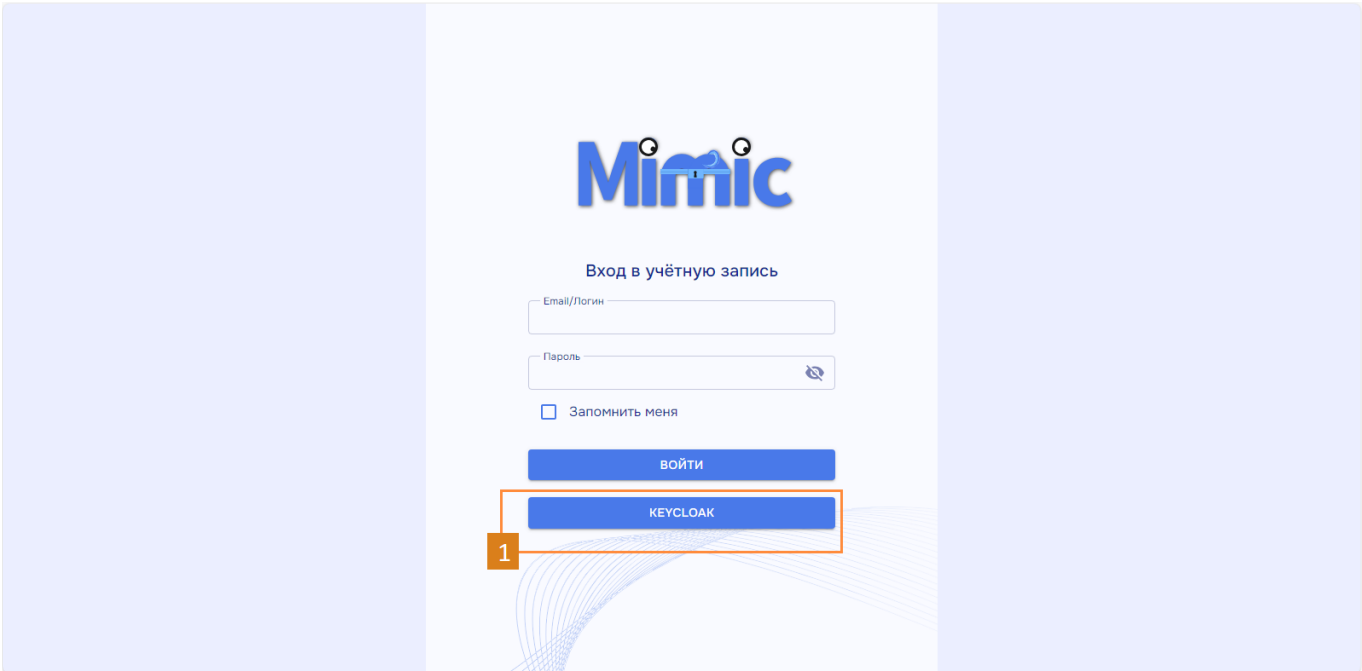
Система **Mimic** на данный момент поддерживает 4 роли:

- Администратор. Может просматривать и вносить изменения во все проекты платформы Mimic, добавлять пользователей в систему, назначать глобальные роли.
- Пользователь. Может просматривать и вносить изменения во все проекты платформы Mimic.
- Зритель. Может просматривать все проекты платформы Mimic.
- Гость. Может создавать проекты и может быть назначен владельцем, пользователем или зрителем на другие проекты.

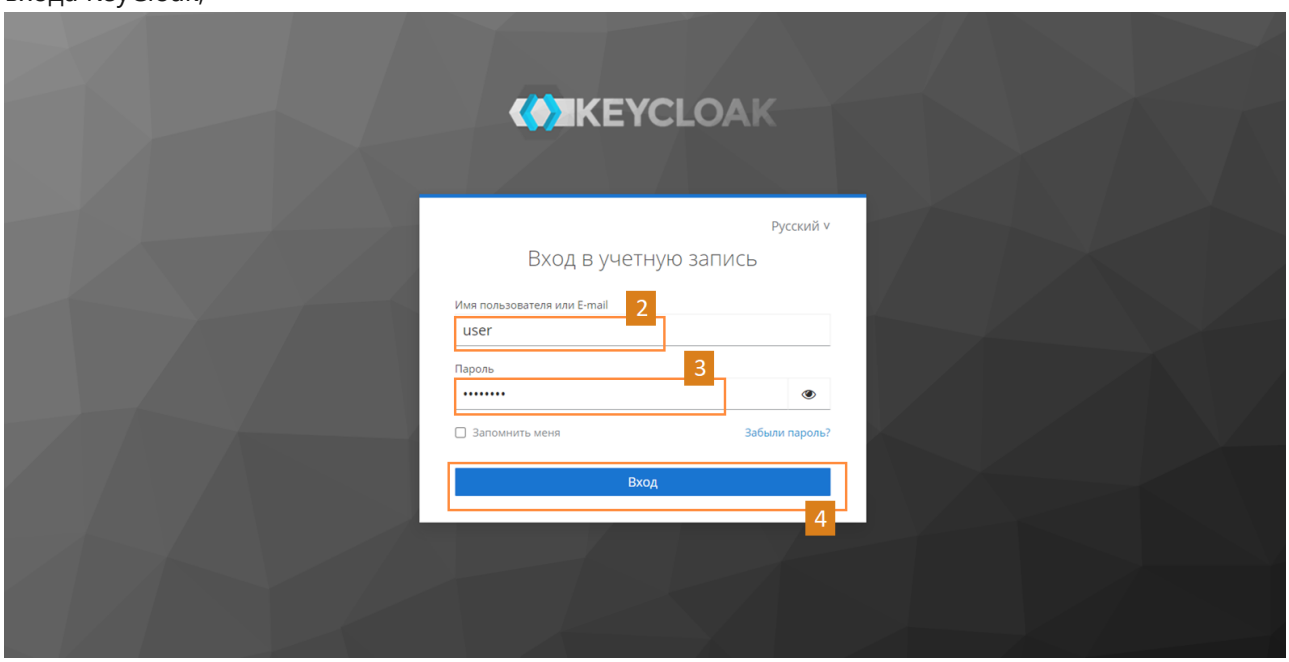
Также каждый пользователь может иметь отдельную роль в рамках определенного проекта (Подробнее: Роли на проекте).

Авторизация пользователя через KeyCloak

Пример авторизации пользователя **Описание действий:**



1. Нажимаем кнопку **KeyCloack** на странице входа, после чего происходит переадресация на окно входа KeyCloak;



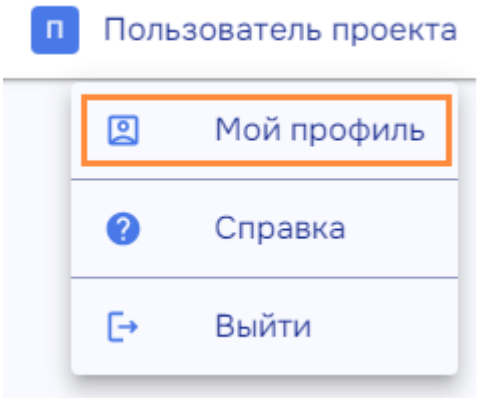
2. Вводим **Логин** от своей учётной записи (те же данные, что и для входа на рабочую машину);
3. Вводим **Пароль** от своей учётной записи (те же данные, что и для входа на рабочую машину);
4. Нажимаем кнопку **Вход**.

Профиль

Профиль пользователя представляет собой персонализированный раздел, который предоставляет пользователю возможность просматривать и управлять своей учетной записью.

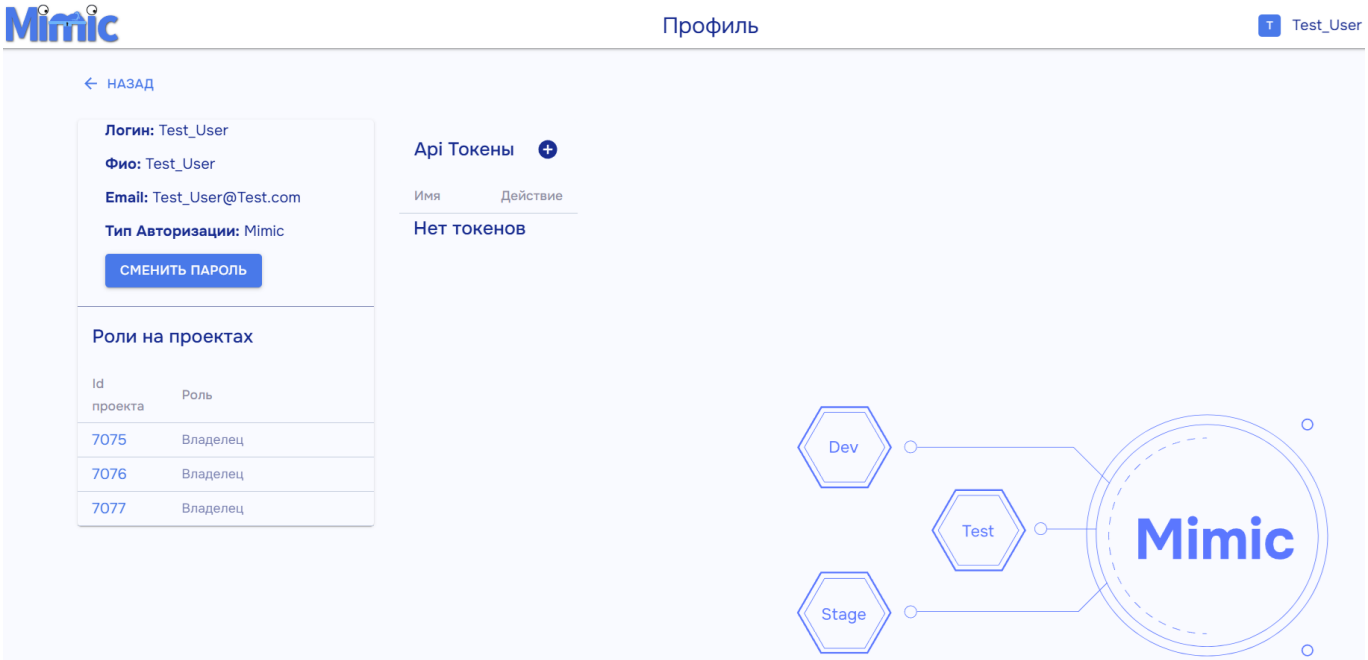
Переход в раздел "Профиль"

Для перехода в раздел Профиль необходимо нажать на свой аватар и перейти в раздел **Мой профиль**:



Наполнение раздела "Профиль"

Интерфейс раздела **Профиль**:



В данном разделе пользователю доступны следующие действия:

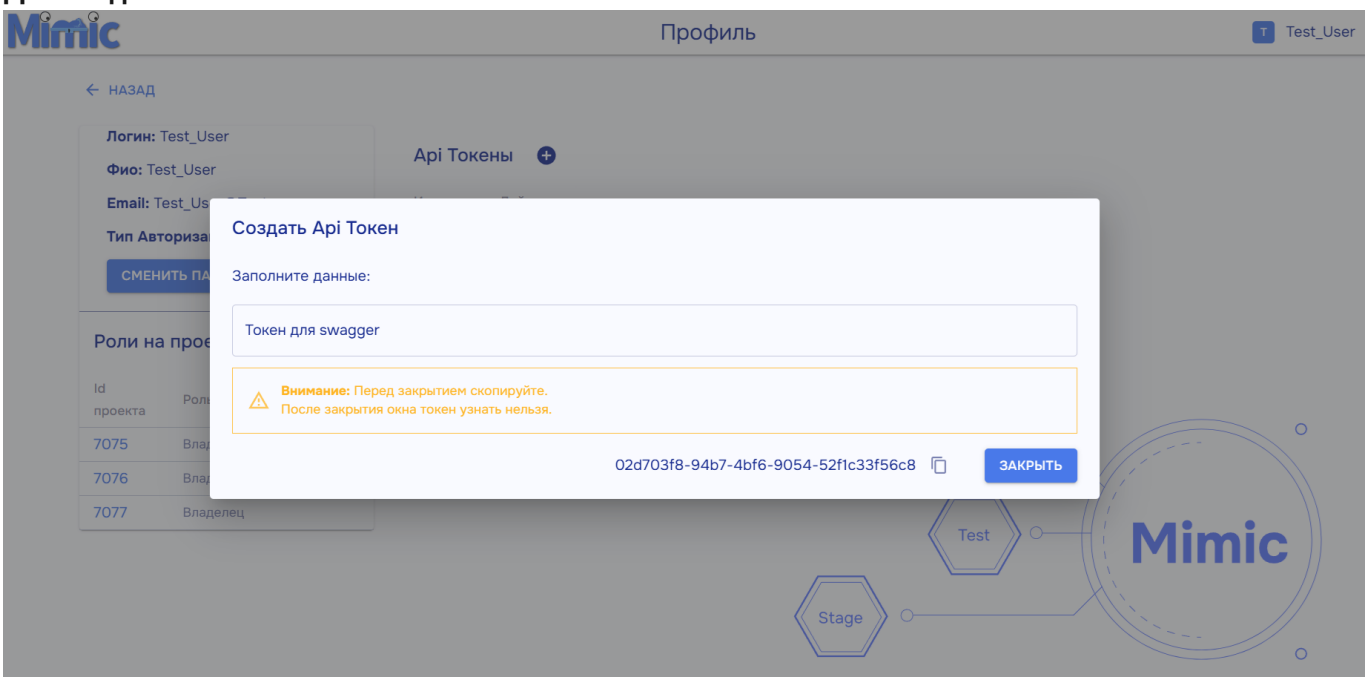
1. **Просмотр личных данных**

- Логин: Уникальный идентификатор пользователя, используемый для входа в систему.
- ФИО: Полное имя пользователя, отображаемое на платформе Mimic.
- Email: Адрес электронной почты, связанный с учетной записью, который используется для уведомлений и восстановления доступа.
- Тип авторизации: Информация о способе входа в систему (Keycloak или Mimic).

2. **Смена пароля** Пользователь может изменить свой пароль. Процесс включает ввод текущего пароля, нового пароля и его подтверждение.
3. **Отслеживание ролей на проектах** Пользователь может просматривать свои текущие роли на различных проектах.
4. **API Токены** Для интеграции с Mimic необходимо сгенерировать API-токен.

> **Токен** — это уникальный идентификатор, который используется для аутентификации и авторизации пользователя при доступе к ресурсам или сервисам.

Для создания нового токена:



a. Введите желаемое название токена и сгенерируйте его; **b. Скопируйте сгенерированный токен**. Обязательно сохраните его в безопасном месте, так как после закрытия окна токен будет недоступен для просмотра.

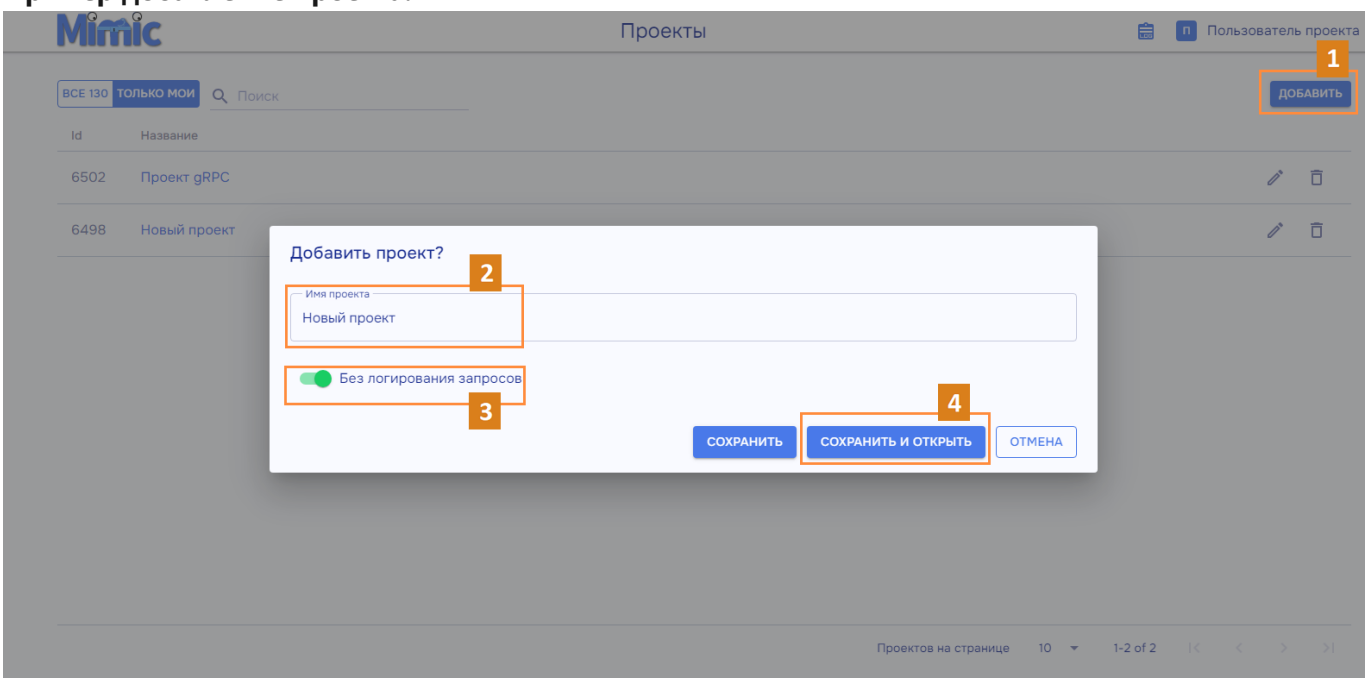
Проект

> **Проект** - это сущность, в рамках которой можно создавать изолированную от других проектов имитацию поведения реального API. На один мокируемый сервис можно создать несколько проектов, например, для изоляции настроек разных команд.

Создание проекта

Создание проекта осуществляется с главной страницы системы mimic, где отображён общий список всех ранее созданных проектов.

Пример добавление проекта:



Описание действий

1. Нажать кнопку **Добавить** в правом верхнем углу экрана;
2. В поле **Имя проекта** ввести желаемое имя создаваемого проекта;
3. Установить признак **Логирования запросов** (подробнее в разделе Журнал запросов);
4. Если после создания проекта требуется редактирование проекта, то нажать соответствующую кнопку **Сохранить и добавить**, если открытие не требуется - **Сохранить**.

Также существует возможность просмотра **Журнала запросов** . Подробнее: Просмотр Журнала запросов

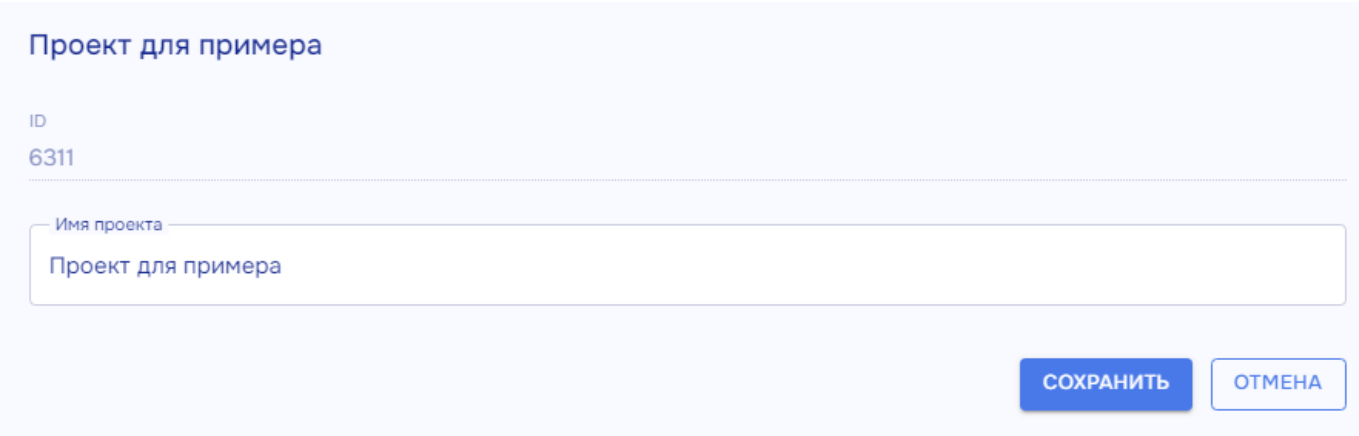
> **Права пользователей**
>
> Создать проект может любой Пользователь. После создания он становится Владельцем этого проекта.

Общие настройки проекта

> **Общие настройки** проекта включают в себя информацию о проекте, которая может быть полезной для идентификации проекта, а также настройку логирования запросов.

За **идентификацию проекта** отвечают два параметра:

1. **ID проекта**: Это уникальный идентификатор вашего проекта, который используется для ссылок на проект в базе данных или других системах. ID проекта нельзя изменить после создания, так как он служит для однозначной идентификации проекта.
2. **Имя проекта**: Это название вашего проекта, которое можно изменить в любое время. Имя проекта обычно используется для более удобной идентификации проекта.



Логирование запросов

В разделе **Логирование запросов** можно **включить или выключить** логирование, а также **настроить правила очистки** журнала логирования, либо же **очистить** журнал. Подробнее про журнал можно узнать в разделе Журнал логирования.

> **Журнал запросов** представляет собой список всех входящих запросов, которые отпраляются в систему.

Со временем журнал логирования может стать **довольно громоздким**, что **затрудняет поиск нужной информации** и негативно сказывается на **производительности** системы. Поэтому **регулярная очистка журнала** логирования является хорошей практикой. Она помогает поддерживать порядок, освобождает место для новых записей и упрощает поиск информации.

> **Управлять очисткой журнала логирования** может только пользователь с ролью Владелец проекта!

Правила очистки журнала логирования

Существуют 4 правила, по которым есть возможность настроить очистку журнала логирования. Правила срабатывают независимо друг от друга и в строгом порядке:

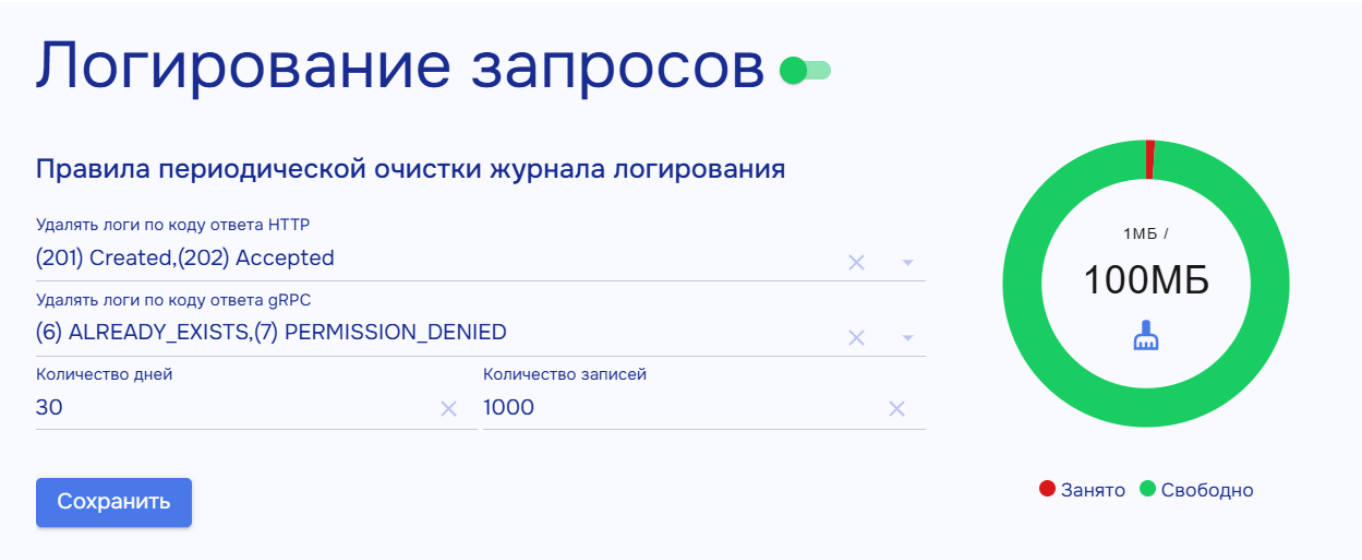
1. Очистка журнала по коду ответа gRPC В журнале копится много записей с входящими и исходящими сообщениями. Исходящие сообщения, в свою очередь, содержат код ответа gRPC. Но не все коды ответа могут представлять интерес для пользователя - удаление записей с такими кодами поможет сосредоточиться только на необходимой информации. Для этого необходимо настроить правило, которое будет удалять нежелательные записи.

2. Очистка журнала по коду ответа HTTP То же самое правило, как и в пункте выше, только для кодов ответа HTTP.

3. Очистка журнала по дням Очистка по дням означает удаление записей, которые старше определенного количества дней. Например, если вам нужно хранить логи только за последние 30 дней, можно задать правило для автоматической очистки записей в 30 дней.

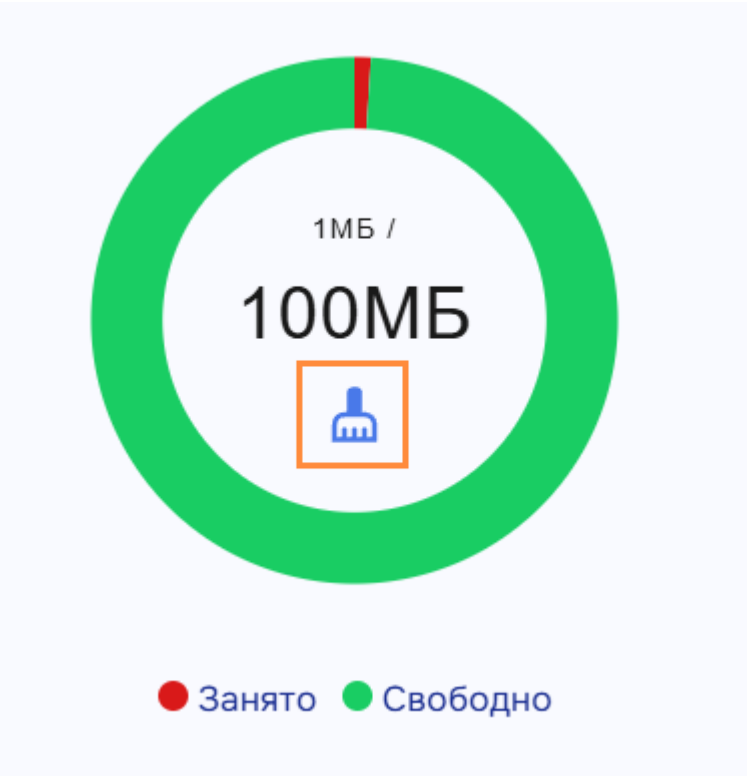
4. Очистка журнала по количеству записей Если журнал стал слишком большим, полезно ограничить его объем, сохранив только необходимое количество записей. Например, можно установить ограничение на 1000 последних записей.

Круговая диаграмма, расположенная справа, демонстрирует количество доступной памяти из всей возможной для данного проекта.



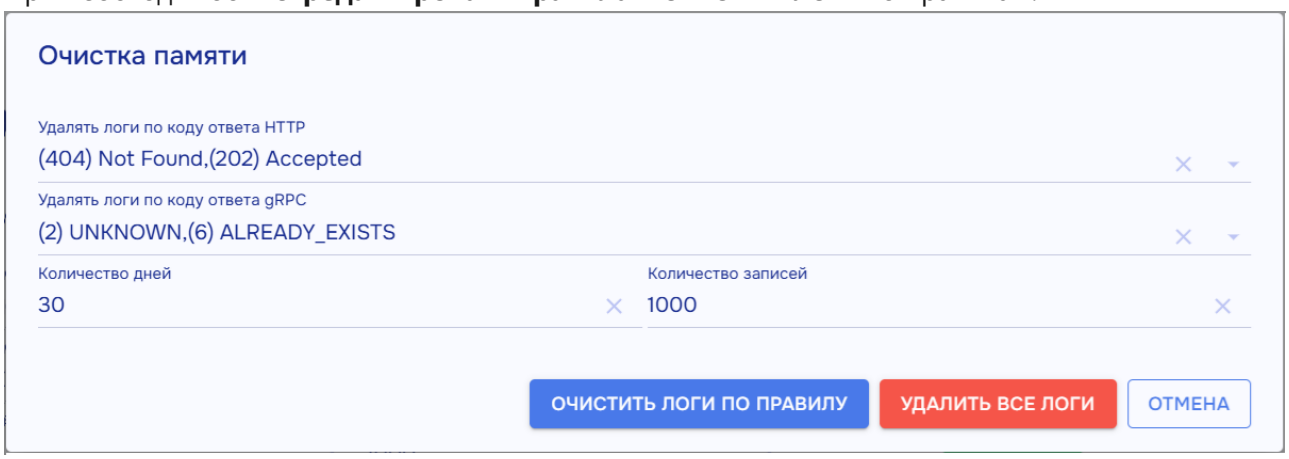
Мгновенная очистка журнала

Также существует возможность **мгновенной очистки** записей в журнале логирования по необходимому проекту. Для этого следует нажать на кнопку  .



После чего появится модальное окно, где можно:

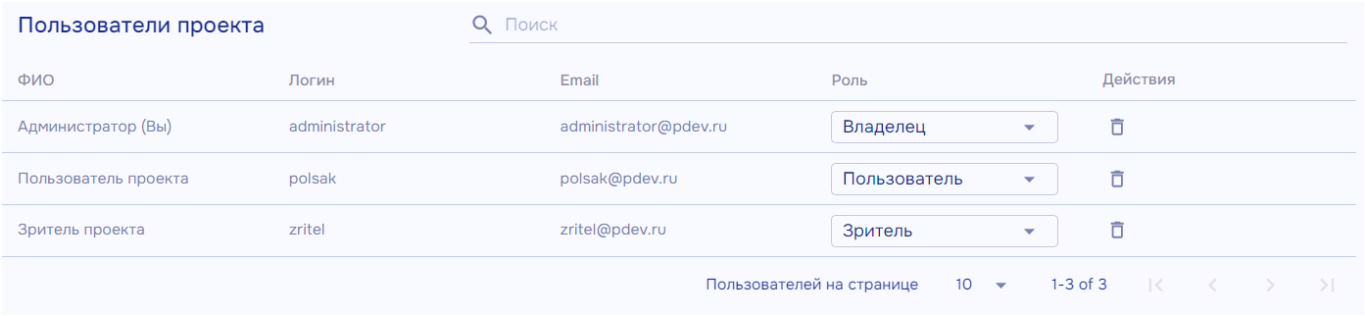
1. **Удалить все логи** по нажатию соответствующей кнопки
2. При необходимости **отредактировать правила** и **очистить логи** по правилам.



> **Права пользователей**
>
> Вносить изменения в настройки проекта может только Владелец проекта и Пользователь.

Права пользователей

Доступ к настройке **"Права пользователей"** предоставляется **исключительно Владельцу проекта** и обеспечивает возможность управления участниками проекта, а также назначения им определенных ролей: **Владелец, Пользователь и Зритель**:



Роли проекта

1. Владелец проекта

- Имеет полный доступ ко всем функциям проекта;
- Может управлять участниками проекта, включая добавление и удаление пользователей;
- Отвечает за назначение ролей другим участникам проекта;
- Имеет полномочия на удаление проекта.

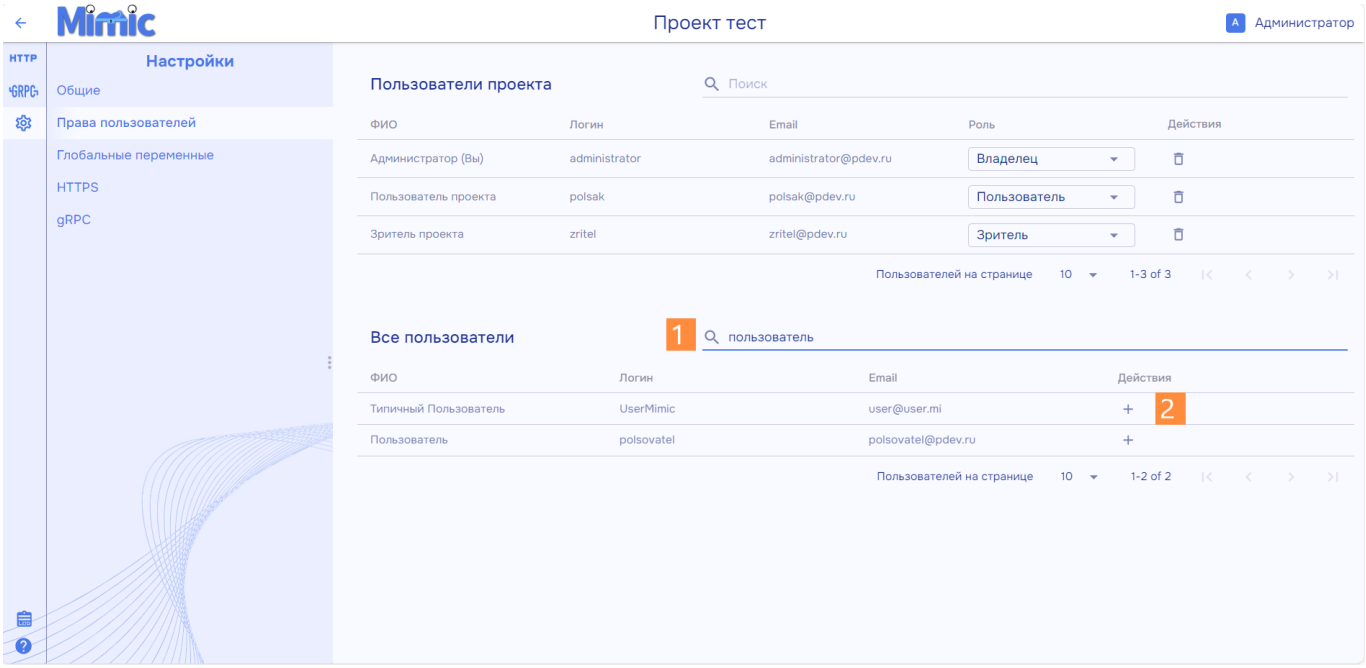
2. Пользователь проекта

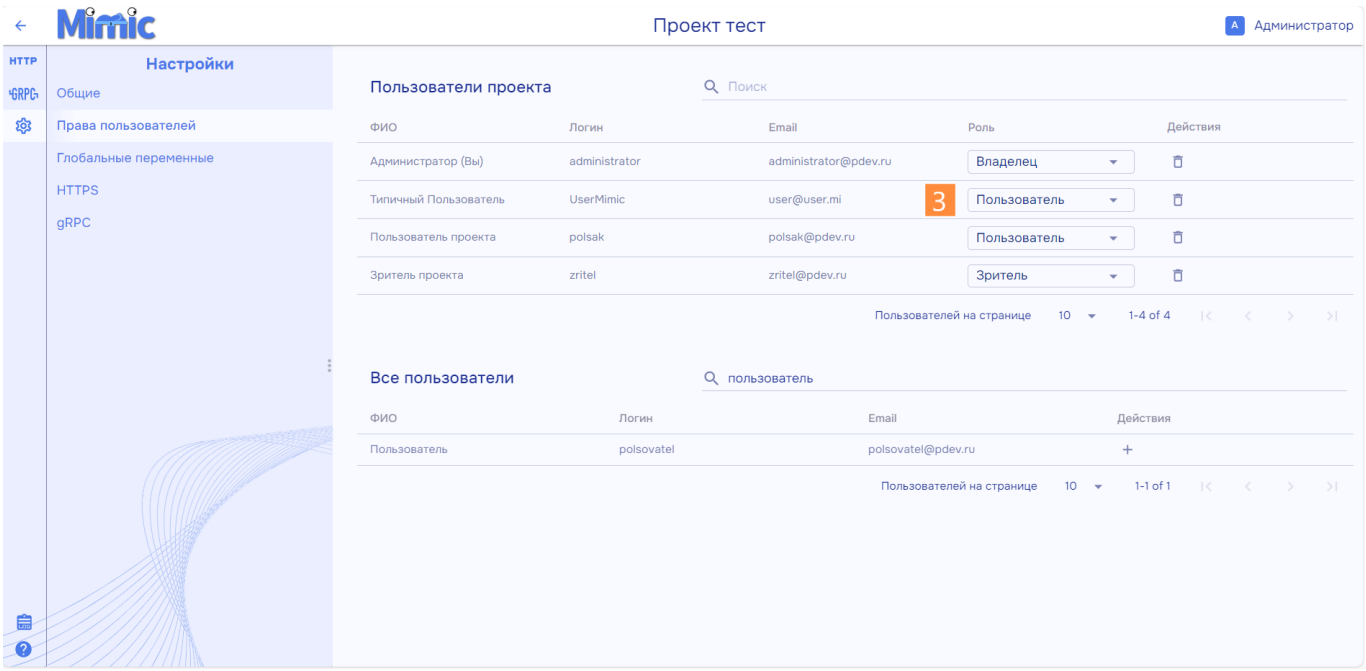
- Имеет полный доступ ко всем функциям проекта, кроме удаления проекта;
- Не имеет права управлять пользователями и их ролями.

3. Зритель проекта

- Имеет доступ только к просмотру информации о проекте (маршруты, правила, ответы по умолчанию, Urls для мокирования);
- Не имеет возможности вносить какие-либо изменения в проект;
- Используется для предоставления доступа к информации о проекте сторонним лицам или заинтересованным сторонам.

Пример добавления нового пользователя в проект и назначения роли:





Описание действий:

1. На вкладке **Права пользователей**, находящейся в меню **Настроек**, осуществить поиск пользователя, которого необходимо добавить в проект;
2. **Добавить** пользователя в пользователи проекта посредсвом нажатия на кнопку **+**;
3. **Изменить роль** пользователя на необходимую. По умолчанию присваивается роль **Пользователь проекта**.

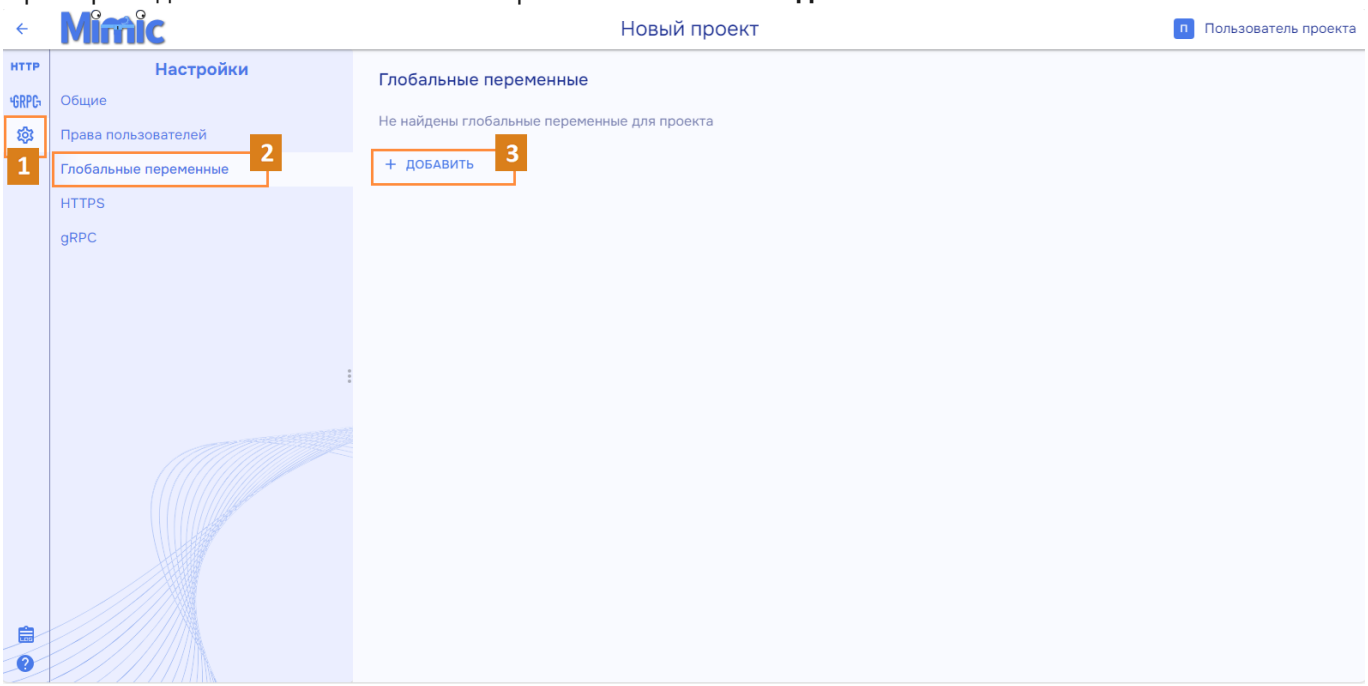
> **Права пользователей**
>
> Доступ к настройке "Права пользователей" предоставляется исключительно Владельцу проекта.

Глобальные переменные

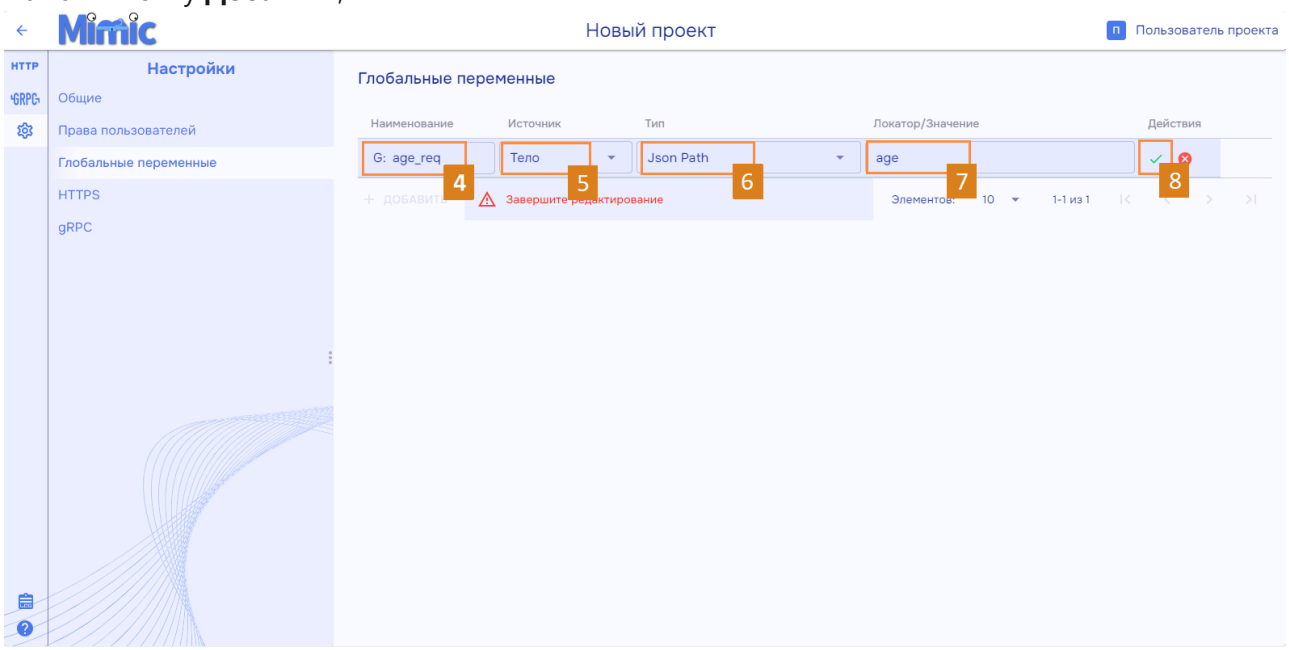
Глобальные переменные - это переменные, доступные для использования в любой части проекта. Используются для выдачи фиксированного значения или значения, которые генерируются динамически в любом маршруте и правиле проекта.

Создание глобальной переменной

Пример создания новой глобальной переменной **Описание действий:**



1. Перейти на вкладку **Настройки**, обозначенную иконкой ;
2. Перейти во вкладку **Глобальные переменные**;
3. Нажать кнопку **Добавить**;



4. Указать имя глобальной переменной в поле **Наименование**, оно должно быть уникальным в рамках проекта;
5. В поле **Источник** выбрать соответствующий источник, откуда хотим получить данные. Например, для получения из тела входящего запроса необходимо выбрать *Body*, из заголовка - *Header* и тд;
6. Выбрать **Тип** переменной, локатор которой будет указываться далее;
7. Указать **Локатор**, который необходим для отбора значения из запроса (для заголовка это будет *Имя* Header, для body это может быть *XPath, JsonPath* или *Regex*-выражение);
8. Нажать **зеленую галочку** ✓ для сохранения созданной глобальной переменной;
9. При необходимости глобальную переменную можно удалить посредством нажатия на значок корзины.

Подробнее о создании переменных можно узнать в разделе Переменные

> **Права пользователей**
>
> Изменять, создавать и удалять глобальные переменные может как Владелец проекта, так и Пользователь. Читателю же доступен только просмотр.

HTTPS

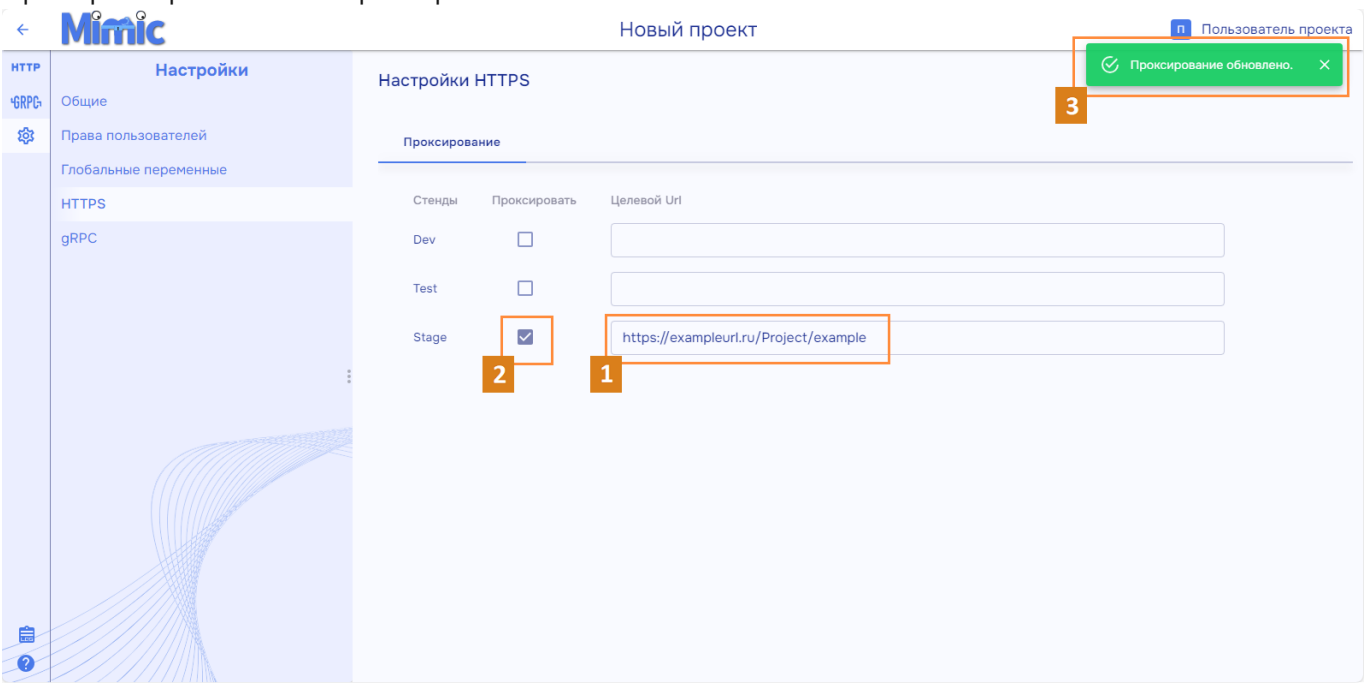
Протокол HTTPS (HyperText Transfer Protocol Secure) — это защищенная версия протокола HTTP, который используется для передачи данных в интернете.

Проксирование HTTPS - это переадресация входящего HTTPS-запроса от клиента на реальный сервер (или сервис) для получение ответа от него.

Когда прокси-сервер получает запрос, он анализирует его и определяет, куда направить этот запрос. ***Если существуют правила маршрутизации или фильтрации***, прокси будет использовать их для принятия решения о том, как обработать запрос. ***Если ни одно из правил маршрута не подходит***, прокси-сервер перенаправляет запрос на целевой URL, указанный в столбце **Целевой Url** сетевому адресу реального сервиса соответствующего контура(стенда).

Настройка блока "Проксирование"

Пример настройки блока Проксирование:



Описание действий:

1. Заполняем поле **Целевой Url** сетевым адресом сервиса, на который будет перенаправлен запрос (контуры изолированы и проксировать из одного контура в другой невозможно);
2. Устанавливаем признак **Проксировать** для соответствующей среды;
3. Сохранение происходит автоматически при потере фокуса

> **Права пользователей**
>
> Настройка проксирования HTTP доступна Владельцу и Пользователю проекта. Читателю доступен просмотр.

gRPC

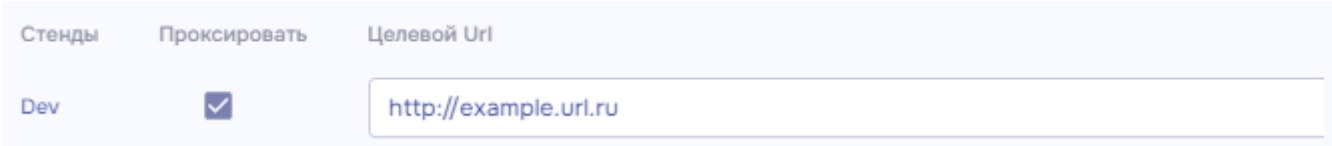
gRPC - это фреймворк для удаленного вызова процедур. Мокирование gRPC является полезным инструментом при разработке и тестировании приложений, использующих gRPC для взаимодействия между компонентами.

Про gRPC Введение в gRPC На какой URL мока настроить сервис?

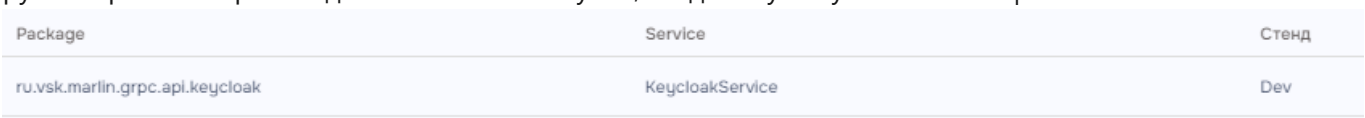
Проксирование

> Блок **"Проксирование"** в Настройках gRPC отвечает за переадресацию входящего запроса на реальный gRPC-сервис для получение ответа от него.

Здесь необходимо указать ***Целевой Url*** для каждого стенда, на который отправится запрос в случае, если ни одно из правил фильтрации не сработает:



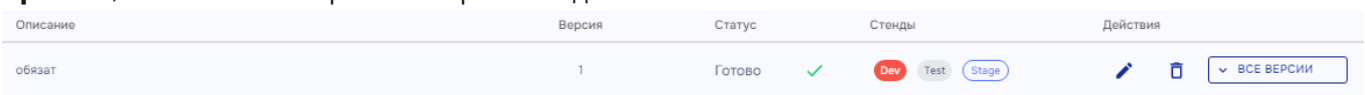
Также в этом блоке происходит заполнение полей в автоматическом режиме ***Package, Service*** и соответствующий ***стенд***, которые заполняются в результате установки соединения с помощью brotobuf-файлов. Название **Package** указывает те **части API (название метода), которые следует мокировать**, **Service - название сервиса для проксирования**. Заполнение таблицы ProtoInfo в ручном режиме происходит только в том случае, когда отсутствуют brotobuf-файлы:



Соединения

Блок **"Соединения"** позволяет установить соединение gRPC **двумя способами**: **1.** С помощью самостоятельной **загрузки файлов protobuf**, на основе которых будет производиться мокирование и проксирование сервиса. **2.** С помощью **отражения сервиса**, то есть **по рефлексии**. Рефлексия позволяет клиентам динамически обнаруживать методы и сообщения, доступные на сервере.

В данном блоке происходит устновка соединения с помощью загрузки необходимых protobuf-файлов или отражения сервиса. Также возможен **просмотр всех существующих соединений проекта**, включая более ранние версии соединений.



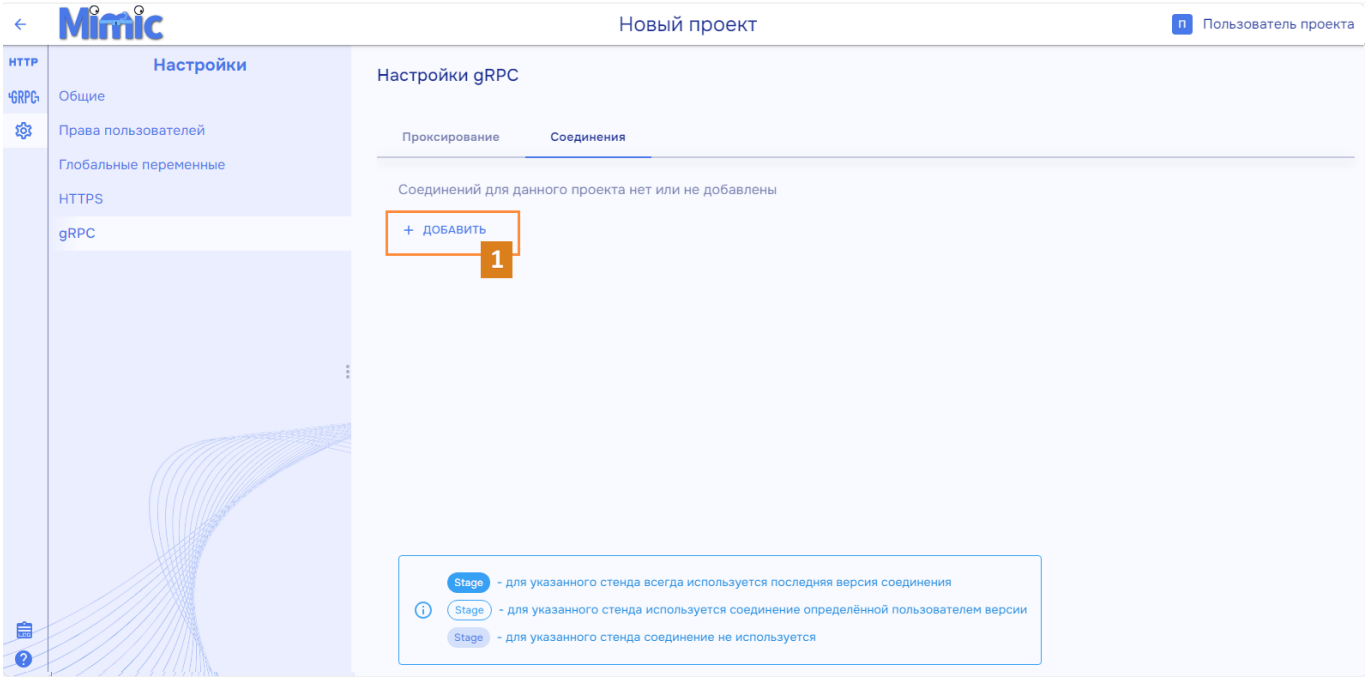
В рамках проекта **для каждого сервиса преимущественно создается индивидуальное соединение**, что позволяет упростить процесс внесения изменений. Например, если необходимо загрузить новый файл, это можно сделать только для нужного соединения, не затрагивая другие. Такой подход позволяет избежать необходимости повторной загрузки всех файлов при внесении изменений в одно соединение.

Файлы protobuf содержат описания структуры данных и методов, которые используются для обмена информацией между клиентом и сервером в gRPC. Эти файлы определяют формат сообщений, которые могут быть отправлены и приняты между клиентом и сервером. И на основе загруженных файлов protobuf в подключениях создается компонент, который реализует взаимодействие между сервисом Mimic (проксирующим сервисом) и вызывающим сервисом.

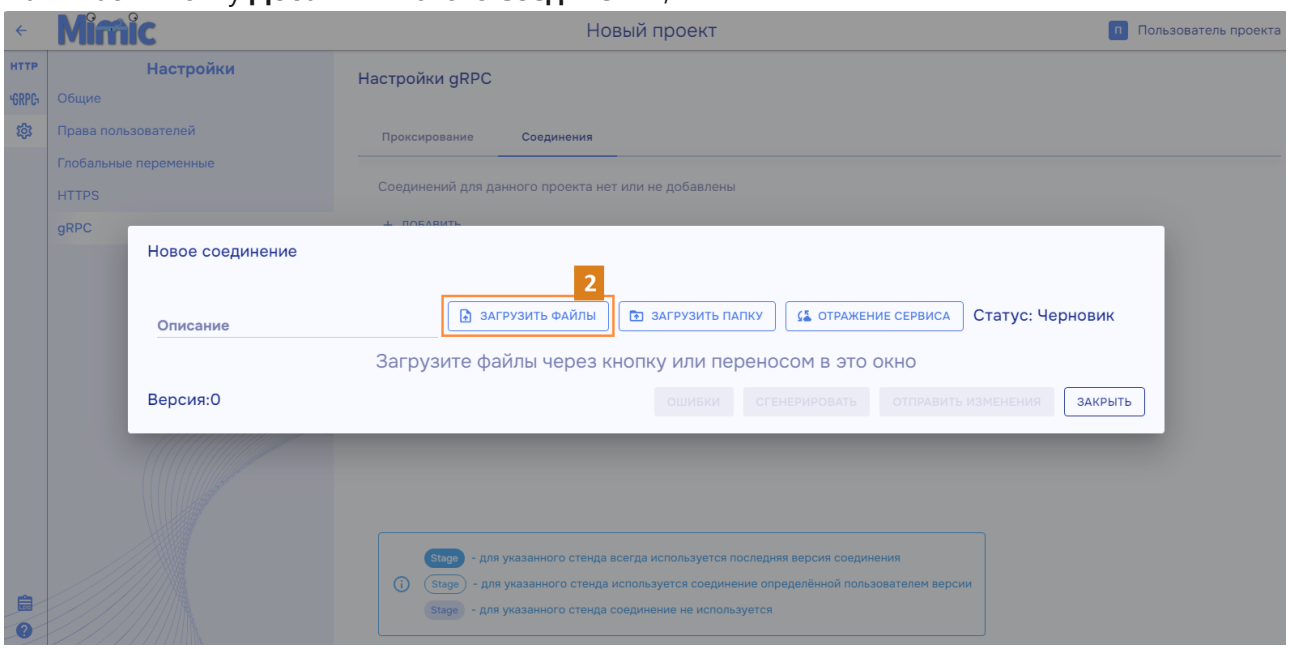
И на основе созданного соединения в таблице **ProtoInfo** блока **"Проксирование"** автоматически появляется информация с Package+Service данного сервиса, при удалении соединения запись также автоматически удаляется.

Создание gRPC-соединения с помощью загрузки protobuf

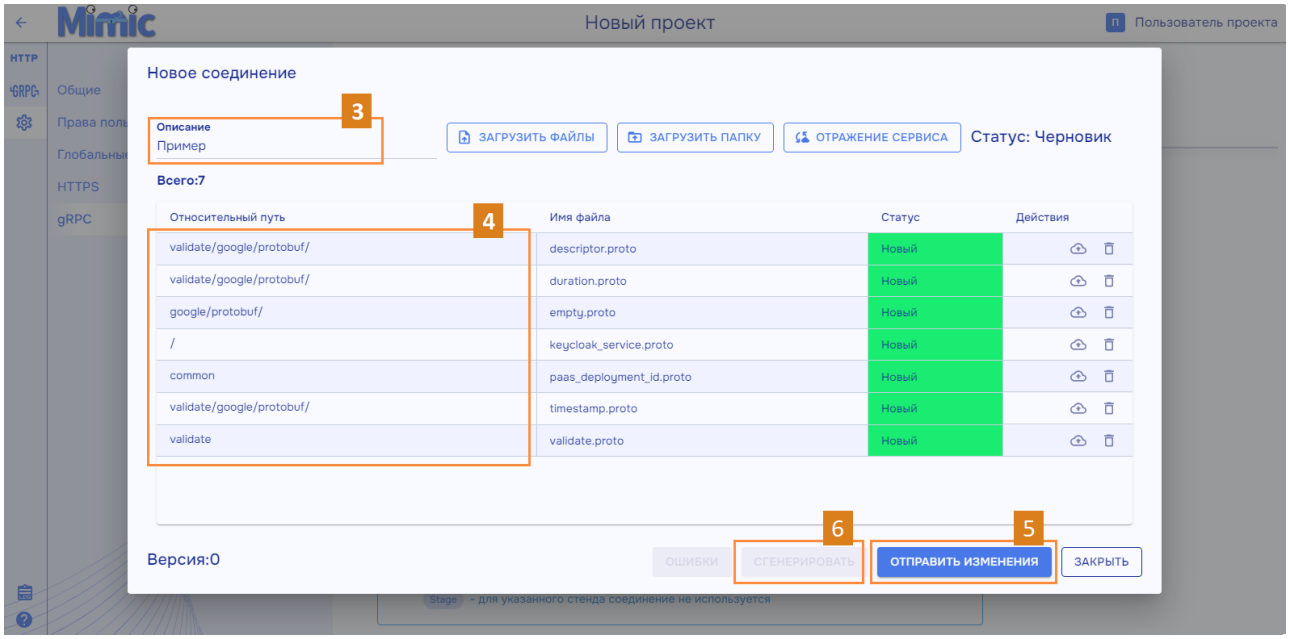
Пример создания соединения Описание действий:



1. Нажимаем кнопку **Добавить** в блоке **Соединения**;

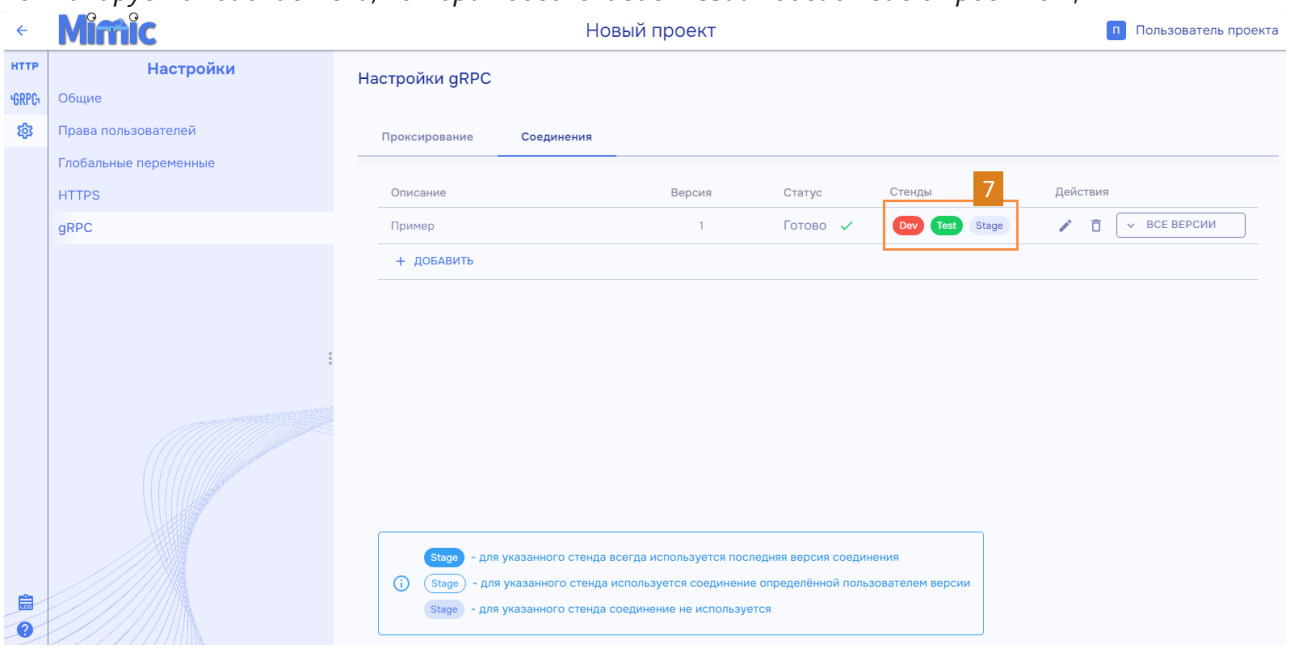


2. **Выполняем загрузку** необходимых файлов или папки с proto-файлами;

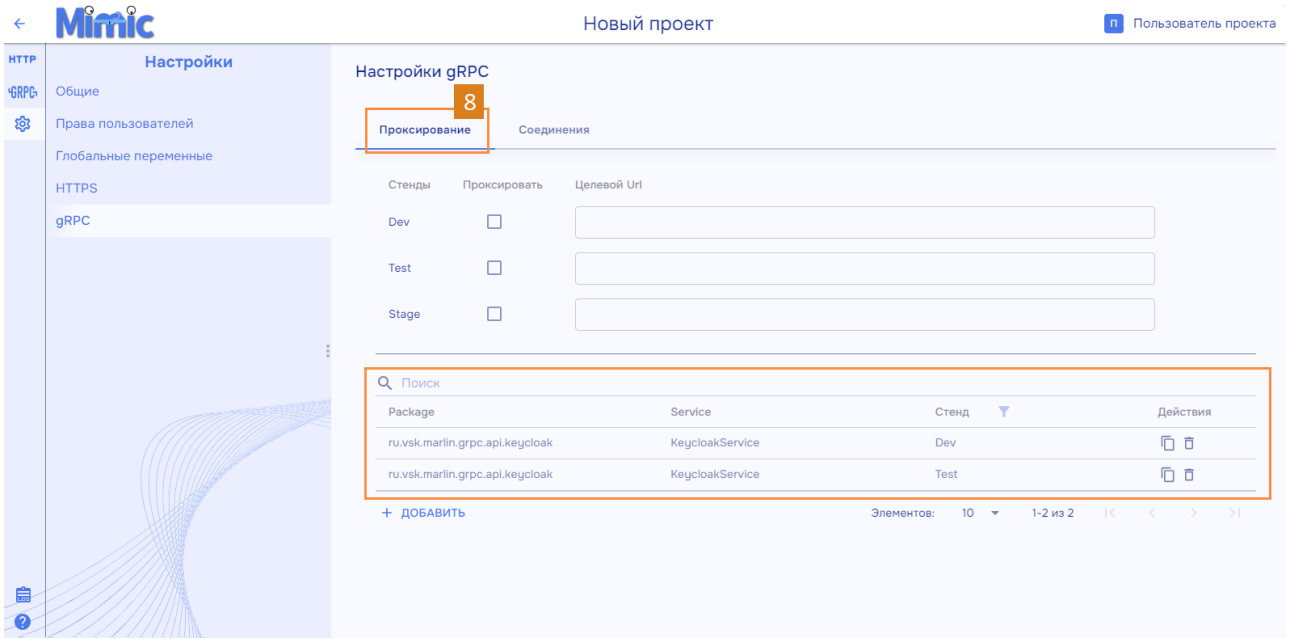


3. Указываем **Описание** подключения, после чего оно оказывается в статусе "Черновик";
4. При необходимости меняем **Относительный путь** файлов;
5. **Подтверждаем действия** нажатием кнопки 

;
6. Нажимаем кнопку **"Сгенерировать"** *При нажатии кнопки "Сгенерировать" из proto-файлов компилируется библиотека, которая обеспечивает взаимодействие с проектом*;



7. Если генерация соединения прошла успешно,то **привязываем нужную версию соединения к выполняемой среде**;



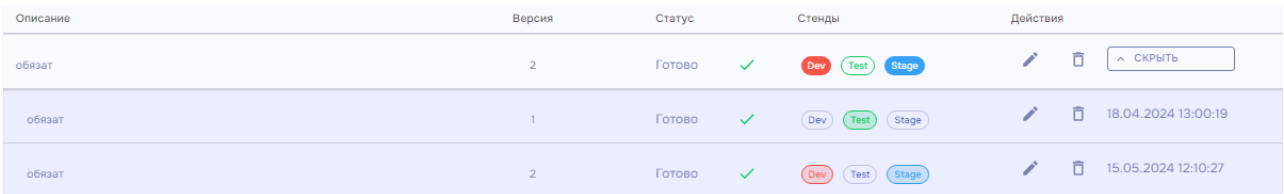
8. В блоке **Проксирование** проверяем, что таблица с **ProtoInfo** успешно обновлена; **При удалении соединения** запись из таблицы **ProtoInfo** также автоматически будет удалена.

Дополнительная информация:

- Если при подключении возникли **ошибки**, то появляется соответствующая иконка 

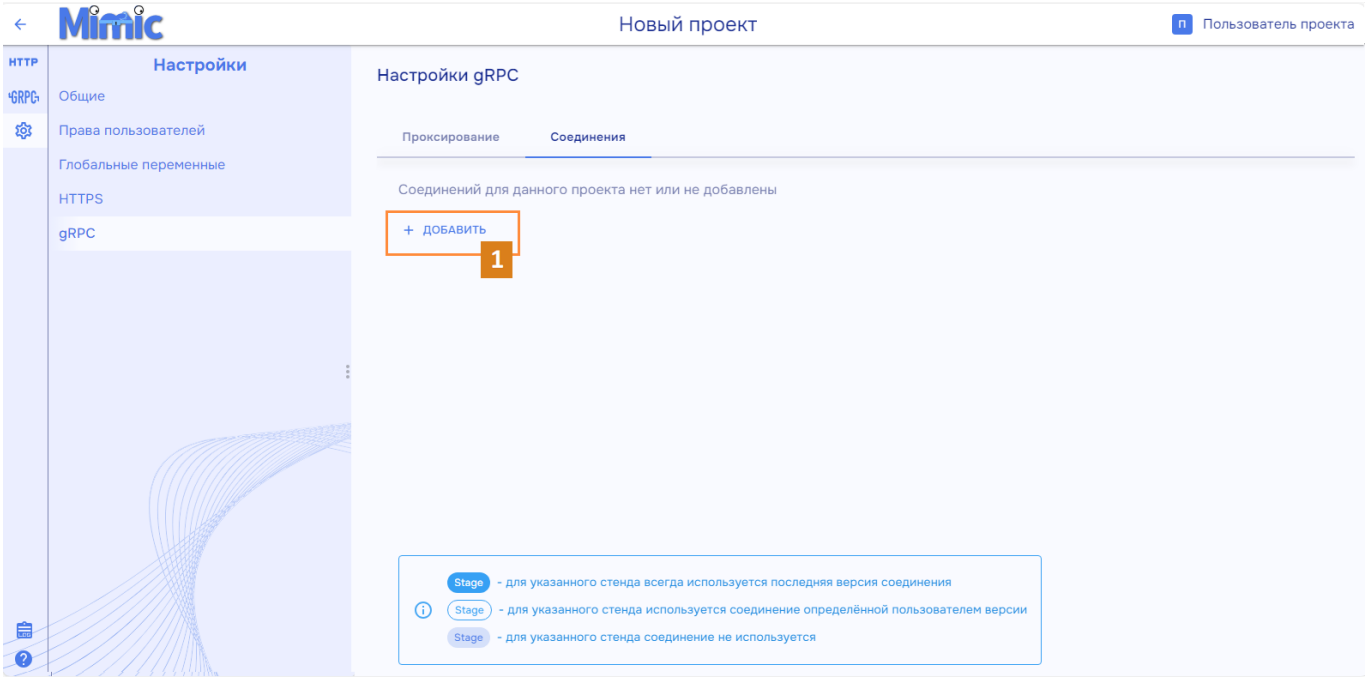
, при нажатии на которую отображается подробная информация об ошибках.
- Проверка **состояния подключения**:  - подключение успешно  - ошибка
- Также возможен просмотр **всех версий** соединений путем нажатия на кнопку 

.

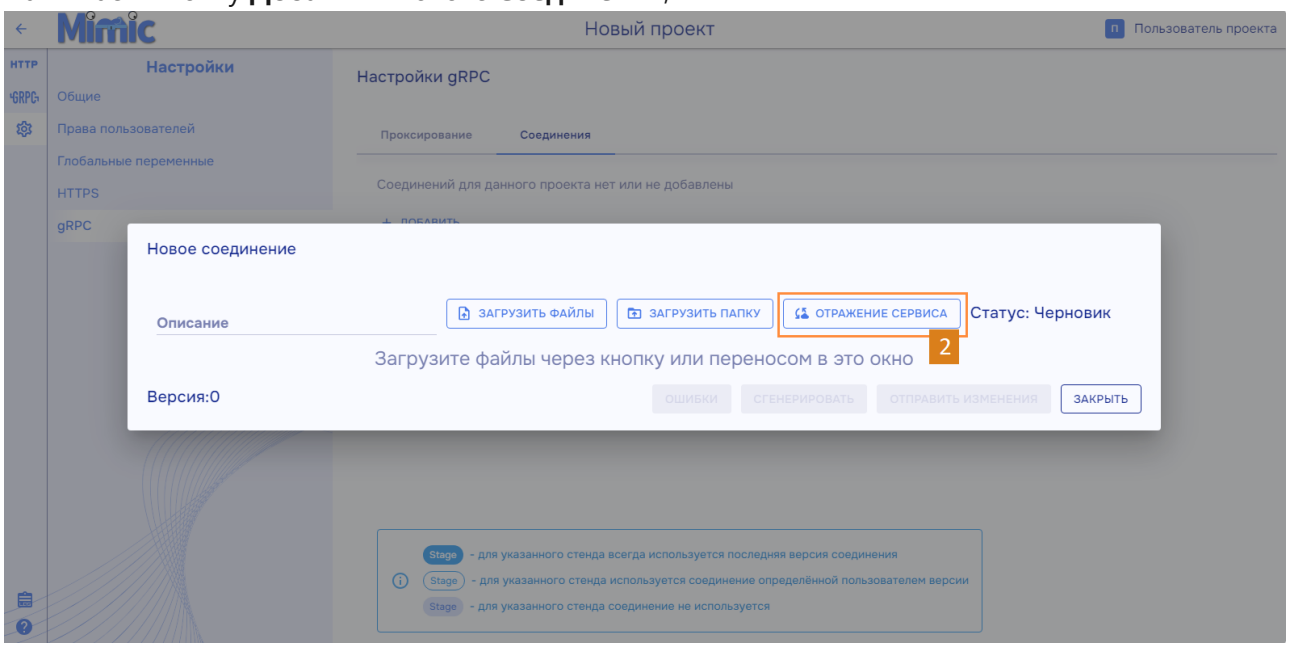


Создание gRPC-соединения с помощью отражения сервиса

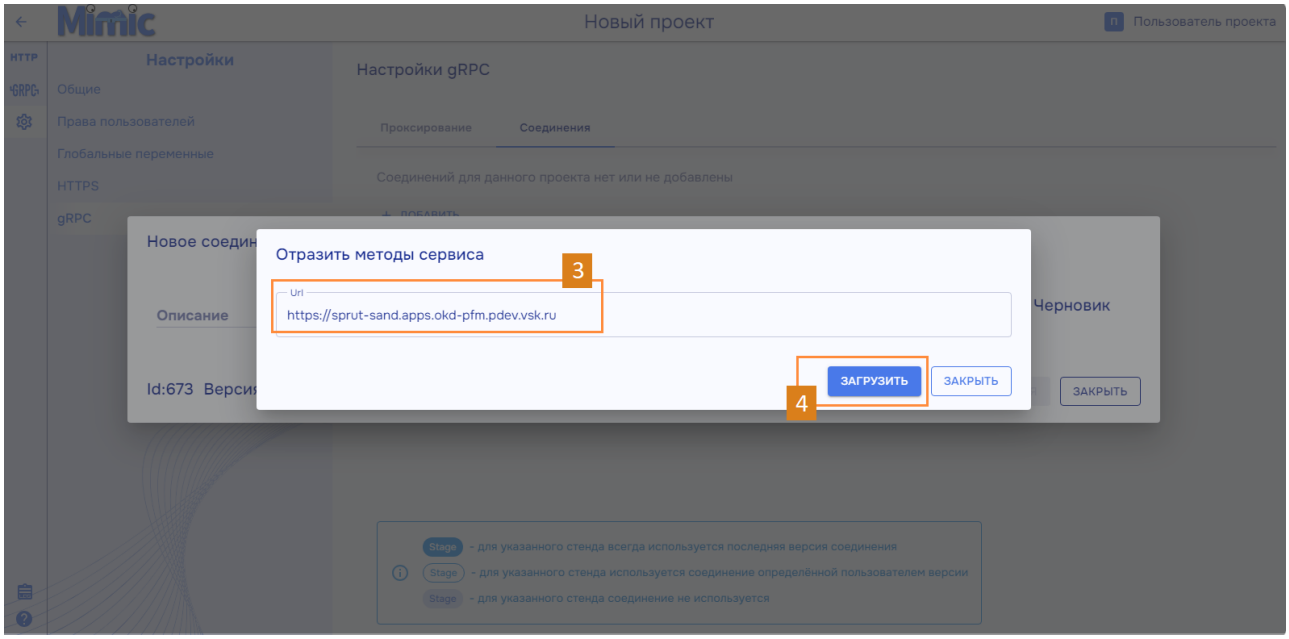
Пример создания соединения Описание действий:



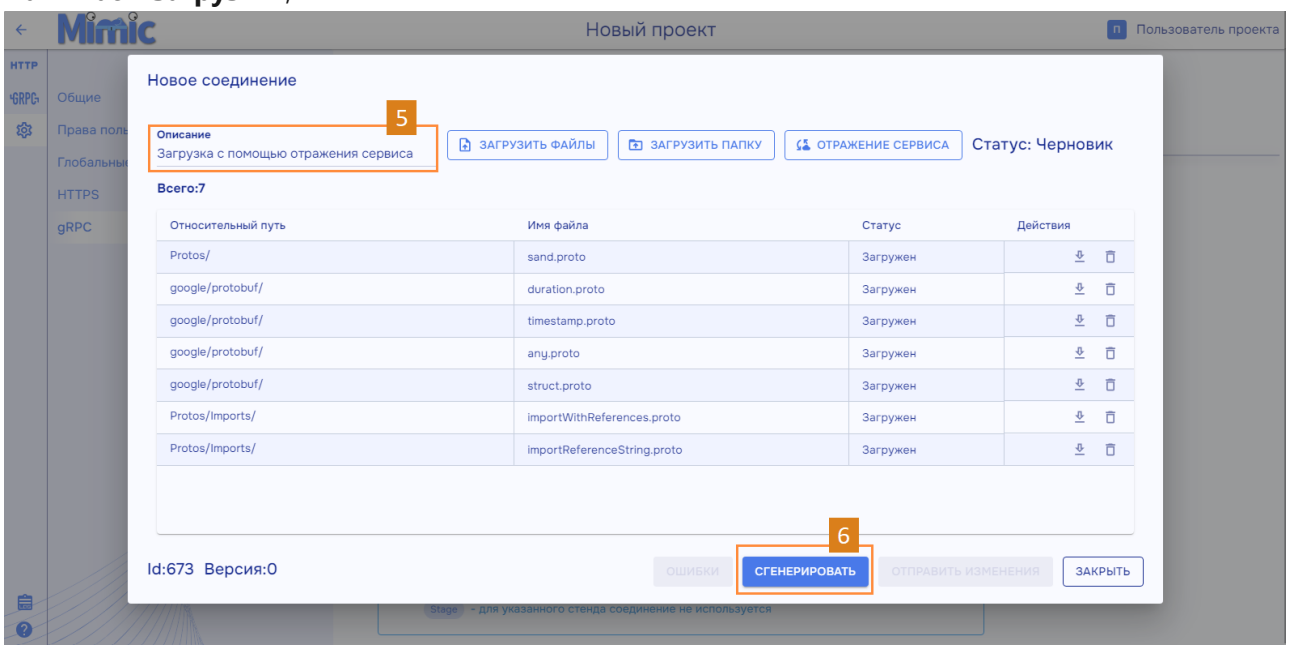
1. Нажимаем кнопку **Добавить** в блоке **Соединения**;



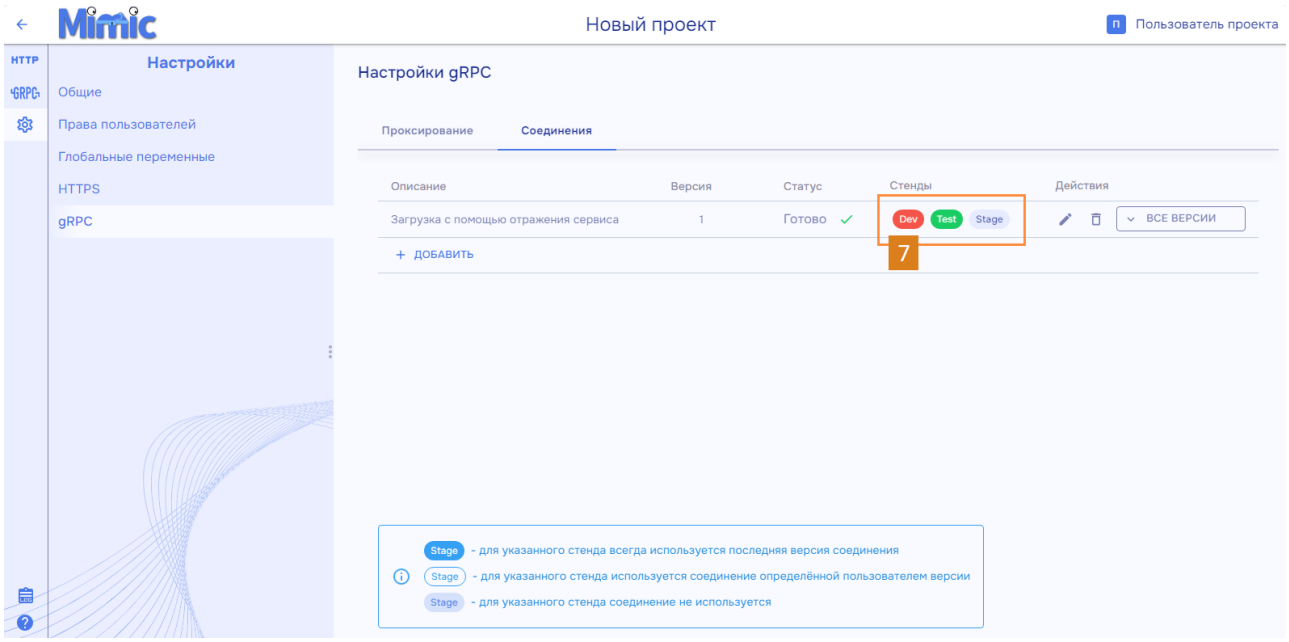
2. Нажимаем кнопку **Отражение сервиса**;



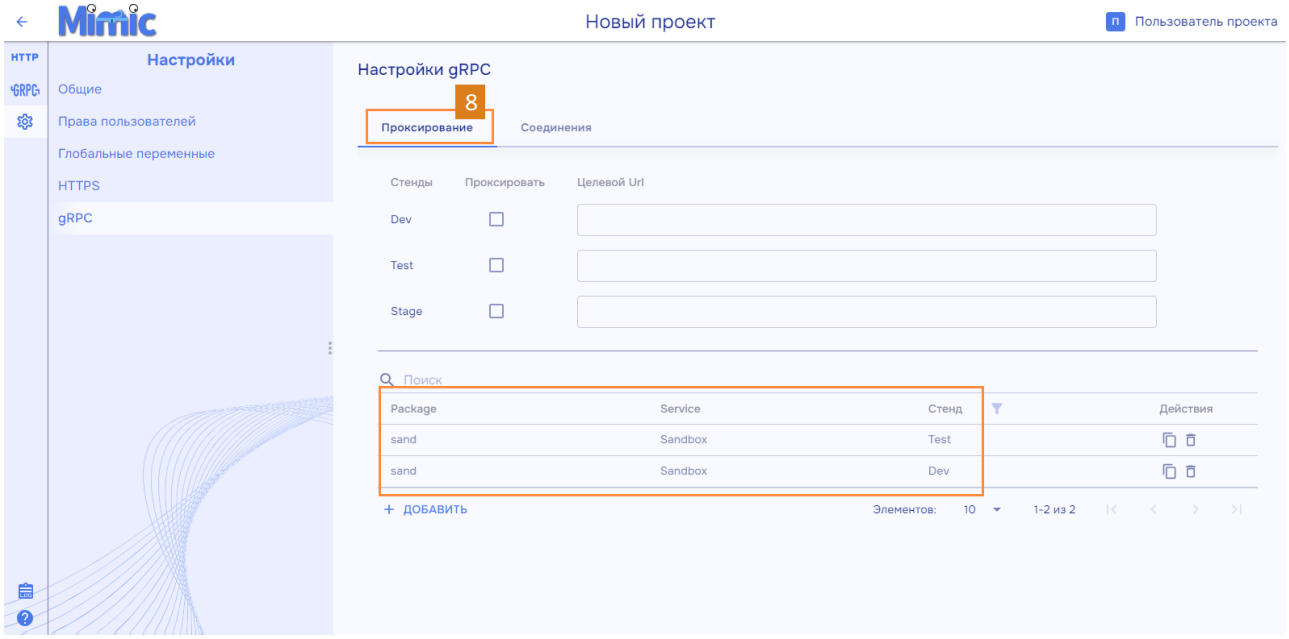
3. В появившемся окне вводим **Url** необходимого сервиса. Подробнне о том, на какой url мока настроить сервис;
4. Нажимаем **Загрузить**;



5. Вводим **Описание** для нового grpc-соединения;
6. Нажимаем кнопку **"Сгенерировать"** *При нажатии кнопки "Сгенерировать" из proto-файлов компилируется библиотека, которая обеспечивает взаимодействие с проектом*;



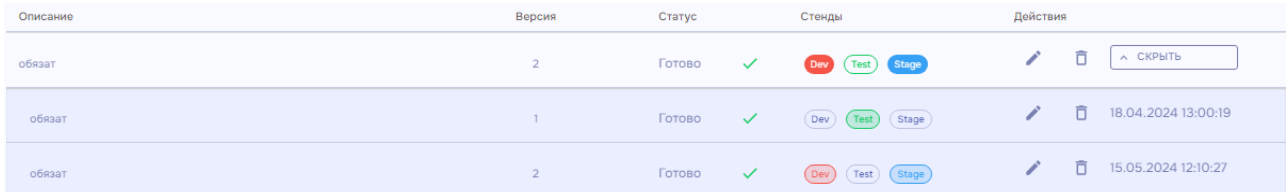
7. Если соединение успешно, то выбираем **стенды** для данного соединения;



8. В блоке **Проксирование** проверяем, что таблица с **ProtoInfo** успешно обновлена. **При удалении соединения** запись из таблицы **ProtoInfo** также автоматически будет удалена;

Дополнительная информация:

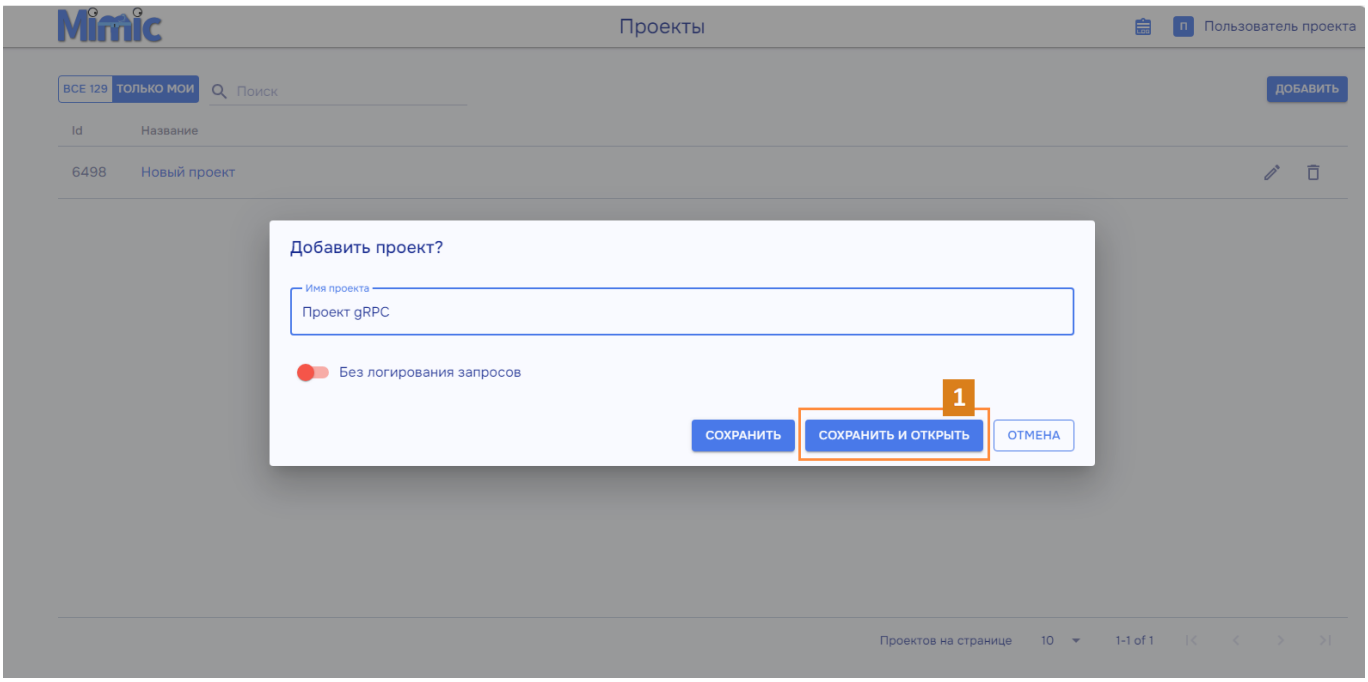
- Если при подключении возникли **ошибки**, то появляется соответствующая иконка ОШИБКИ, при нажатии на которую отображается подробная информация об ошибках.
- Проверка **состояния подключения**: ✓ - подключение успешно ✗ - ошибка
- Также возможен просмотр **всех версий** соединений путем нажатия на кнопку ВСЕ ВЕРСИИ .



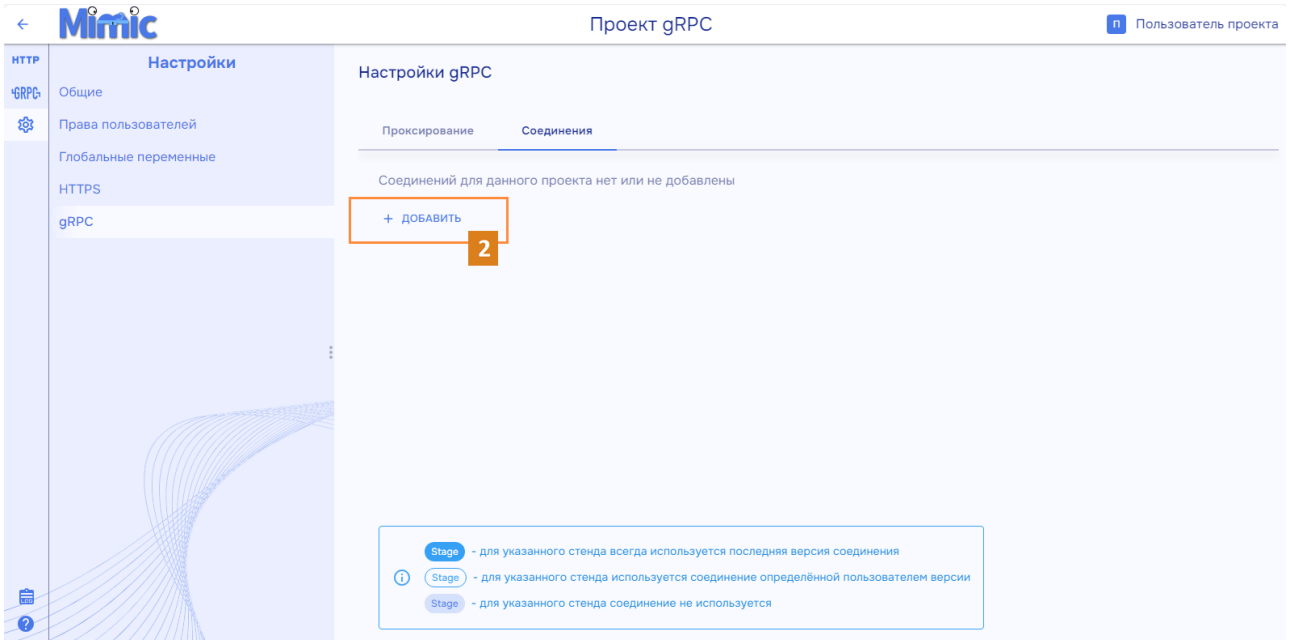
Пример работы с gRPC

Создание проекта, установка соединения gRPC, добавление метода

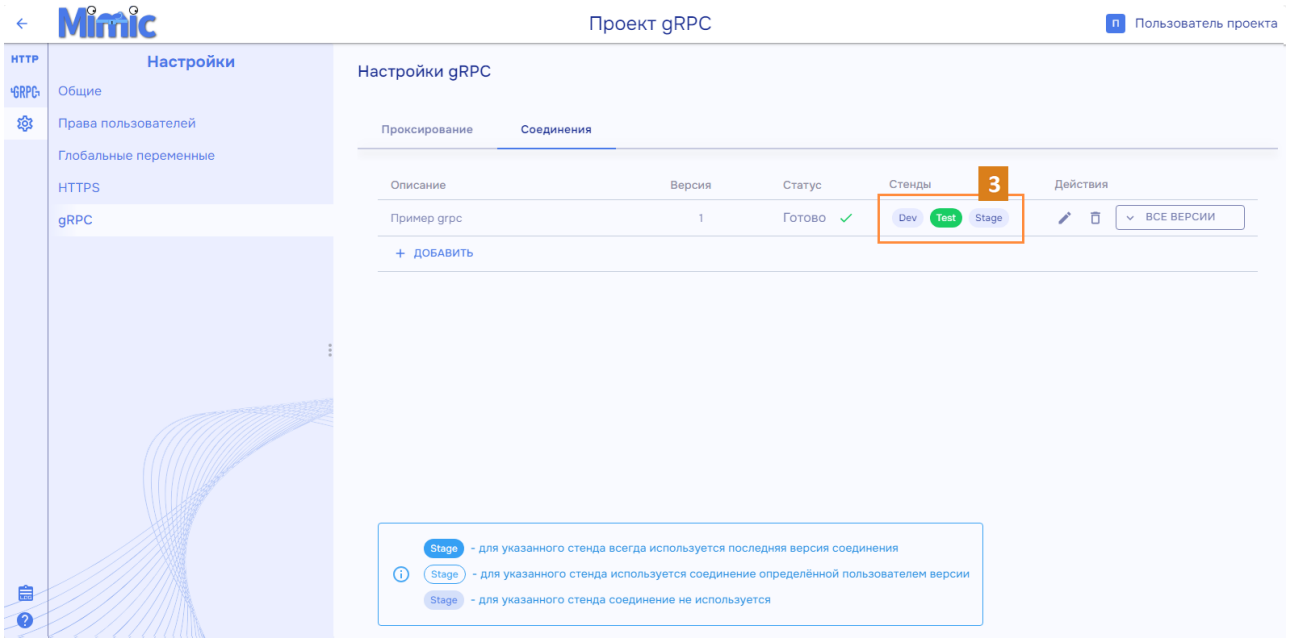
Описание действий:



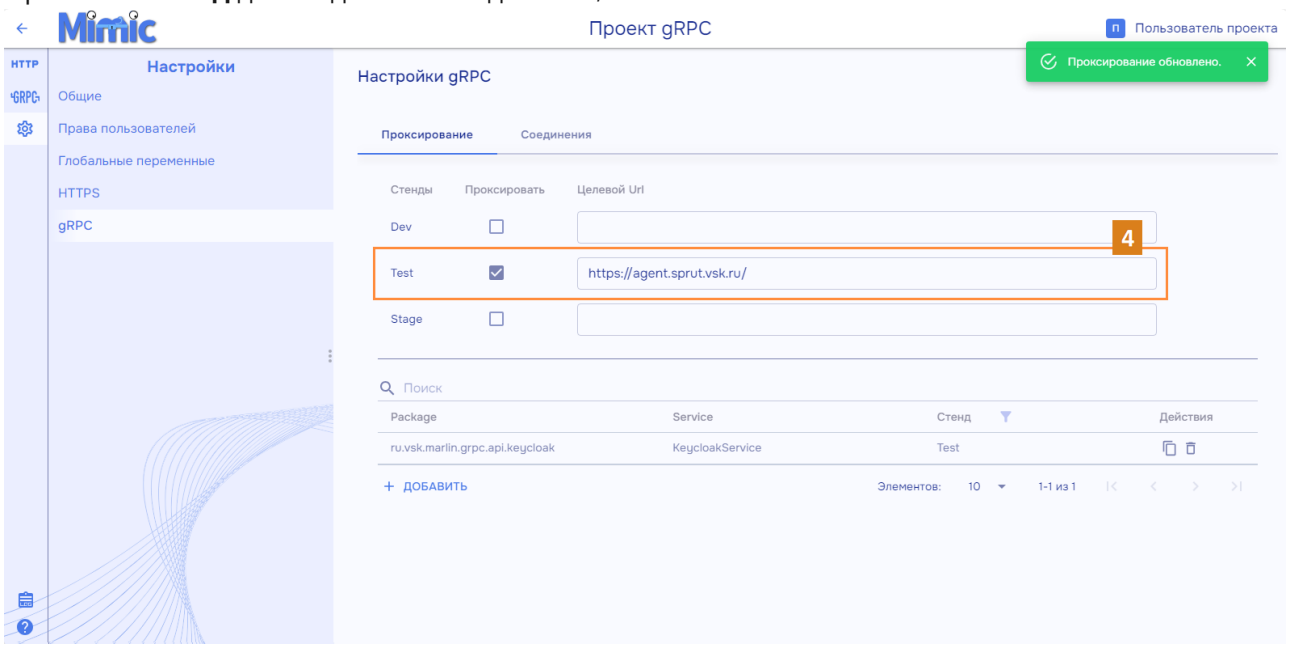
1. **Создадим проект** (подробнее в разделе Проект);



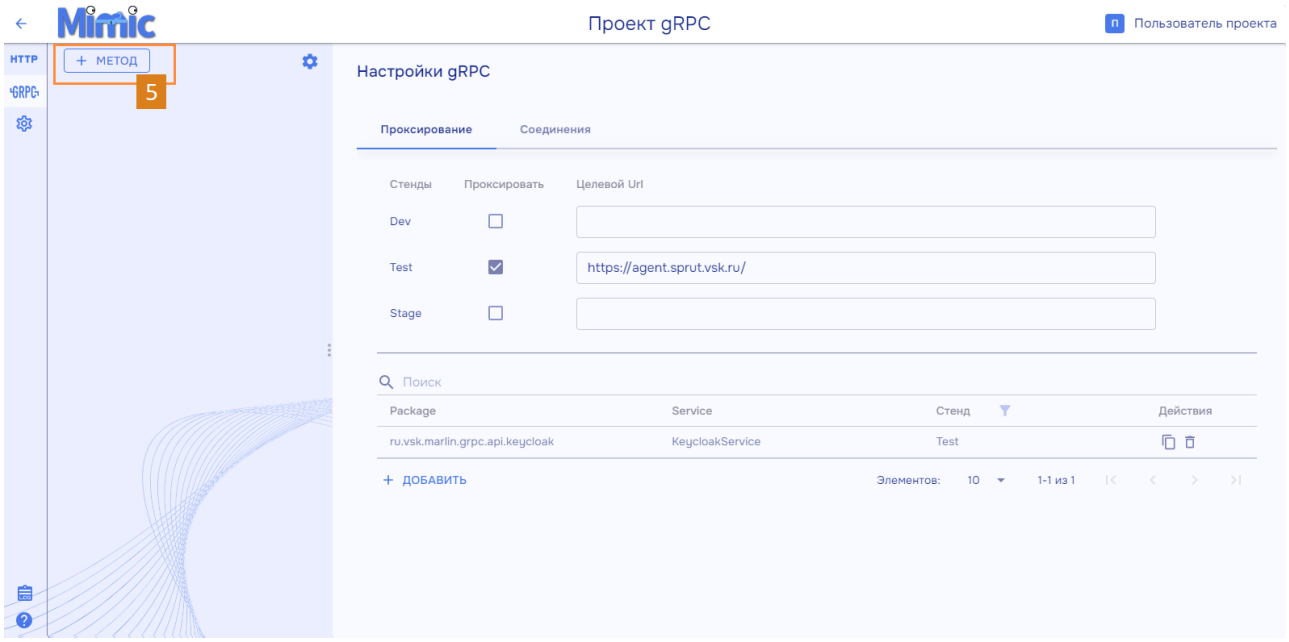
2. **Установим соединение gRPC** во вкладке "Соединения" (подробнее выше глава "Создание gRPC-соединения");



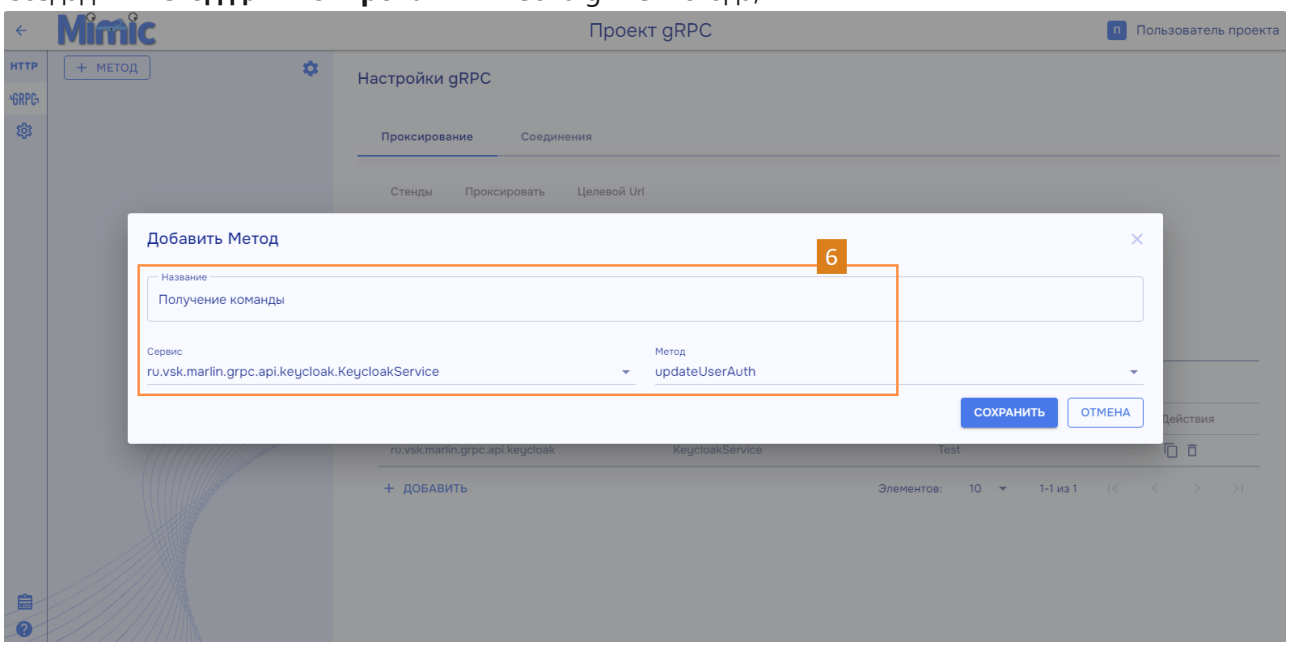
3. Проставим **стенд** для созданного соединения;



4. На вкладке "Проксирование" введем **URL для проксирования** и проставим соответствующую галочку для проксирования url;



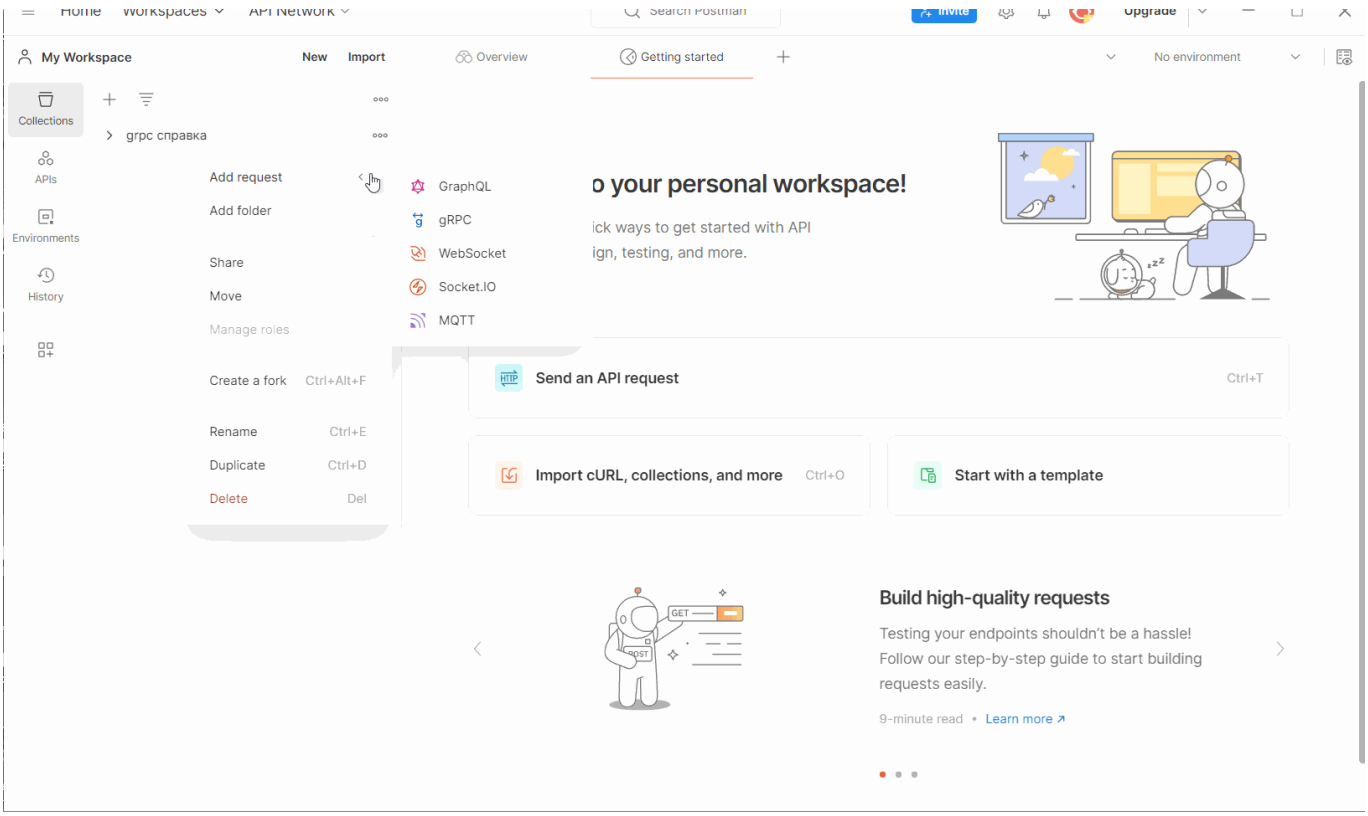
5. Создадим **метод для мокирования** вызова gRPC-метода;



6. Заполним **Название**, выберем **Сервис** и **Метод** из предложенного списка, далее сохраним созданнный метод.

Для проверки мока будем использовать Postman, ниже показана его преднастройка:

Преднастройка Postman для тестирования gRPC

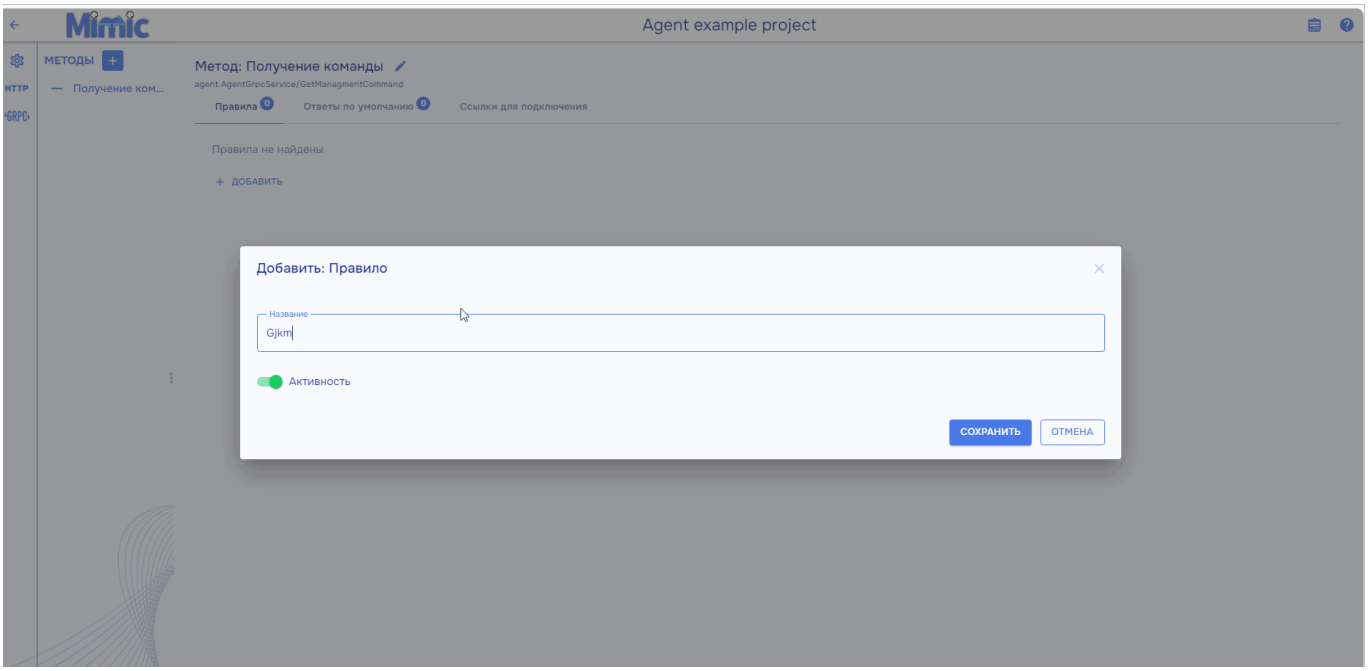


Описание действий:

1. В своей коллекции **создаем gRPC-запрос** посредством нажатия на кнопку ⋯;
2. Импортируем proto-файл с помощью нажатия кнопки **Import a .proto file**;
3. **Импортируем как Api**, вводим соответствующее название для Api;
4. В поле Select a method **выбираем необходимый метод**;
5. В поле Enter Url **вводим необходимый Url**. Подробнне о том, на какой url мока настроить сервис;
6. Закрываем замок и **отправляем запрос** с помощью кнопки **Invoke**.

Создадим правило в интерфейсе Mimic и проверим его с помощью Postman:

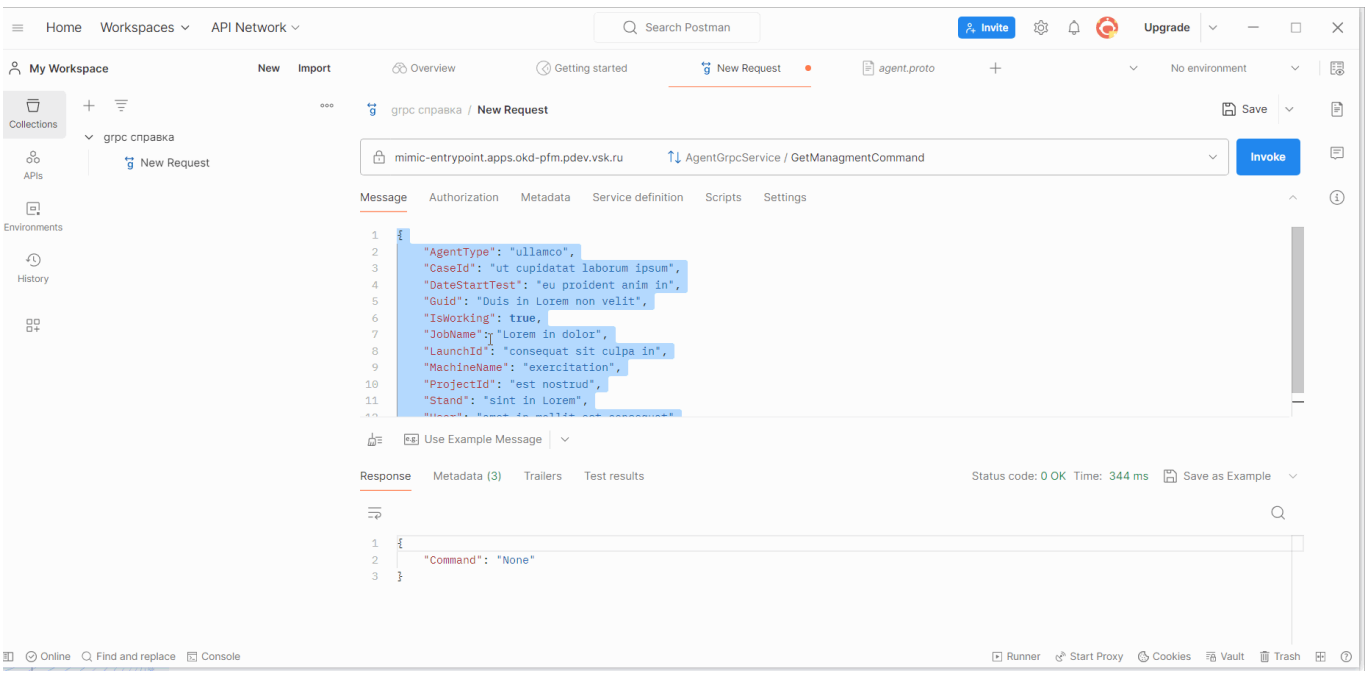
Создание правила с фильтром, локальной переменной и телом ответа



Описание действий:

1. Создаем **новое правило** для текущего метода;
2. Добавляем **фильтр** (для gRPC-правил доступны фильтры только по телу запроса): задаем наименование поля при помощи локатора и само значение поля (которое должно быть для срабатывания фильтра);
3. Создаем **локальную переменную** во вкладке "Переменные" и копируем ее название при помощи соответсвующей иконки;
4. Задаем **тело ответа** с использованием скопированной переменной. При необходимости ставим задержку при ответе.

Проверка срабатывания правила при помощи Postman



Описание действий:

1. При помощи кнопки "Use example message" создаем **пример сообщения** (сообщение **не попадает под критерии правила**) в Postman;

2. При помощи кнопки "Invoke" **отправляем запрос** -> **Правило не сработало**, поэтому запрос ушел в реальный сервис;
3. **Меняем запрос** так, чтобы он **попадал под критерии** созданного правила;
4. При помощи кнопки "Invoke" **отправляем запрос** -> **Правило сработало**, получаем заранее заданное тело ответа;
5. Отправляем **запрос еще раз** -> Получаем **новый ответ** (так как была использована переменная для случайной генерации символов);
6. В **Журнале** проверяем запросы, выполненные для данного проекта. **Для запросов, которые ушли на реальный сервер** (правило не сработало),возможно посмотреть только сам запрос без ответа. **Для тех запросов, для которых правило сработало**, возможен просмотр и запроса, и ответа.

Подробнее про работу с методами gRPC Методы (gRPC).

> **Права пользователей**
>
> Настройка проксирования gRPC доступна Владельцу и Пользователю проекта. Читателю доступен просмотр.

Kafka

Kafka — это распределенная платформа потоковой передачи сообщений, предназначенная для обработки больших объемов информации в реальном времени.

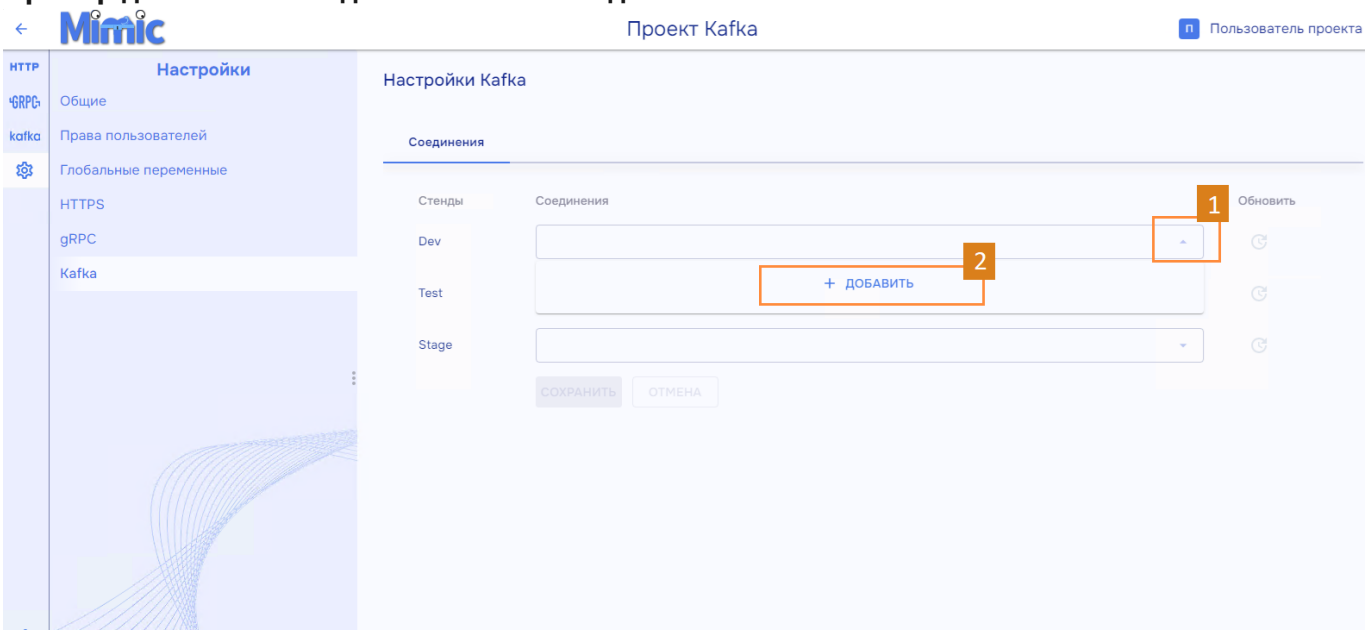
Соединения

Блок **"Соединения"** позволяет добавить соединение с брокером Kafka на необходимый **стенд**:

1. Вы можете создать **новое соединение**, указав хост (адрес брокера) и порт (номер порта, на котором работает брокер Kafka).
2. Если у вас уже есть **сохраненные соединения**, вы можете выбрать одно из них из списка.

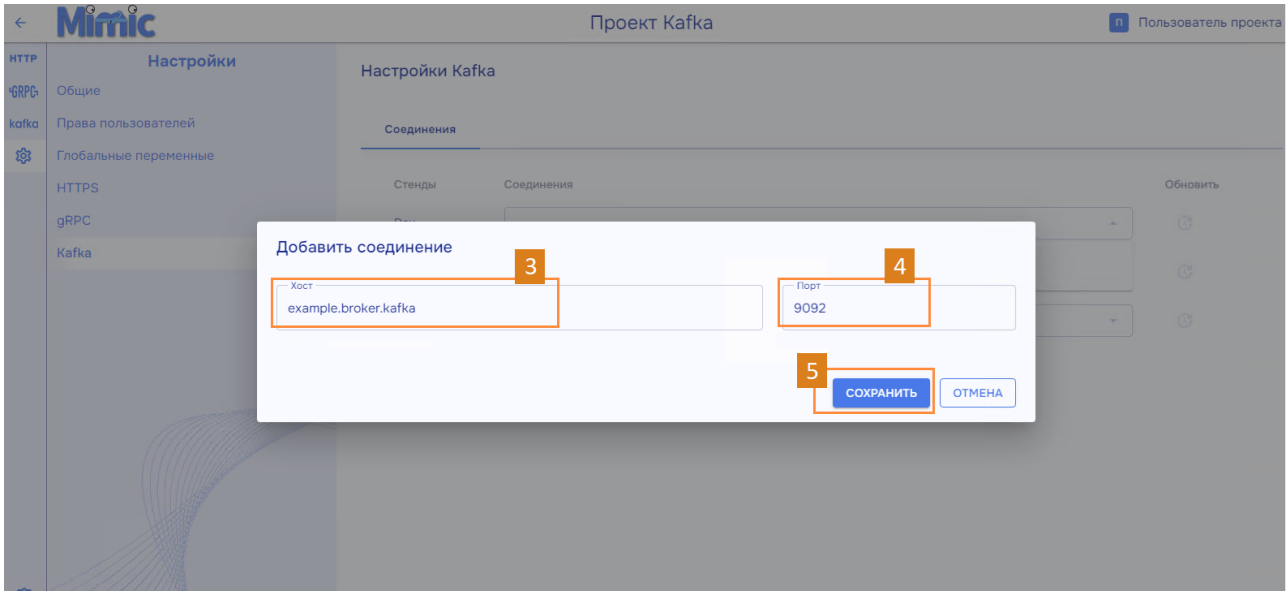
Добавлние Kafka-соединения

Пример добавления соединения Описание действий:

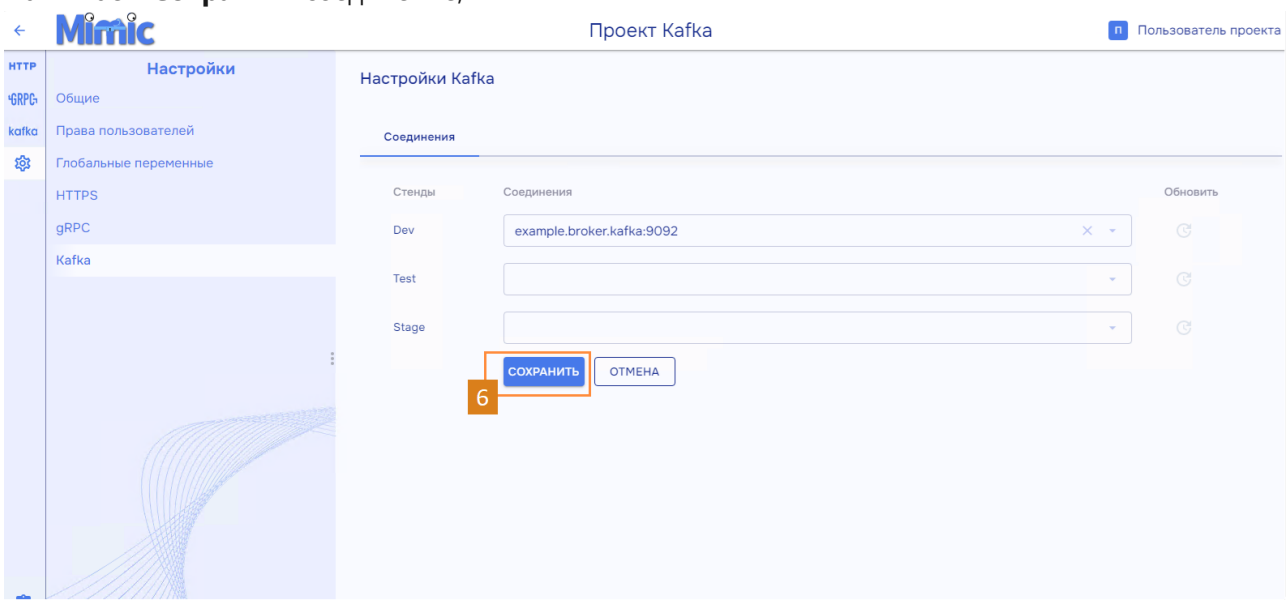


1. На нужном стенде **раскрываем список соединений**;

2. Нажимаем кнопку **Добавить**;



3. Указываем **хост** (адрес) брокера;
4. Указываем **порт** брокера Kafka;
5. Нажимаем **Сохранить** соединение;



6. **Сохраняем** изменения. При сохранении нового соединения запускается процесс создания топиков в системе Kafka.

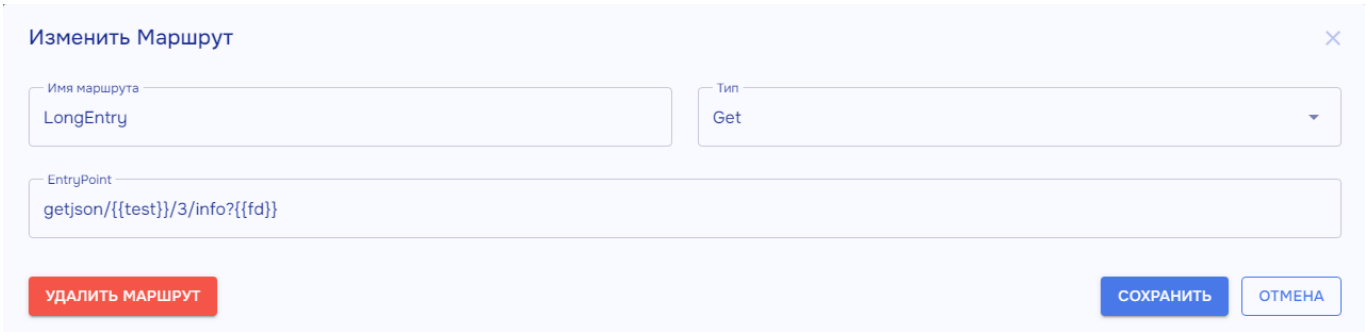
> **Права пользователей**
>
> Настройка проксирования Kafka доступна Владельцу и Пользователю проекта. Читателю доступен просмотр.

Маршруты (HTTPS)

> **Маршрут** - это сущность мока (EndPoint) HTTPS, в рамках которой пользователь может делать настройки для нужного отображения ответа по созданным правилам.

Страница маршрута включает в себя следующие блоки: **Редактирование маршрута**, **Правила**, **Ответы по умолчанию**, **Ссылки для подключения**, **Настройки HTTPS**.

Редактирование маршурта позволяет изменить **имя маршрута**, **его тип** и **точку входа** (EntryPoint):



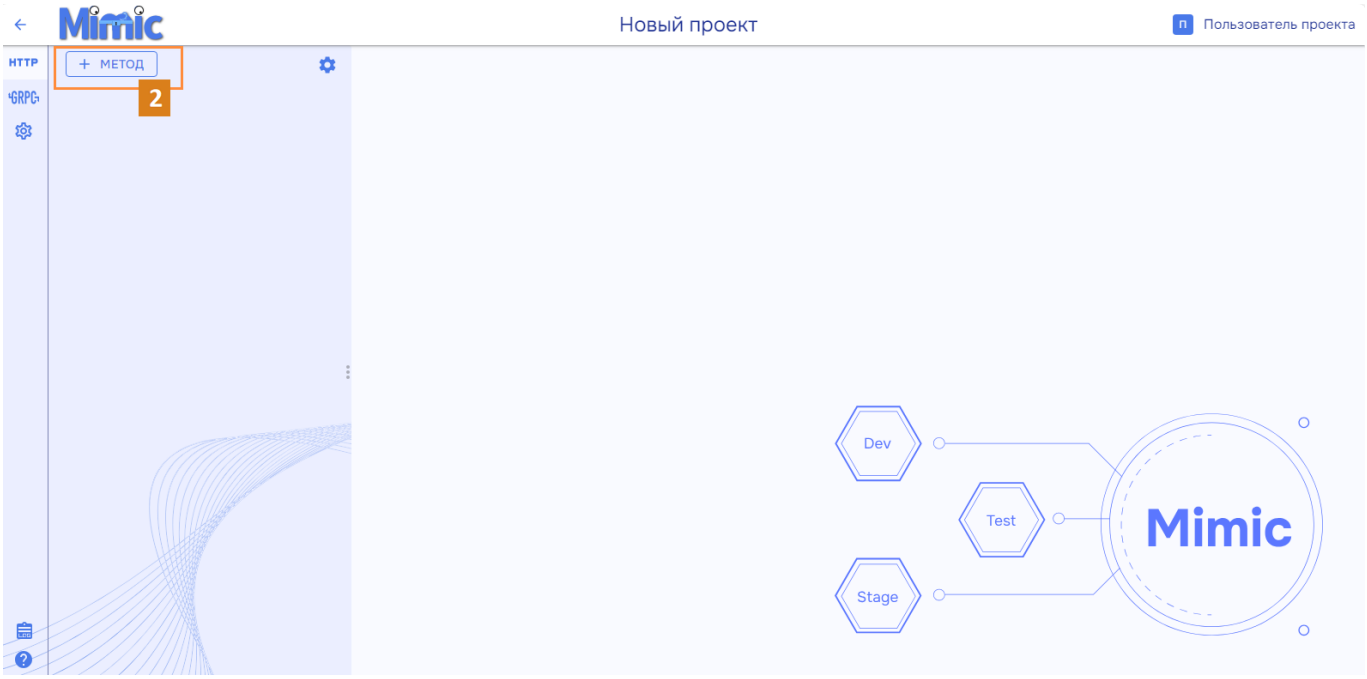
Добавление маршрута

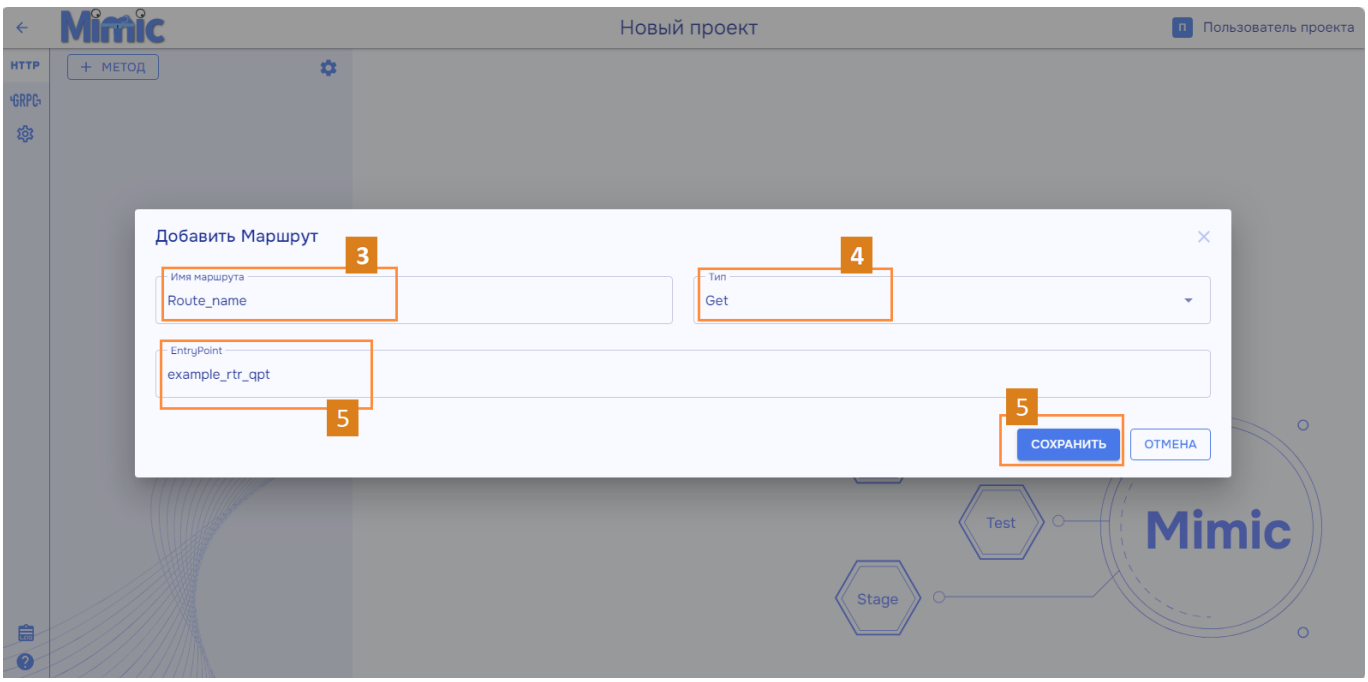
Добавление маршрута может быть выполено ***двумя способами***: ***1)*** путем добавление нового маршрута с заполнением всех соответствующих полей; ***2)*** путем добавления нового маршрута на основе уже существующего - копирование маршрута.

> Если у проекта отсутствуют маршруты, то создание нового возможно только первым способом - с заполением всех полей.

В остальных случаях возможен второй способ, то есть создание маршрута на основе уже существующего (копирование) с его дальнейшим редактированием. При копировании маршрута создаются и правила.

1. Добавление нового маршрута с заполнением полей Пример добавления маршрута:





Описание действий

1. Открыть проект, в который необходимо добавить маршрут;
2. Нажать кнопку добавления маршрута, располагающуюся в верхней части меню маршрутов;
3. Указать **Имя маршрута**;
4. Указать **Тип запроса**:

- **Get** - получение информации или доступ к данным ресурса по URL, расположенным на сервере
- **Post** - добавление или замена данных
- **Put** - обновление существующих на сервере ресурсов
- **Patch** - частичное изменение ресурса
- **Delete** - удаление данных

5. Указать **EntryPoint** (точка входа) В url он будет добавлен через "/" как показано на рисунке ниже:

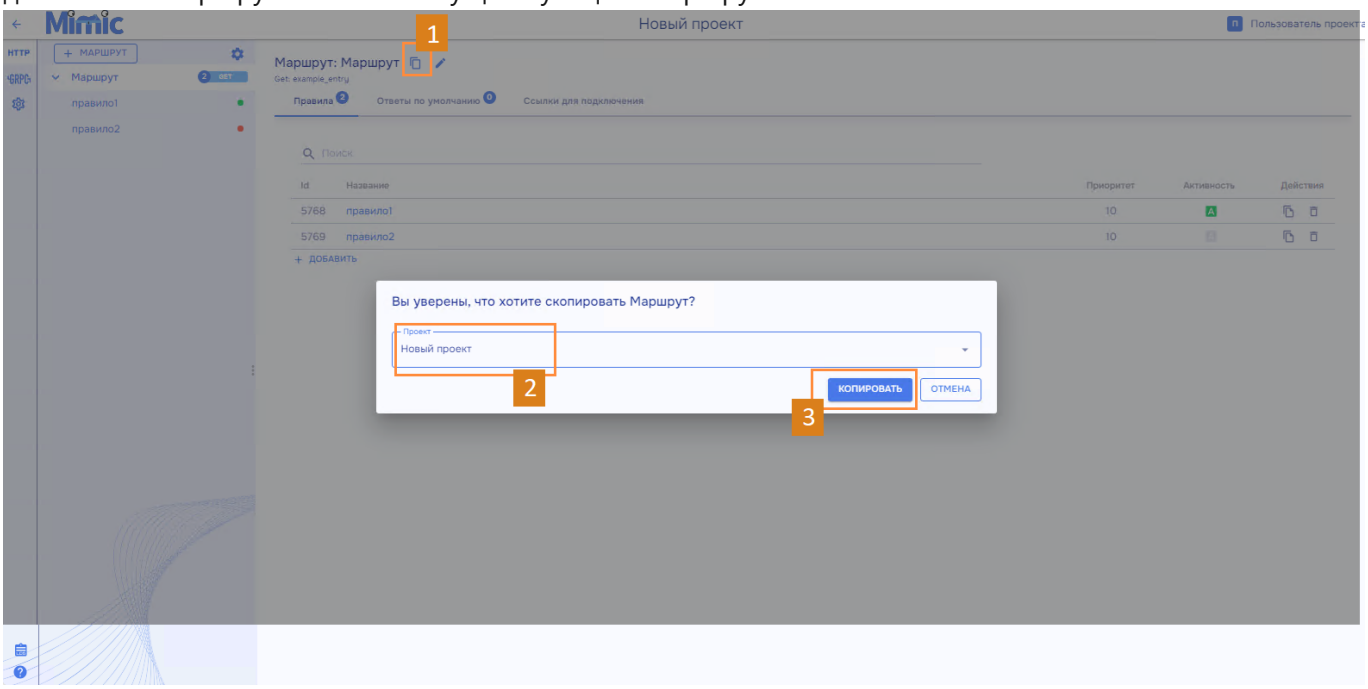
```
https://mimic-entrypoint.apps.okd-pfm.pdev.vsk.ru/220/createleonid
---------- ------------------------------------------ ---- ----------
1          2                                          3    4
```

1. Протокол
2. Доменное имя
3. ProjectId (ID проекта в Mimic)
4. **EntryPoint**

Данную строку можно найти в блоке **Urls** на странице маршрута

6. Сохранить маршрут.

2. Добавление маршрута на основе существующего (копирование маршрута) Пример добавления маршрута на основе существующего маршрута:



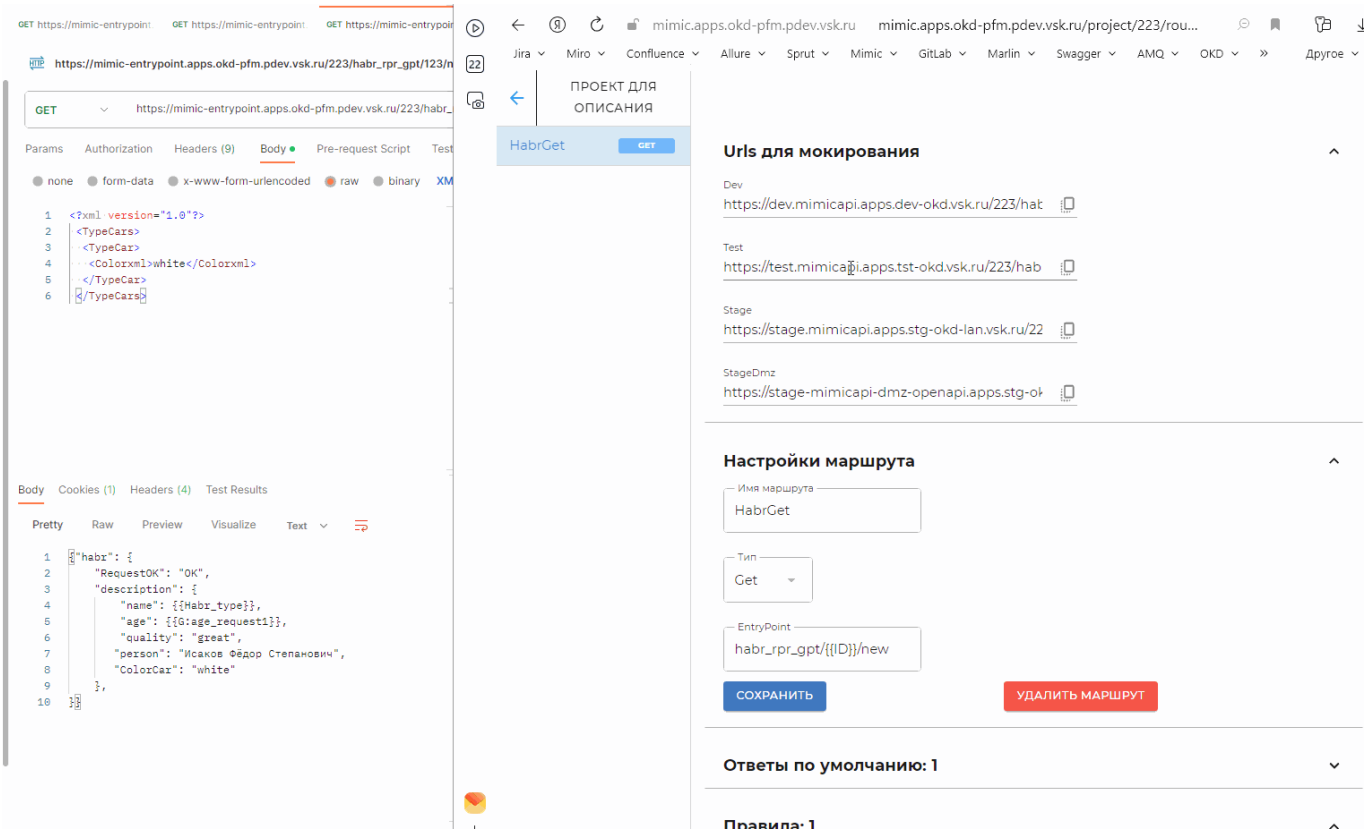
Описание действий

1. На странице необходимого для копирования маршрута нажимаем кнопку **Копировать** ;
2. Выбираем **Проект**, в который необходимо скопировать данный маршрут;
3. Нажимаем кнопку **Копировать**;
4. Открываем **созданный маршрут**;
5. Переходим в режим **редактирования** маршрута;
6. Вносим **необходимые изменения**.

На какой URL мока настроить сервис?

> **Urls** - это адреса подключения к созданному моку конкретного маршрута. Они автоматически генерируется с разбитием по контурам и позволяют скопировать значения для возможности указать в настройках тестируемого сервиса.

Пример использования Urls для мокирования:



Описание действий

1. Выбираем нужный контур
2. Копируем адрес подключения
3. Вставляем адрес в настройки тестируемого сервиса или Postman (так мы будем обращаться в мок)
4. Если есть переменные в адресе, например {{Id}}, заменяем их на конкретные значения.

> **Права пользователей**
>
> Создавать маршруты и управлять ими могут Владелец проекта и Пользователь, просмотривать - Читатель.

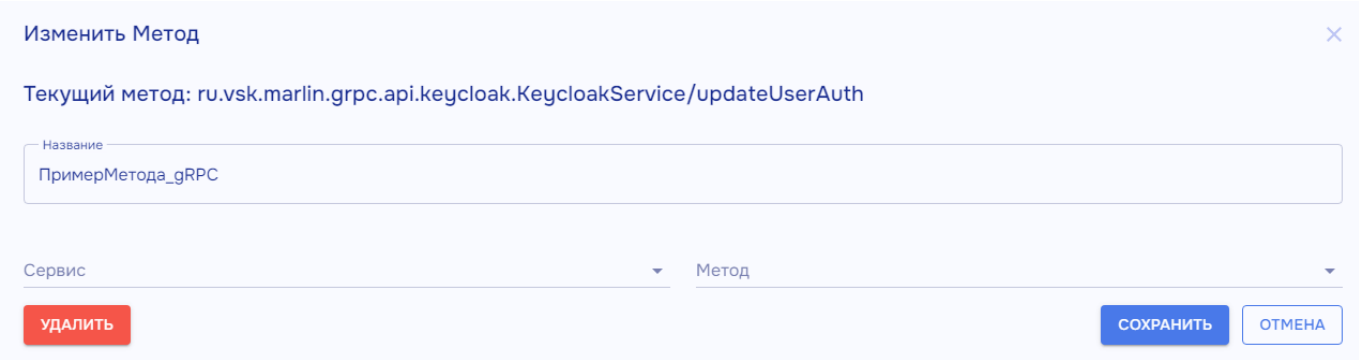
Методы (gRPC)

> **Метод** - это сущность мока (EndPoint) gRPC, в рамках которой пользователь может делать настройки для нужного отображения ответа по созданным правилам.

Добавление метода gRPC становится доступно только после установки gRPC-соединения (подробнее в разделе Настройки. gRPC)

Страница метода включает в себя следующие блоки: **Редактирование метода**, **Правила**, **Ответы по умолчанию**, **Ссылки для подключения**, **Настройки gRPC**.

Редактирование метода позволяет изменить **название метода**, **его тип** и **полный путь** до метода:



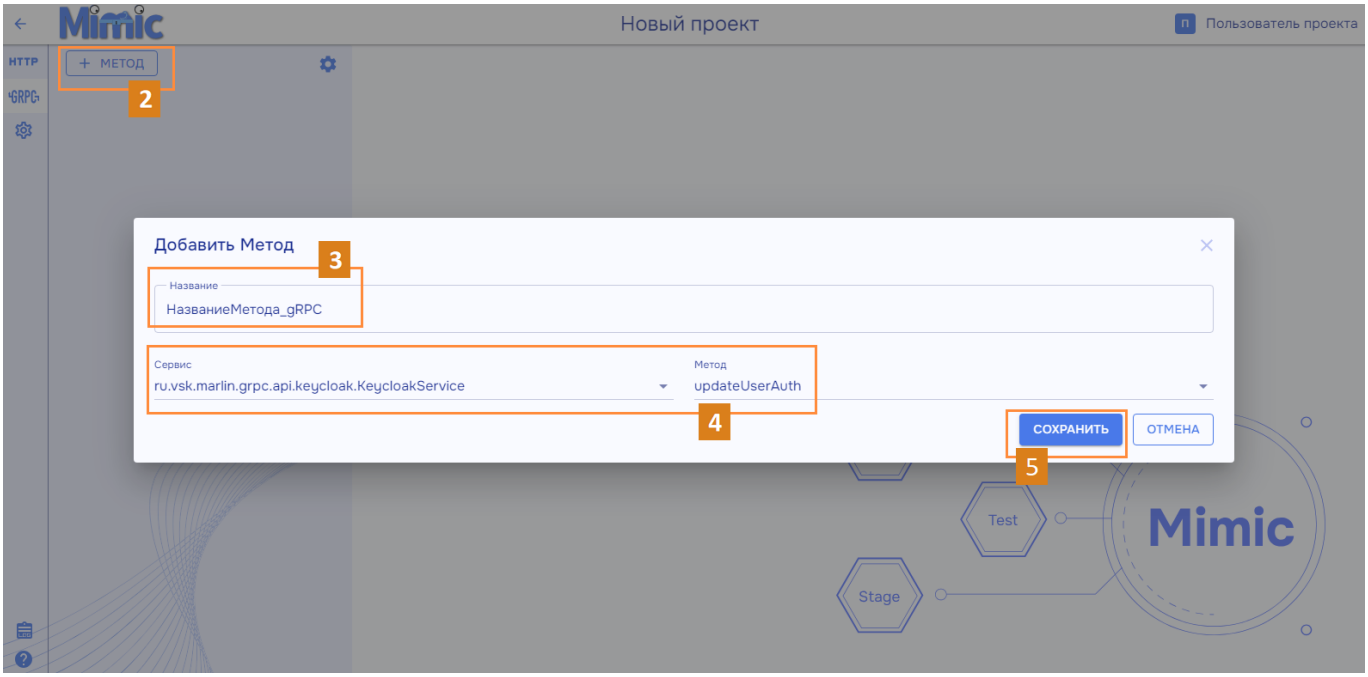
Добавление метода

Добавление метода может быть выполено ***двумя способами***: ***1)*** путем добавление нового метода с заполнением всех соответствующих полей; ***2)*** путем добавления нового метода на основе уже существующего - копирование метода.

> Если у проекта отсутствуют методы, то создание нового возможно только первым способом - с заполением всех полей.

В остальных случаях возможен второй способ, то есть создание метода на основе уже существующего (копирование) с его дальнейшим редактированием. При копировании метода создаются и правила.

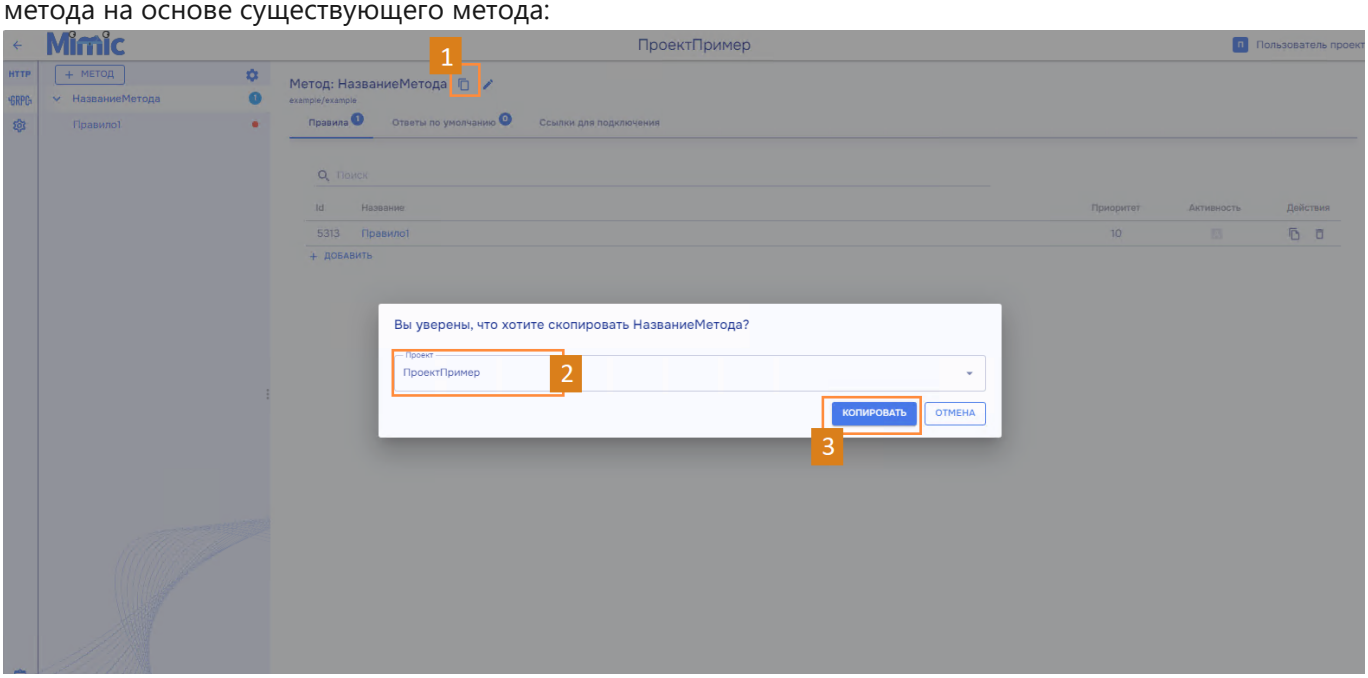
1. Добавление нового метода с заполнением полей Пример добавления метода:



Описание действий

1. Открыть проект, в который необходимо добавить маршрут;
2. Нажать кнопку добавления, располагающуюся в верхней части меню методов;
3. Указать **Название метода**;
4. Выбрать необходимый **Сервис** и **Метод** из выпадающего меню;
5. Нажать кнопку **Сохранить**.

2. Добавление метода на основе существующего (копирование маршрута) Пример добавления метода на основе существующего метода:



Описание действий

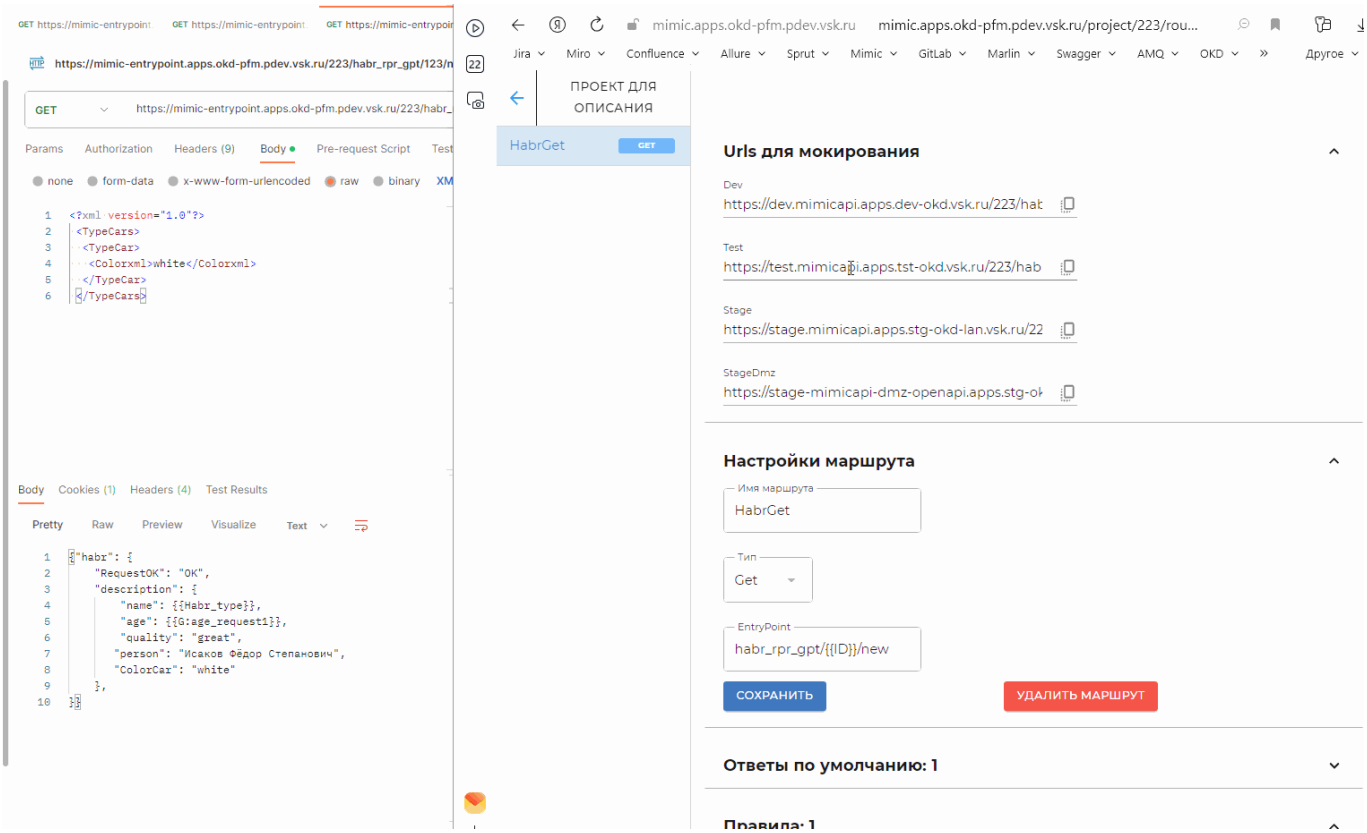
1. На странице необходимого для копирования метода нажимаем кнопку **Копировать** ;
2. Выбираем **Проект**, в который необходимо скопировать данный метод;
3. Нажимаем кнопку **Копировать**;

4. Открываем **созданный метод**;
5. Переходим в режим **редактирования** метода;
6. Вносим **необходимые изменения**.

На какой URL мока настроить сервис?

> **Urls** - это адреса подключения к созданному моку конкретного метода. Они автоматически генерируется с разбитием по контурам и позволяют скопировать значения для возможности указать в настройках тестируемого сервиса.

Пример использования Urls для мокирования:



Описание действий

1. Выбираем нужный контур
2. Копируем адрес подключения
3. Вставляем адрес в настройки тестируемого сервиса или Postman (так мы будем обращаться в мок)
4. Если есть переменные в адресе, например {{Id}}, заменяем их на конкретные значения.

> **Права пользователей**
>
> Создавать методы и управлять ими могут Владелец проекта и Пользователь, просматривать - Читатель.

Правила

Правила - это условия отбора необходимого ответа в зависимости от фильтров в рамках маршрута/метода. Помимо фильтров, правило содержит переменные и логику выдачи ответа в случае срабатывания.

Создание правила

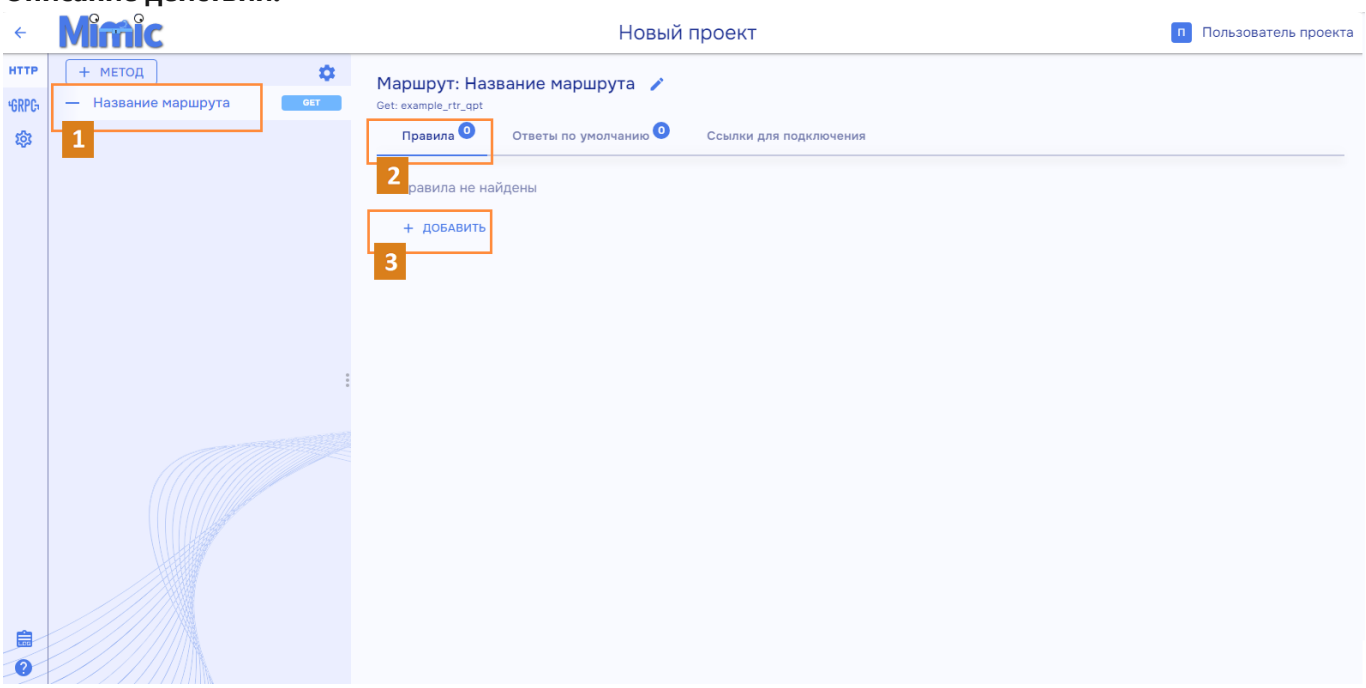
Создание правила может быть выполено ***двумя способами***: ***1)*** путем добавление нового правила с заполнением всех соответствующих полей; ***2)*** путем добавления нового правила на основе уже существующего - копирование правила.

Если у проекта отсутствуют правила, то создание нового правила возможно только первым способом - **с заполением всех полей**.

В остальных случаях возможен и второй способ, то есть **создание правила на основе уже существующего (копирование)** с его дальнейшим редактированием. При копировании правила создаются и вложенные объекты: фильтры, переменные, тело ответа.

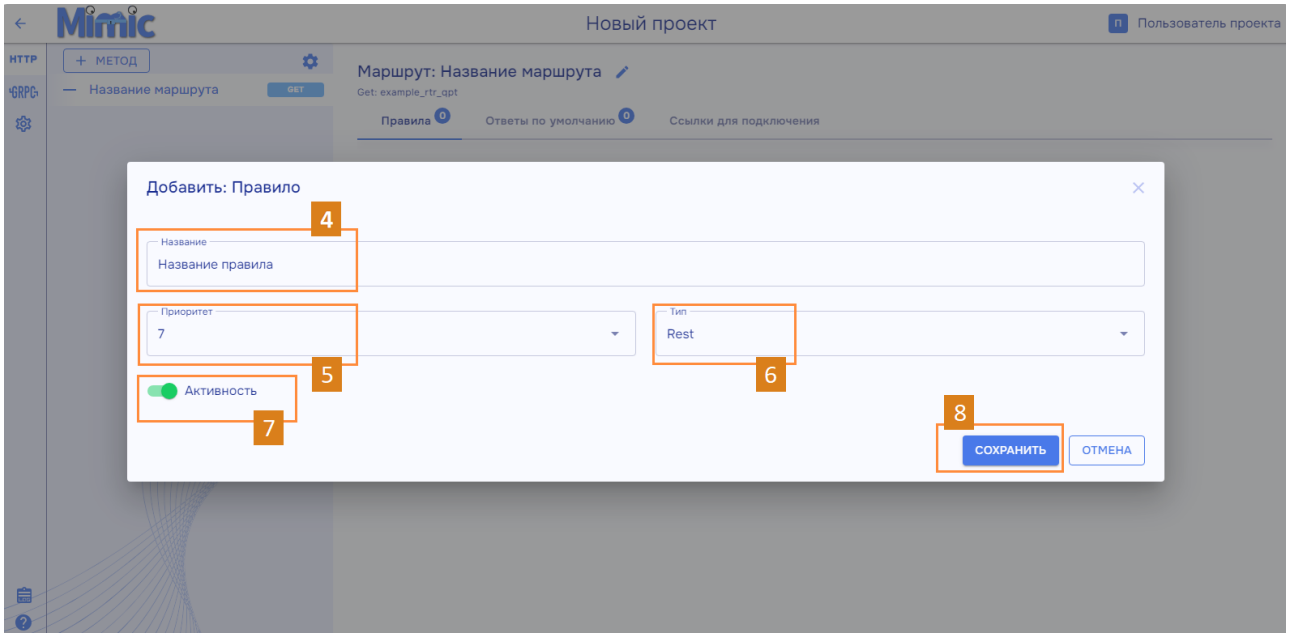
1. Добавление нового правила с заполнением полей

Описание действий:



1. Переходим в необходимый **маршрут**;
2. На странице маршрута раскрываем блок **Правила**;

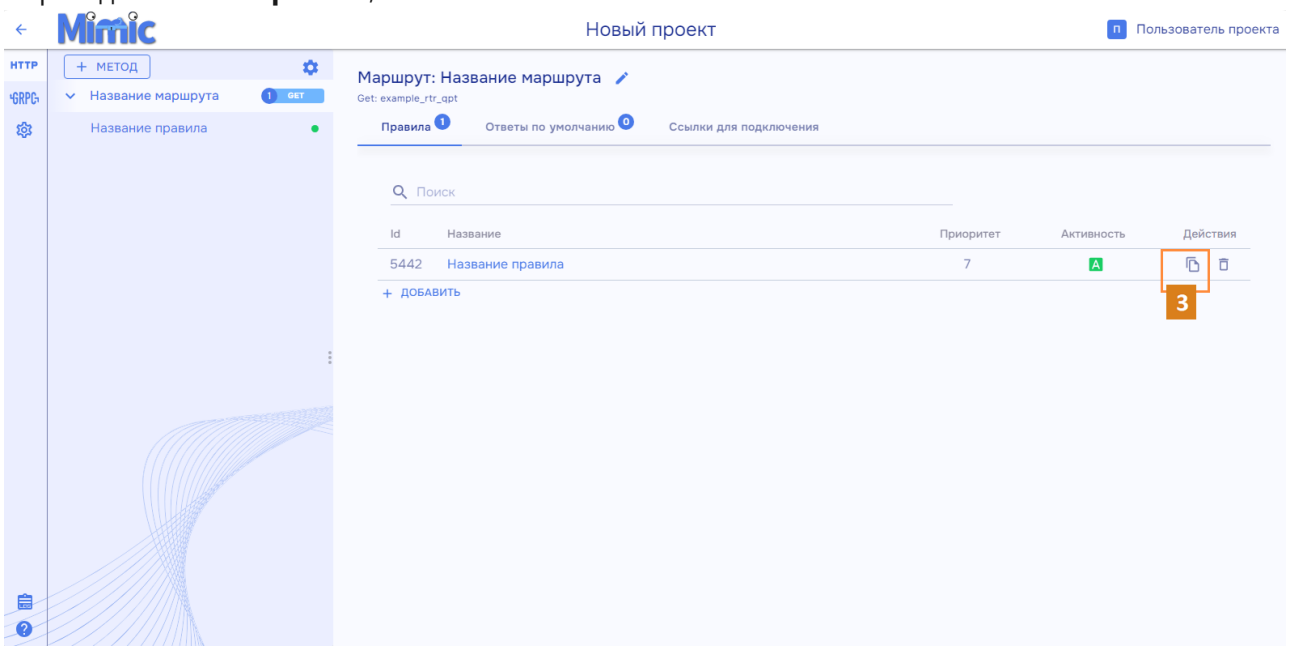
3. Нажимаем кнопку **Добавить**;



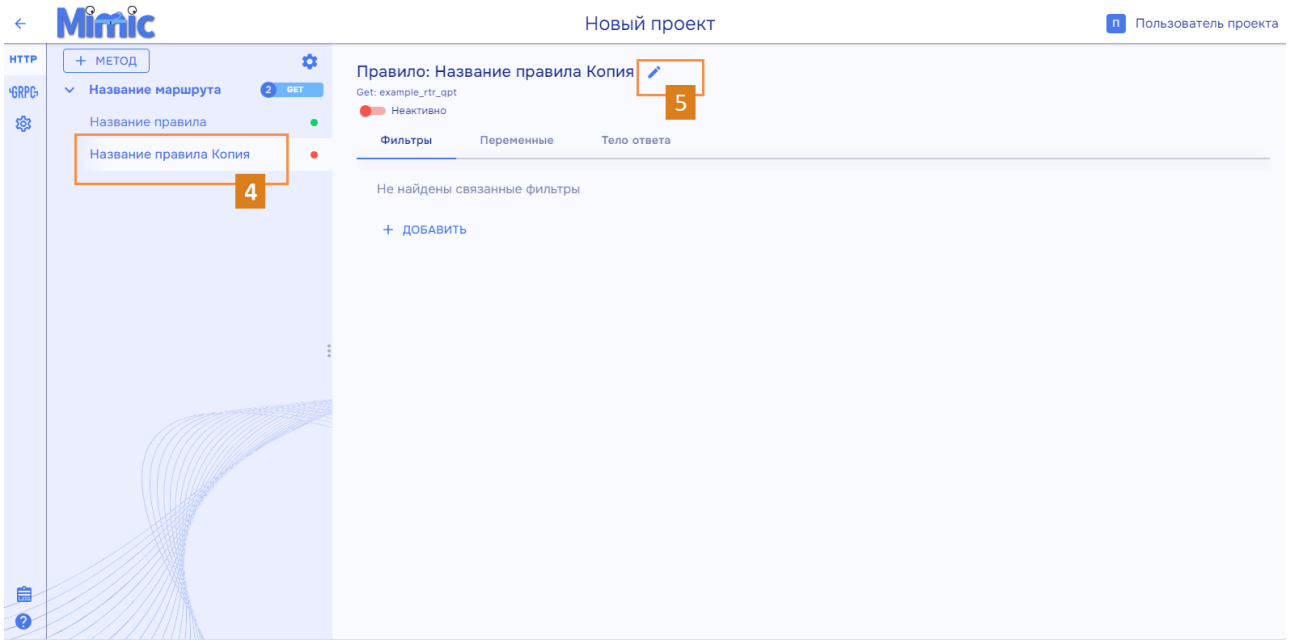
4. Указываем **Название правила**;
5. Выбираем **Приоритет** (от 1 до 10) Приоритет ставим в том случае, если входящий запрос может подойти для нескольких правил и нам нужно установить порядок обработки. Чем меньше значение приоритета, тем выше приоритет обработки и срабатывания;
6. Указываем **Тип запроса**: REST, SOAP. По нему будет происходить настройка фильтров для использования правила;
7. **Активность правила** оставляем **активным**, если использование правила необходимо в текущий момент, и меняем на **неактивное**, если временно использовать не нужно. Также менять активность правила возможно посредством API;
8. Нажимаем кнопку **Сохранить** для последующего заполнения правила.

2. Добавление правила на основе существующего (копирование правила) Пример добавления правила на основе существующего правила **Описание действий:**

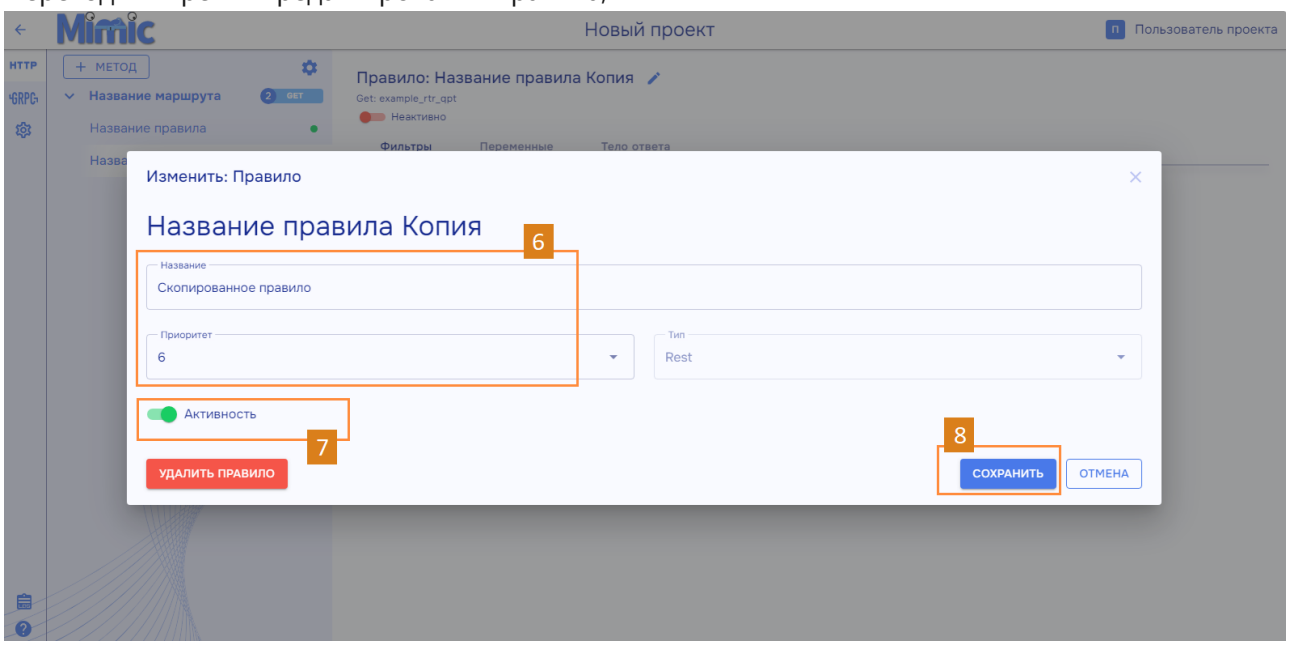
1. Переходим на страницу маршрута, в котором нужно создать правило;
2. Переходим в блок **Правила**;



3. Нажимаем кнопку **Копировать**. **Правило копируется неактивным.** ;



4. Открываем созданное правило;
5. Переходим в режим редактирования правила;



6. Вносим необходимые изменения в поля;
7. **Активность правила** переключаем на **активно**, если использование правило необходимо в данный момент, и оставляем **неактивным**, если временно использовать не нужно;
8. Нажимаем кнопку **Сохранить** для записи внесённых изменений.

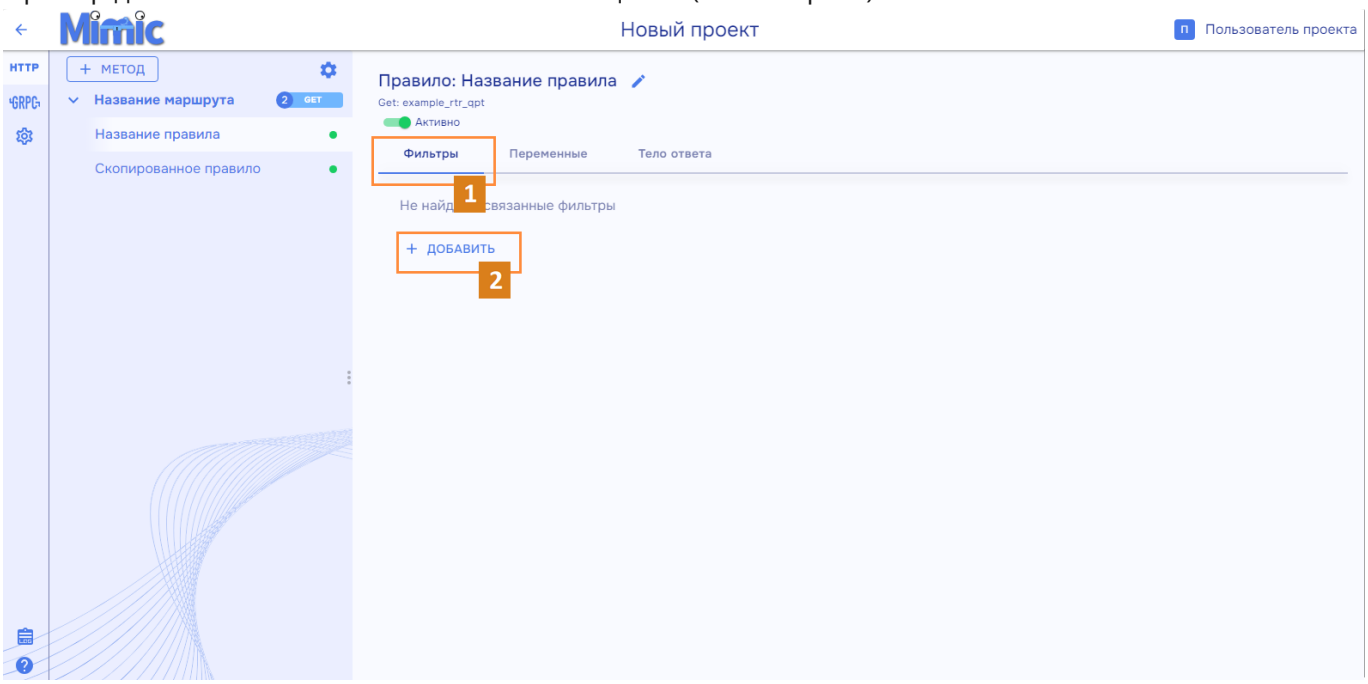
> **Права пользователей**
>
> Создавать правила и управлять ими могут Владелец проекта и Пользователь, просматривать - Читатель.

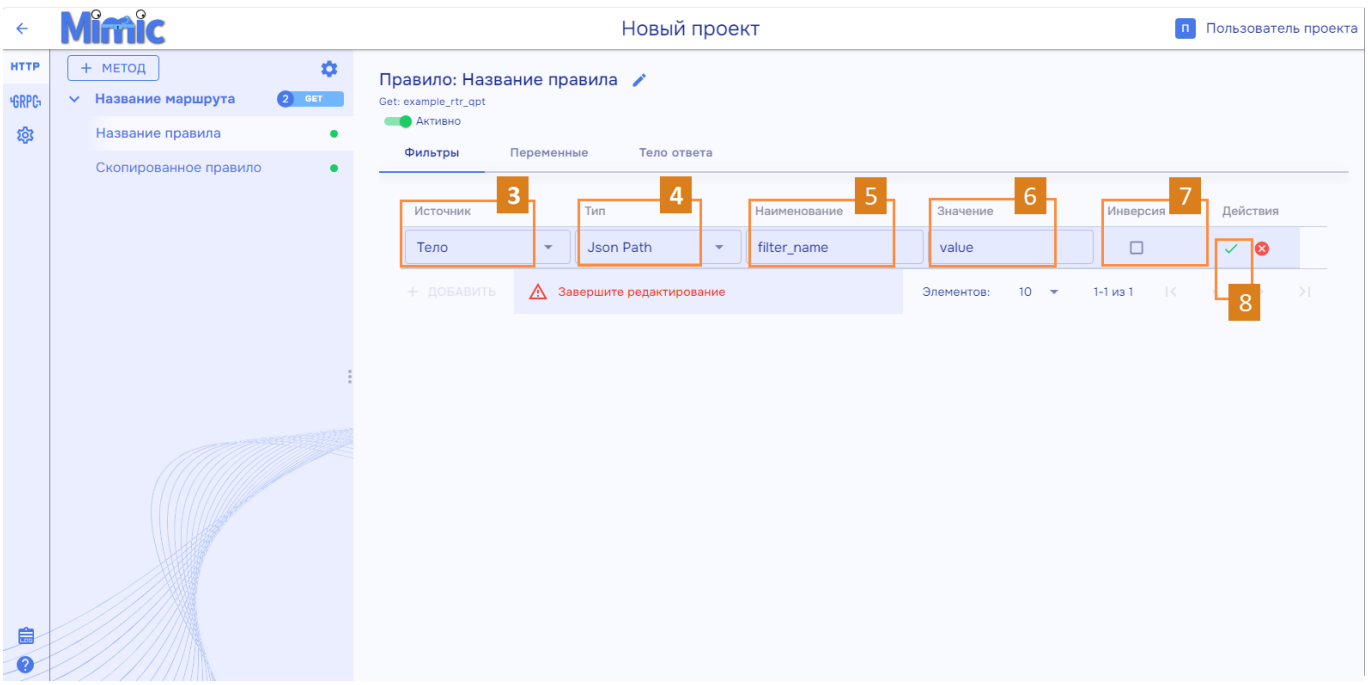
Фильтры

Фильтры - это сущность, которая используется для отбора и проверки значений во входящих запросах. Они позволяют определить условия, которые должны быть выполнены, чтобы сработало определенное правило.

Создание фильтров

Пример добавления значения в элемент сообщения (тело запроса):





Описание действий

1. На странице правила переходим во вкладку **Фильтры**;
2. Нажимаем на **кнопку добавления** нового фильтра;

3. Заполняем **Источник**;
4. Заполняем **Тип**;
5. Заполняем **Наименование**;
6. Заполняем **Значение**;
7. При необходимости проставляем признак **Инверсия**, в результате применения которого будут отбираться **все значения, кроме тех, которые были введенны пользователем в предыдущем пункте**;
8. Нажимаем ✓ для сохранения.

Виды фильтров

1) Фильтр по заголовкам входящего запроса

Для создания **фильтра по заголовкам** входящего запроса необходимо:

- В блоке **Фильтры** добавить новый фильтр посредством нажатия кнопки **Добавить**;
- Выбрать источник для получения данных: **Заголовок**;
- Указать **тип** отбора:
  - **Ключ** - фиксированное значение параметра по ключу, где имя принимаемого параметра и ожидаемое значения указывается в соответствующих полях **Наименование** и **Значение**;
  - **Регулярное выражение** - это тип, используемый для отбора значения через Regex-выражения (Примеры использования Regex). В этом случае в поле **Наименование** указывается имя переменной, а в поле **Значение** указывается выражение для отбора строки или ожидаемое значение. Для проверки выражения можно использовать Проверка выражения отбора Regex

Фильтр срабатывает при совпадении значения в заданном заголовке. Пример:

Источник	Тип	Наименование	Значение	Инверсия	Действия
Заголовок	Регулярное выражение	parameter_name	[a-b]	☐	🗑
Заголовок	Ключ - Значение	parameter_name	value	☐	🗑

2) Фильтр по параметрам входящего запроса

> **Параметры в url-запросе** - это то, что передаётся после "?".

Например:

https://host.ru/MethodName?Parameter1=value1&Parameter2=value2

Если необходимо, чтобы правило срабатывало **в зависимости от переданного параметра**, то:

- Необходимо в блоке **Фильтры** добавить новый фильтр посредством нажатия кнопки **Добавить**;
- Выбрать источник для получения данных: **Параметр(url)**;
- Указать **тип** отбора:
  - **Ключ** - фиксированное значение параметра по ключу, где имя принимаемого параметра и ожидаемое значения указывается в соответствующих полях **Наименование** и **Значение**;
  - **Регулярное выражение** - это тип, используемый для отбора значения через Regex-выражения (Примеры использования Regex). В этом случае в поле **Наименование**

указывается имя переменной, а в поле **Значение** указывается выражение для отбора строки или ожидаемое значение. Поиск производится сразу по всей строке параметров переданных **после "?"**. Для проверки выражения можно использовать Проверка выражения отбора Regex

Пример:

Источник	Тип	Наименование	Значение	Инверсия	Действия
Параметр (Url)	Регулярное выражение	parameter_name	[1-5, a-b]	☐	🗑
Параметр (Url)	Ключ - Значение	parameter_name	value	☐	🗑

3) Фильтр по переменной из адреса обращения

В некоторых запросах **значения могут передаваться в самом адресе**. Например:
https://host.ru/project/158/route/56/rule То есть в этом запросе в пути фигурирует идентификатор проекта **IdProject**, равное **158**, и идентификатор правила **IdRoute**, равное **56**.

Если необходимо **создать фильтр для такого маршрута**, то:

- В **EntryPoint** на странице маршрута будет задана строка **project/{{IdProject}}/route/{{IdRoute}}/rule**;
- Необходимо в блоке **Фильтры** добавить новый фильтр посредством нажатия кнопки **Добавить**;
- Выбрать источник для получения данных: **Адрес (url)**;
- Указать **тип** отбора:
    - **Ключ** - фиксированное значение, где **Наименование** - это имя переменной из **EntryPoint** маршрута, для примера выше это может быть **E:IdProject** (E: - означает что это не локальная переменная, а переменная маршрута), **Значение** - это ожидаемая строка, для примера выше это **158**.
    - **Регулярное выражение** - это тип, используемый для отбора значения через Regex-выражения (Примеры использования Regex). В этом случае в поле **Наименование** указывается имя переменной, а в поле Значение указывается выражение для отбора строки или ожидаемое значение Для проверки выражения можно использовать Проверка выражения отбора Regex.

Пример:

Источник	Тип	Наименование	Значение	Инверсия	Действия
Адрес (Url)	Регулярное выражение	E:IdProject	[1-35]	☐	🗑
Адрес (Url)	Ключ - Значение	E:IdProject	35	☐	🗑

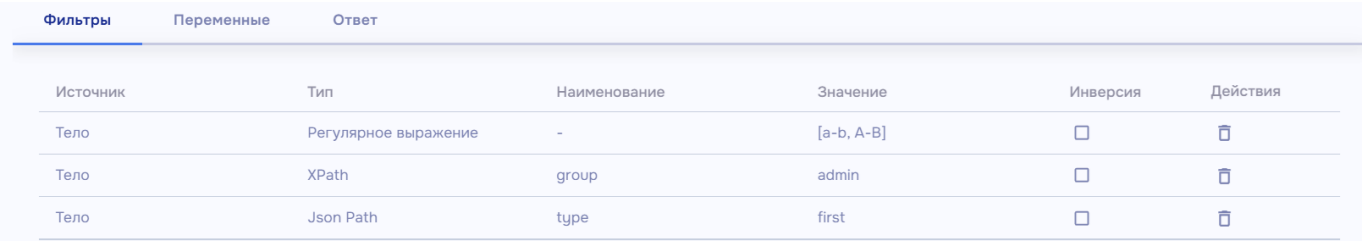
4) Фильтр по телу входящего запроса

Если нужно, чтобы правило срабатывало по значениям указанных в теле входящего запроса, то:

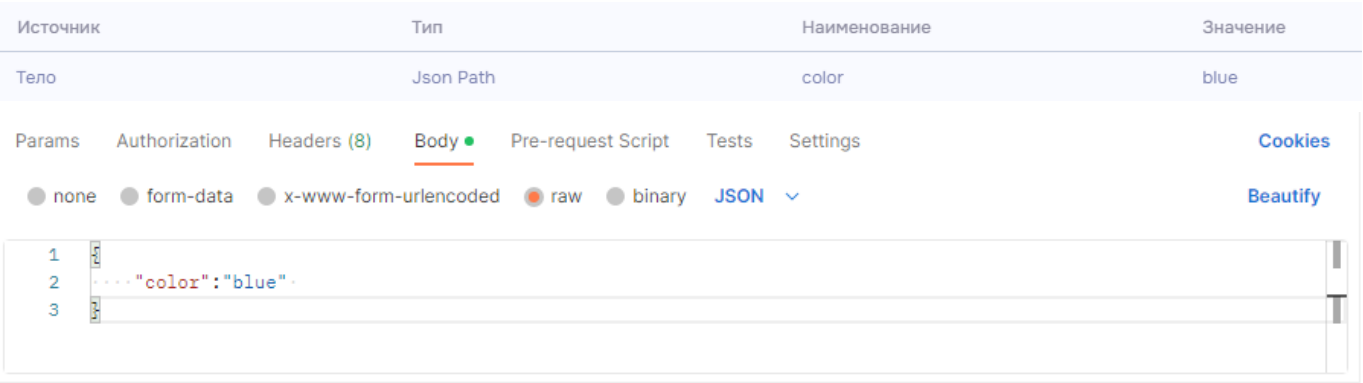
- Необходимо в блоке **Фильтры** нажать на кнопку **"Добавить"**;
- Выбрать источник для получения данных: **Тело**;
- Выбрать необходимый **Тип** выражения отбора: **JsonPath, XPath** или **Регулярное выражение**:
    - **JsonPath** - язык запросов для выбора и извлечения значений из JSON-контента, где **Наименование** - это запрос на языке JsonPath, следующий **после "$."** Проверка запроса JsonPath.

- **XPath** - язык запросов для выбора и извлечения значений из XML-контента, где **Наименование** - это запрос на языке XPath, следующий после "//". Проверка запроса XPath. **Значение** - это ожидаемое значение найденного результата.
- **Регулярное выражение** - это тип, используемый для отбора значения через Regex-выражения (Примеры использования Regex), в этом случае поле **Наименование** заполняется произвольно (так как отбор происходит по всему телу), а в поле **Значение** прописывается регулярное выражение для отбора строки. Для проверки выражения можно использовать Проверка выражения отбора Regex.

Пример:



Пример с Postman:



Признак инверсии

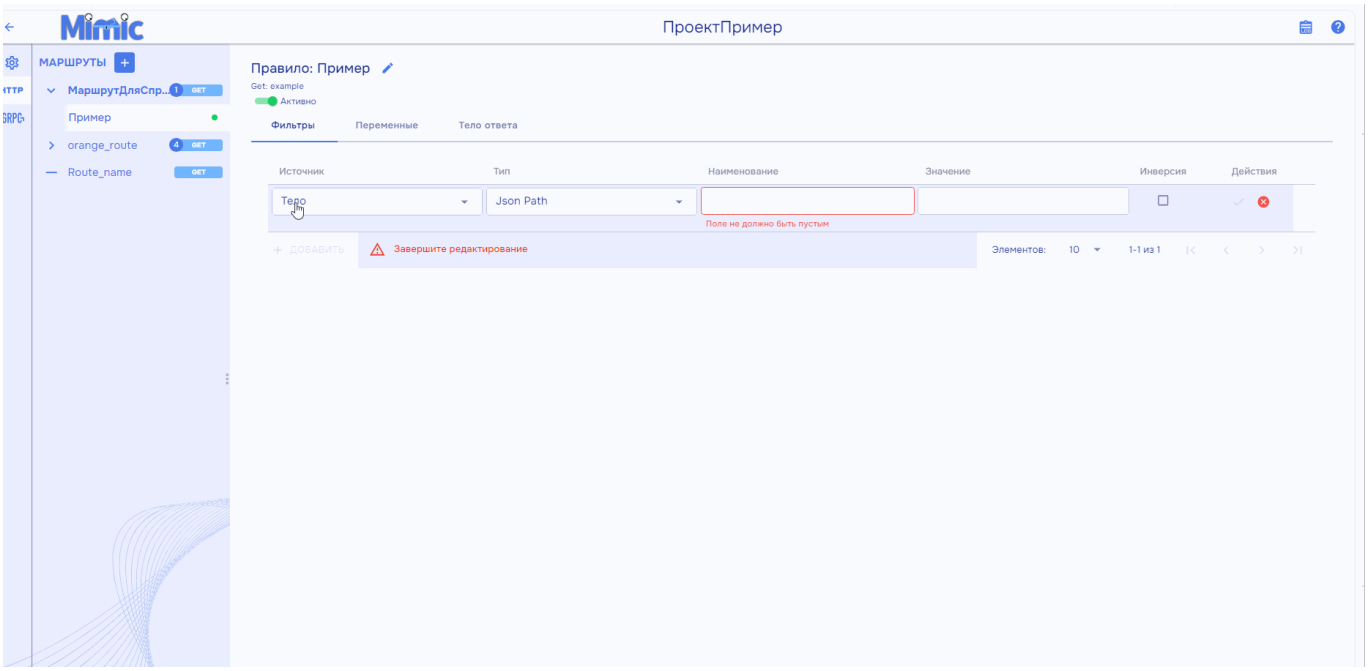
Если требуется, чтобы фильтр срабатывал всегда, **кроме указанного условия**, то необходимо указать **признак инверсии** у соответствующего фильтра.

Примечание

> Если после наименования фильтра есть звездочка (Тело запроса*), то это значит, что один из объектов в процессе редактирования.

Пример добавления фильтра и корректное получение ответа на примере фильтра в теле запроса типа JsonPath

Пример получения ответа, используя в фильтре JsonPath:



Описание действий

1. Добавляем в правило фильтр по телу запроса, заполняя: источник, тип, локатор, значение;
2. В Postman указываем тело запроса, с тегом и значением идентичному в фильтре **"color":"blue"**, чтобы наш фильтр отобрал нужное нам правило;
3. Проверяем, что **фильтр сработал** -> И в **Body** ответа Postman **получаем тело ответа**, предварительно настроенное на вкладке **Тело ответа**;
4. Ставим **инверсию** для правила -> Проверяем в Postman, что **теперь правило не срабатывает**.

> **Права пользователей**
>
> Создавать фильтры и управлять ими могут Владелец проекта и Пользователь, просмотривать - Читатель.

Переменные

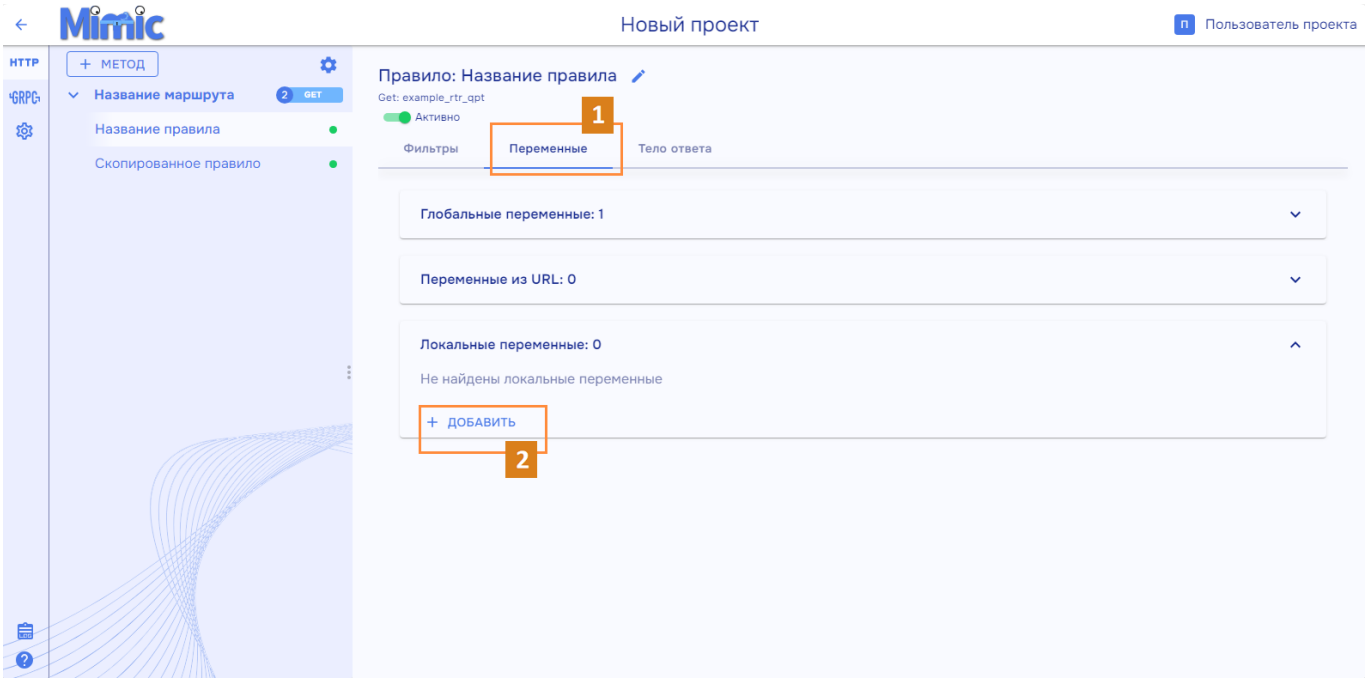
Переменные - сущность, которая нужна, чтобы в ответе мока выдавать гибкий набор значений, который зависит от потребностей пользователя. В ответ может прийти сгенерированное ФИО, GUID, дата, инкрементированный Id и многое другое.

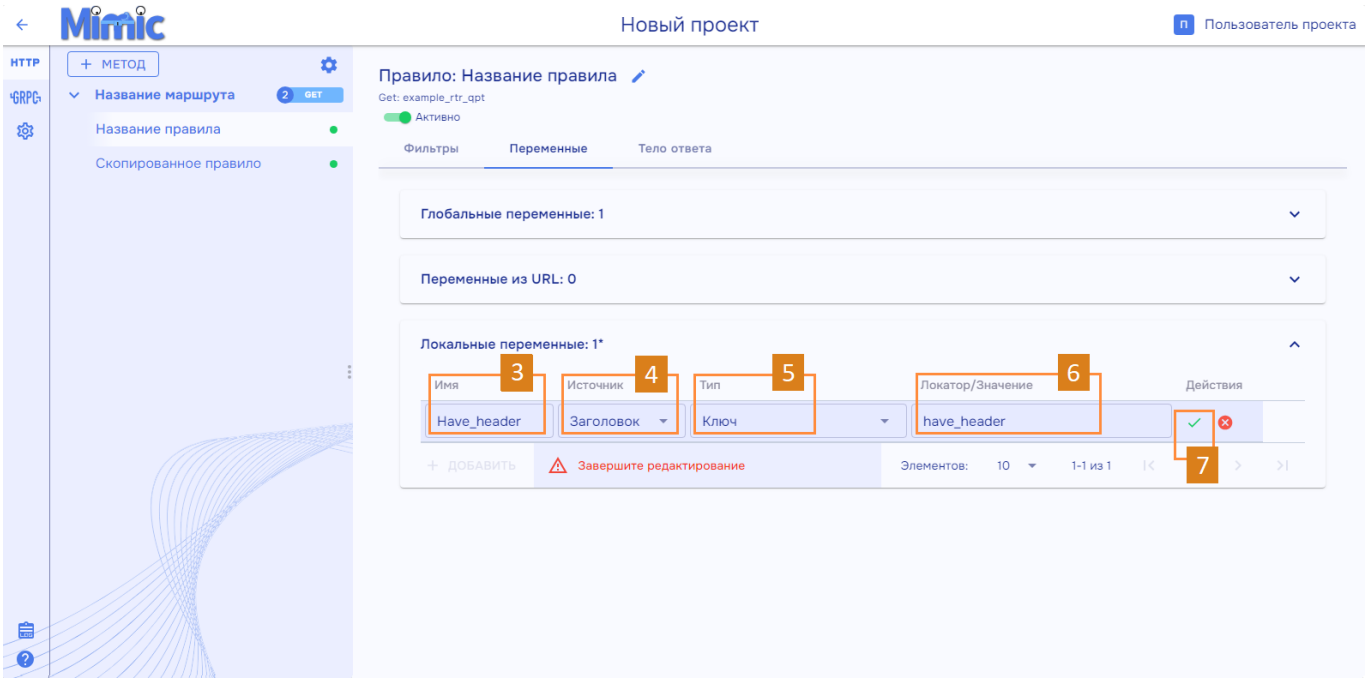
Локальные переменные

Локальные переменные относятся к конкретному правилу. Следовательно, их имена и значения используются только в рамках одного правила. Переменные позволяют сформировать ответный запрос, используя отобранные или сгенерированные в них значения.

Создание локальных переменных

Пример заполнения локальной переменной:





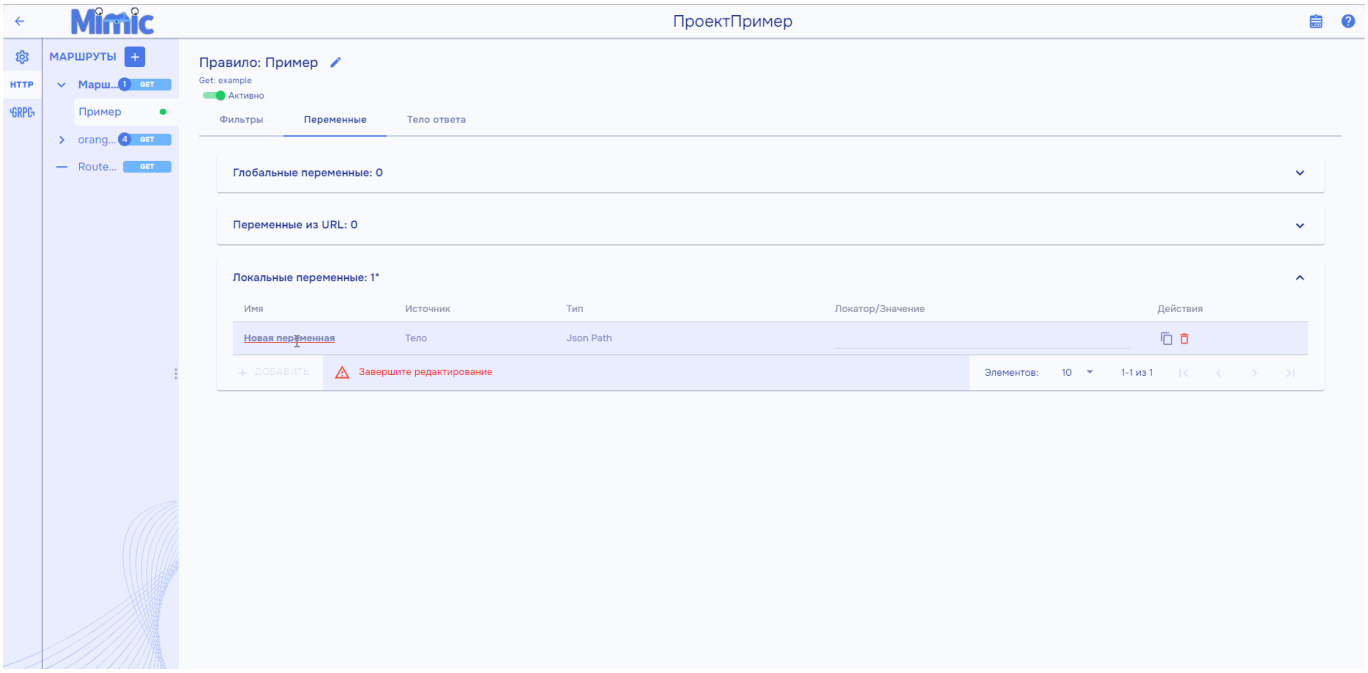
Описание действий

1. На странице правила переходим во вкладку **Переменные**;
2. Нажимаем на **кнопку добавления** новой локальной переменной;
3. Устанавливаем имя локальной переменной в поле **Имя**, оно должно быть уникальным в рамках правила;
4. В поле **Источник** выбираем способ получения значения. Это может быть часть входящего запроса (Headers, Body или параметры). На изображении выше мы выбираем источник заголовки входящего запроса *Header*;
5. Выбираем **Тип** значения, которое будем указывать в поле **Локатор/Значение**. На изображении выше выбираем тип **Key**, т.е. будем отбирать Заголовок по его переданному имени;
6. Указываем **Локатор/Значение**. Он необходим для того чтобы по ключу выбрать необходимую пару из определенного места сообщения (место сообщения указано в поле "Источник"). На изображении выше мы указываем **Have_header**, т.е. при срабатывании правила, в созданной нами переменной будет храниться значение заголовка *Have_header* из входящего запроса, и мы сможем использовать его в дальнейшем.
7. Нажимаем ✓ для сохранения.

Использование локальной переменной

Локальную переменную мы можем использовать в теле или в заголовках ответа, указав ее имя в двойных фигурных скобках: **"{{Имя переменной}}"**.

Пример использования локальной переменной в теле ответа:



Описание действий

1. Выбираем правило, которое срабатывает на наш запрос из Postman;
2. Добавляем локальную переменную с названием **"Habr_Type"**, источником **Заголовок**, типом **Ключ**, локатором **HabrType**, копируем переменную для вставки в ответ;
3. В Postman на вкладке **Headers** добавляем заголовок **HabrType** со значением **Done**;
4. В Mimic на странице **Тело ответа** вставляем желаемый ответ, включающий в себя тег **"Result":** и переменную **"{{HabrType}}"**;
5. Выполняем запрос в Postman и получаем ответ, где тег **"Result"** имеет значение **"Done"** из заголовка переданного нами запроса.

Описание типов переменных по источникам

Источник "Header" или Источник "Parameter"

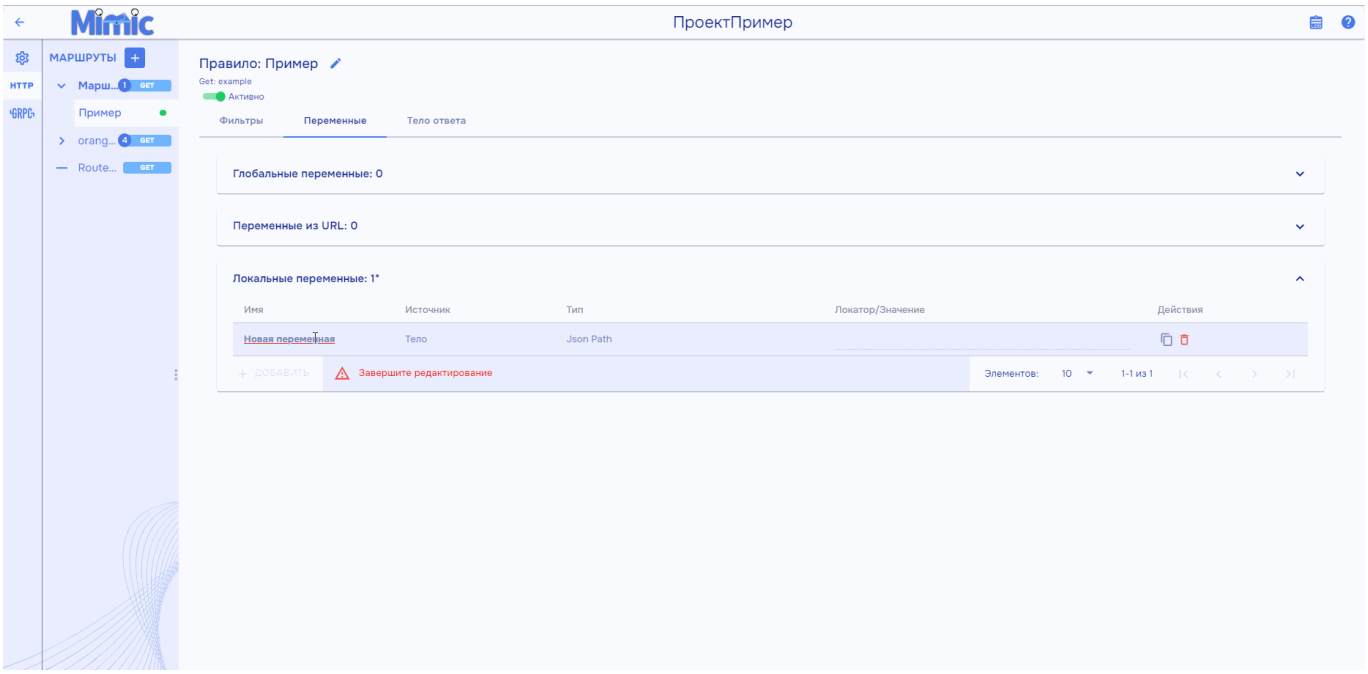
- **Key** - отбор по ключу или имени заголовка/параметра из переданного запроса. Имя для поиска заголовка/параметра указывается в поле **"Локатор/Значение"**. По нему, при срабатывании правила, переменная сможет отобрать значение и сохранить его до передачи в ответ.

- **Regex** - регулярное выражение. Используется для отбора строки из выданного источника и сохранении его значения до выдачи в ответе. Ниже приведены примеры использования.

Источник "Тело"

- **JsonPath** - язык запросов для JSON:
  - Ссылка на описание синтаксиса
  - Ссылка на проверку отбора значения используя JsonPath
- **XPath** - язык запросов для XML:
  - Ссылка на описание синтаксиса
  - Ссылка на проверку отбора значения используя XPath

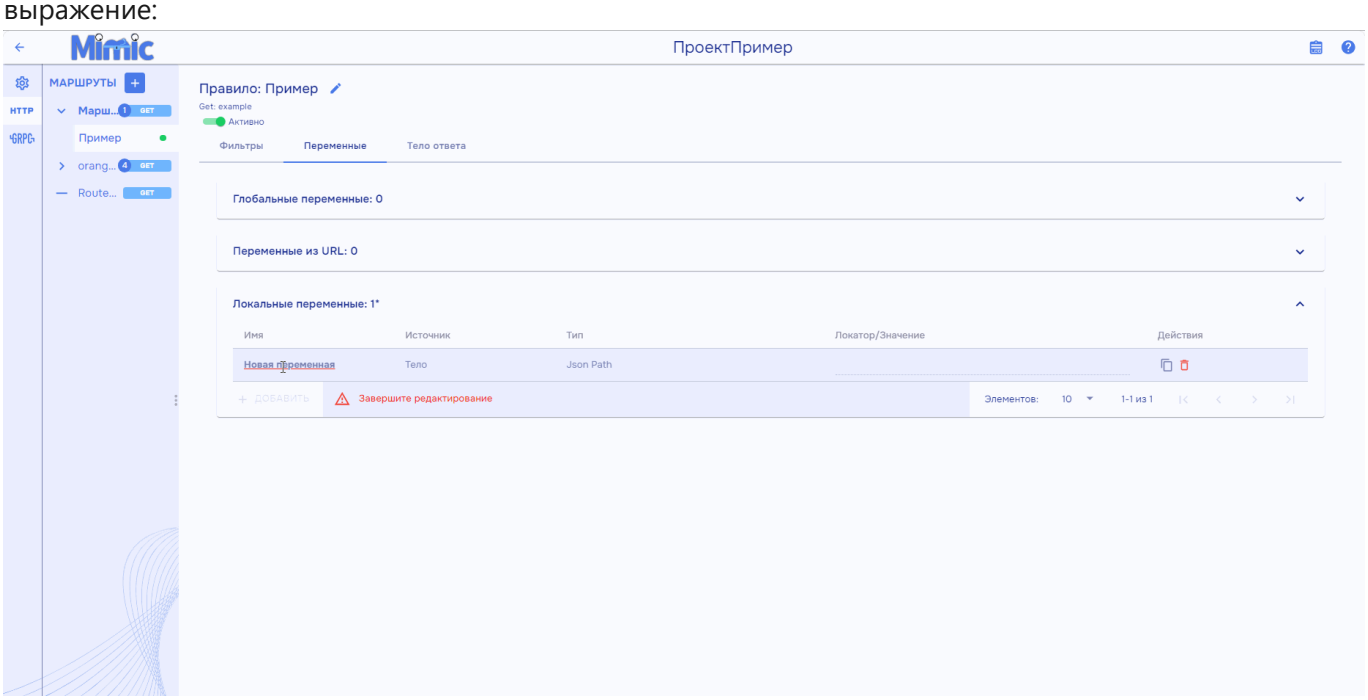
Пример отображения ответа с использованием синтаксиса XPath:



- **Regex** - регулярное выражение. Используется для отбора строки из выданного источника и сохранении его значения до выдачи в ответе.
    - Примеры использования Regex
    - Ссылка на проверку выражения отбора используя Regex

Пример использования регулярного выражения

Пример использования регулярного выражения с использованием Тела с типом Регулярное выражение:



В данном примере mimic с помощью регулярного выражения "[a-zA-Z]+" отбирает только буквы из переданной строки с буквами и цифрами и сохраняет полученное значение в переменную result_type. Затем использует значение переменной для выдачи в ответе.

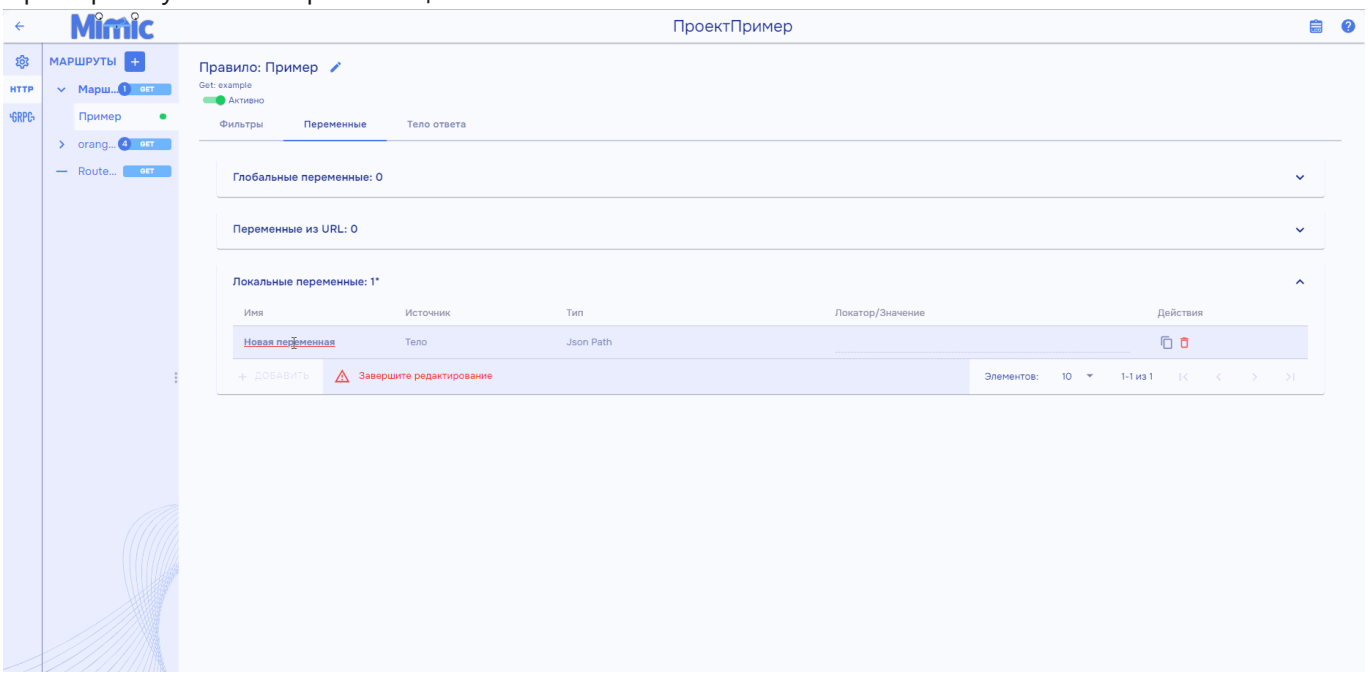
Источник "Своё значение"

- **Статичное** - данный тип переменной позволяет передавать статичное значение при каждом запросе без последующих изменений.

- **Инкрементируемое Число** - данный тип переменной позволяет передавать числовое значение, которое будет инкрементироваться (n+1) после повторного запроса. Такой тип данных можно использовать, например, для явного отслеживания количества запросов или формирования уникальных Id. Начальное значение n указывается в поле **"Локатор/Значение"**, и будет выдано в первом ответе.

Пример использования инкрементируемого значения

Пример получения инкрементации в ответе:



Описание действий

1. Добавляем локальную переменную в выбранное правило;
2. Указываем название переменной, например **"result_type"**;
3. Указываем источник **Своё значение**, тип - **Инкрементируемое Число**;
4. В поле **"Локатор/значение"** ставим в качестве примера единицу, копируем переменную для вставки в ответ;
5. На вкладке **"Тело ответа"** непосредственно в самом теле указываем переменную. Пример из изображения: **"Result": "{{result_type}}"**;
6. В **Postman** делаем вызов метода и видим, что первый раз нам выдало единицу, а с каждым последующим вызовом значение Result увеличивается на 1. Просмотреть следующее выдаваемое значение можно, если открыть данную переменную в Mimic.

- **Создать GUID** - тип переменной, который позволяет при каждом срабатывании правила генерировать новый уникальный GUID.

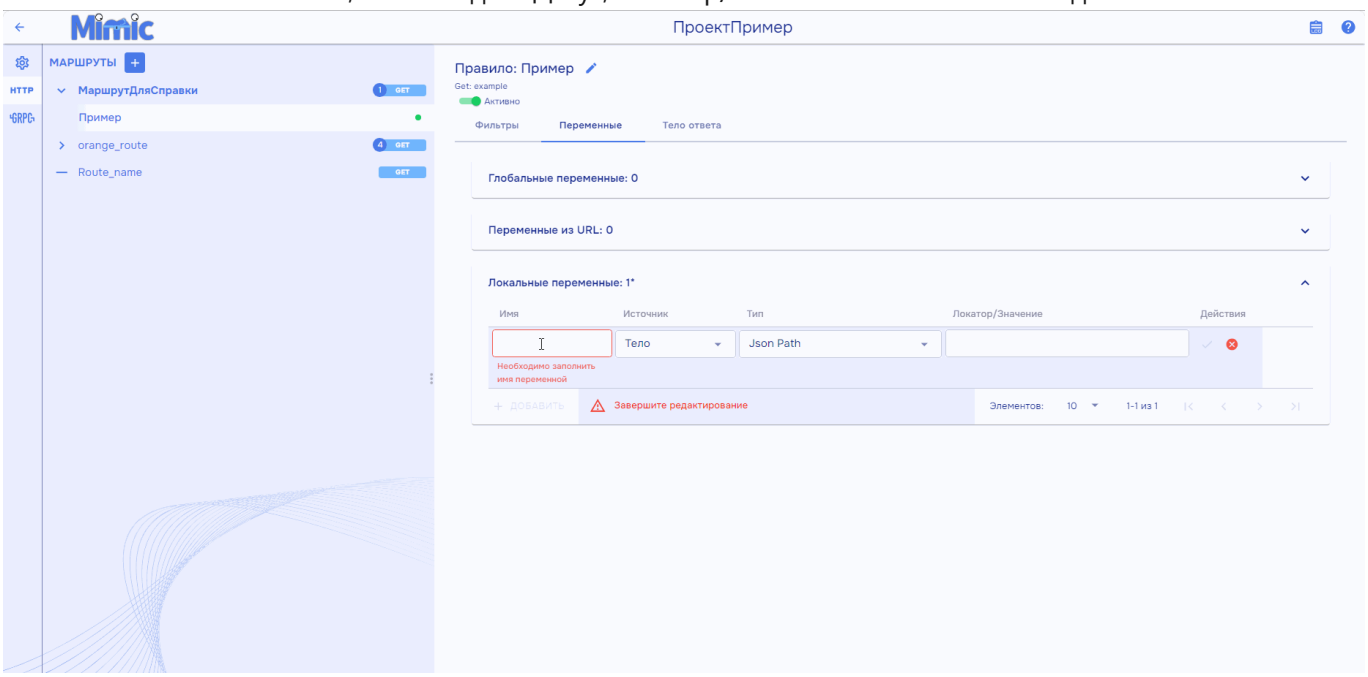
- **Создать текст (кириллица)** - тип переменной, который позволяет при каждом срабатывании правила генерировать строку заданной длинны, состоящую из случайных символов в кириллице. Количество генерируемых символов указывается в поле **"Локатор/значение"**.

- **Создать текст (латиница)** - тип переменной, который позволяет при каждом срабатывании правила генерировать строку заданной длинны, состоящую из случайных символов в латинице. Количество генерируемых символов указывается в поле **"Локатор/значение"**.

- **Создать Дату** - тип переменной, который позволяет сгенерировать значение даты и времени. Существуют варианты использования фиксированной даты или относительной переключением свитча. **Относительная дата** - это дата и время на момент запроса. Дополнительно можно выбрать различные форматы времени, указать смещение +/- в днях, месяцах, годах, часах, минутах, секундах. Т.е. если мы настроим относительную дату со смещением на 3 дня и 2 часа, то в переменную сохранится значение **даты и времени входящего запроса по Гринвичу** +3 дня и + 2 часа.

Пример использования переменной с генерацией даты:

Источник "Своё значение", тип "Создать Дату", локатор/значение "Относительная дата"

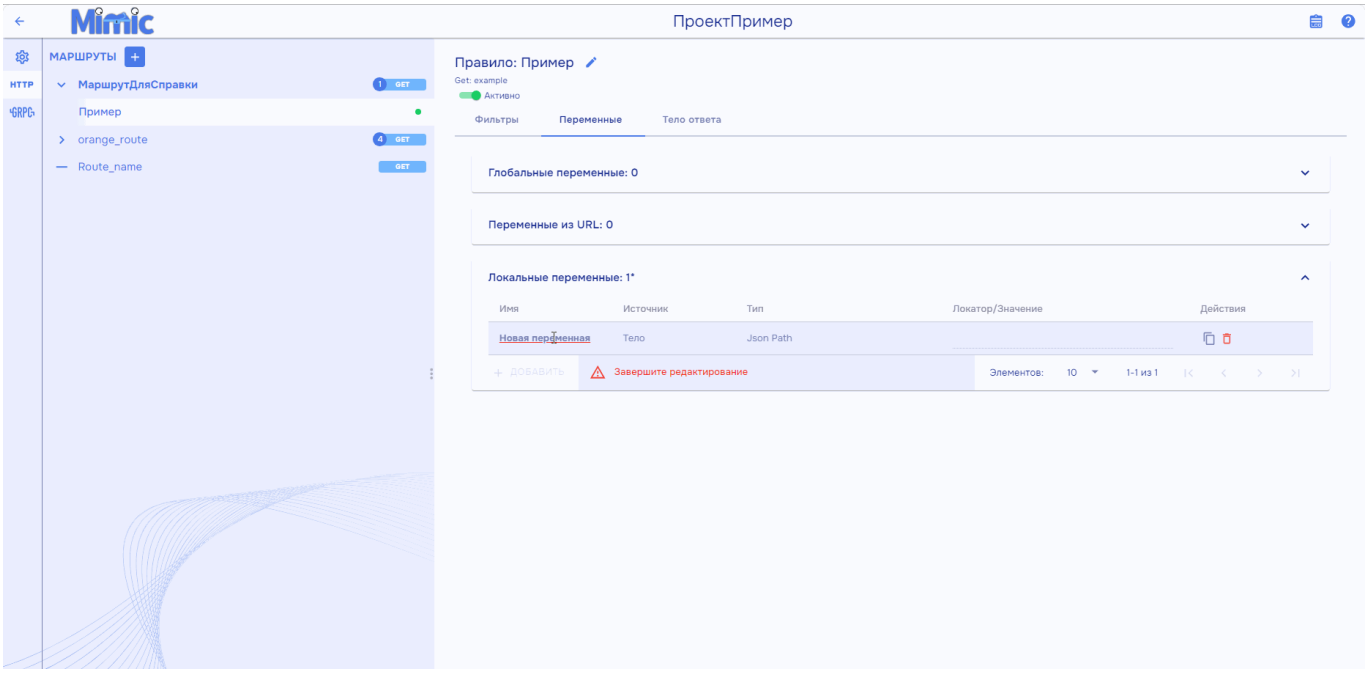


- **Создать число** - тип переменной, который позволяет в ответе генерировать определенное случайное числовое значение. Количество разрядов числового значения указывается в поле **"Локатор/значение"**

- **Создать ФИО** - тип переменной, который позволяет в ответе генерировать ФИО или отдельные его элементы: **имя, фамилия, отчество.** Также можно выставить отображение инициалов в ФИО вместо имени и отчества. При каждом запросе будем получать новое ФИО.

Пример генерации ФИО

Пример генерации ФИО по типу переменной PersonGenerated:



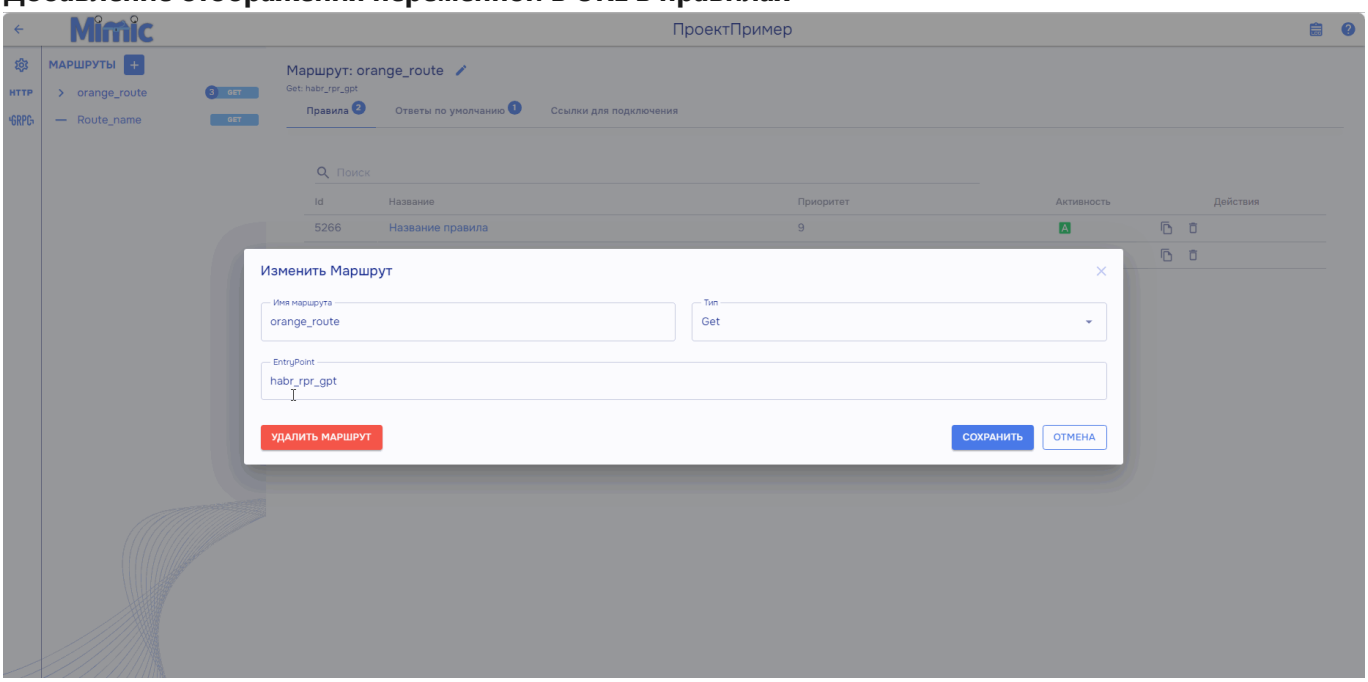
Переменные из URL

В некоторых запросах значения могут передаваться в самом адресе. Например, https://host.ru/project/158/route/56/rule. То есть в этом запросе в пути фигурирует идентификатор проекта **IdProject**, равное **158** и идентификатор правила **IdRoute**, равное **56**. В этом случае на форме маршрута в поле **EntryPoint** необходимо указать адрес, используя переменные. Для запроса выше это будет **project/{{IdProject}}/route/{{IdRoute}}/rule**.

В каждом правиле маршрута на вкладке **Переменные** в блоке **Переменные из Url** мы получим созданные переменные с приставкой E: - **E:IdProject** и **E:IdRoute**, которые будут хранить переданные в запросе значения.

Пример использования переменной в URL

Добавление отображения переменной в URL в правилах



Описание действий

1. Переходим в **Настройки маршрута**;
2. Указываем в **EntryPoint** параметр **ID**. В примере выше: **habr_rpr_gpt/{{ID}}/new**;
3. Сохраняем настройки маршрута;
4. Переходим в редактирование правила;
5. Переходим в раздел **Переменные->Переменные из URL**;
6. Проверяем, что появилась новая строка с переменной из URL.

Глобальные переменные

В правиле на вкладке **Переменные** в блоке **Глобальные переменные** отображаются переменные проекта без возможности редактирования с префиксом **G:**, созданные на форме **Настройки проекта** по правилам обычных переменных. Узнать подробнее можно в разделе Настройка проекта.

> **Права пользователей**
>
> Создавать переменные и управлять ими могут Владелец проекта и Пользователь, просмотривать - Читатель.

Формирование ответного сообщения

> **Ответное сообщение** - это настраиваемая структура данных, состоящая из заголовков, кода ответа и тела ответа.

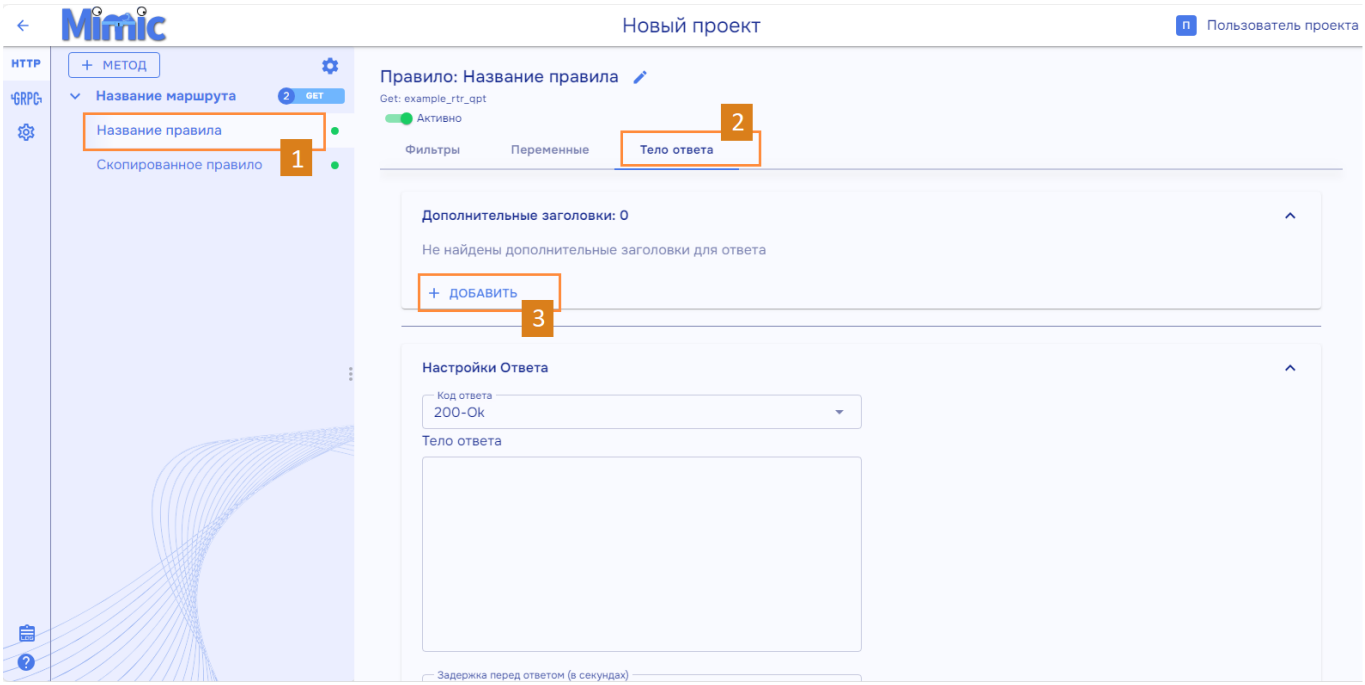
Находится в конкретном правиле, формирует и передаёт ответ в случае срабатывания всех условий правила. Подробней про условия в разделе Фильтры.

Добавление тела ответа

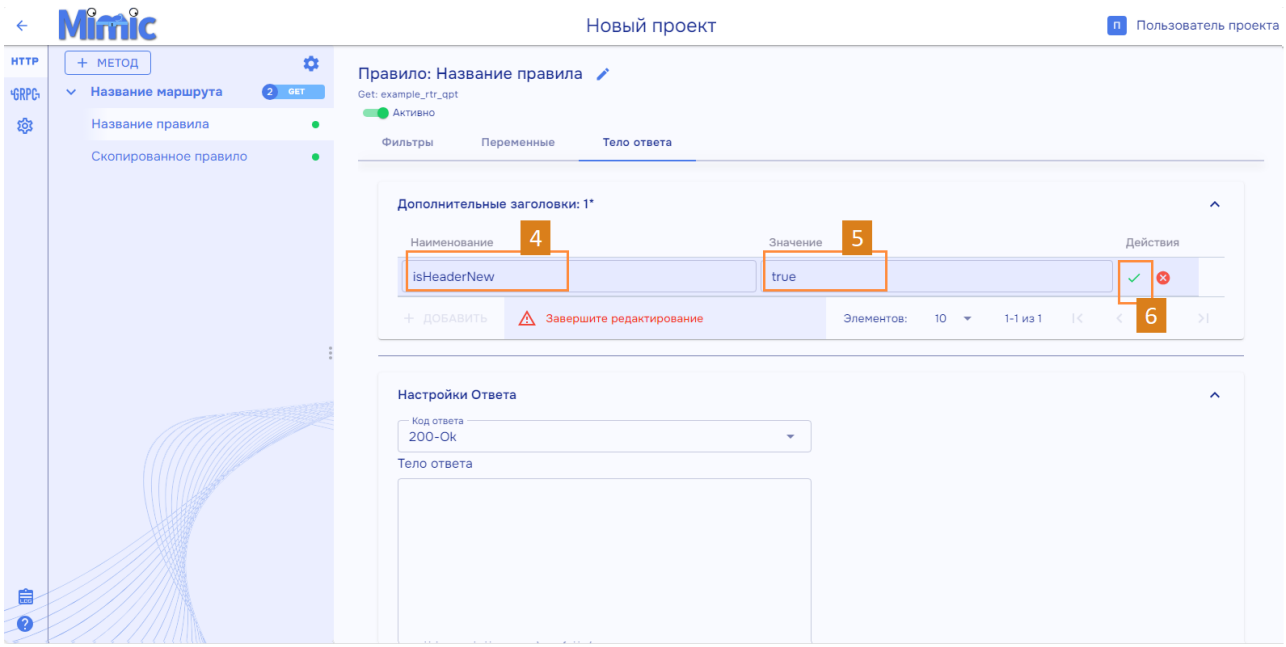
Добавление дополнительных заголовков

В блоке **"Дополнительные заголовки"** можно добавить заголовки для ответного сообщения. Для этого нужно заполнить **"Название"** заголовка и передаваемое **"Значение"**

Пример добавления дополнительного заголовка в теле ответа **Описание действий:**



1. Переходим на страницу **Правила**;
2. Переходим во вкладку **Тело ответа**;
3. Нажимаем на кнопку добавления нового дополнительного заголовка в разделе **Дополнительные заголовки**;



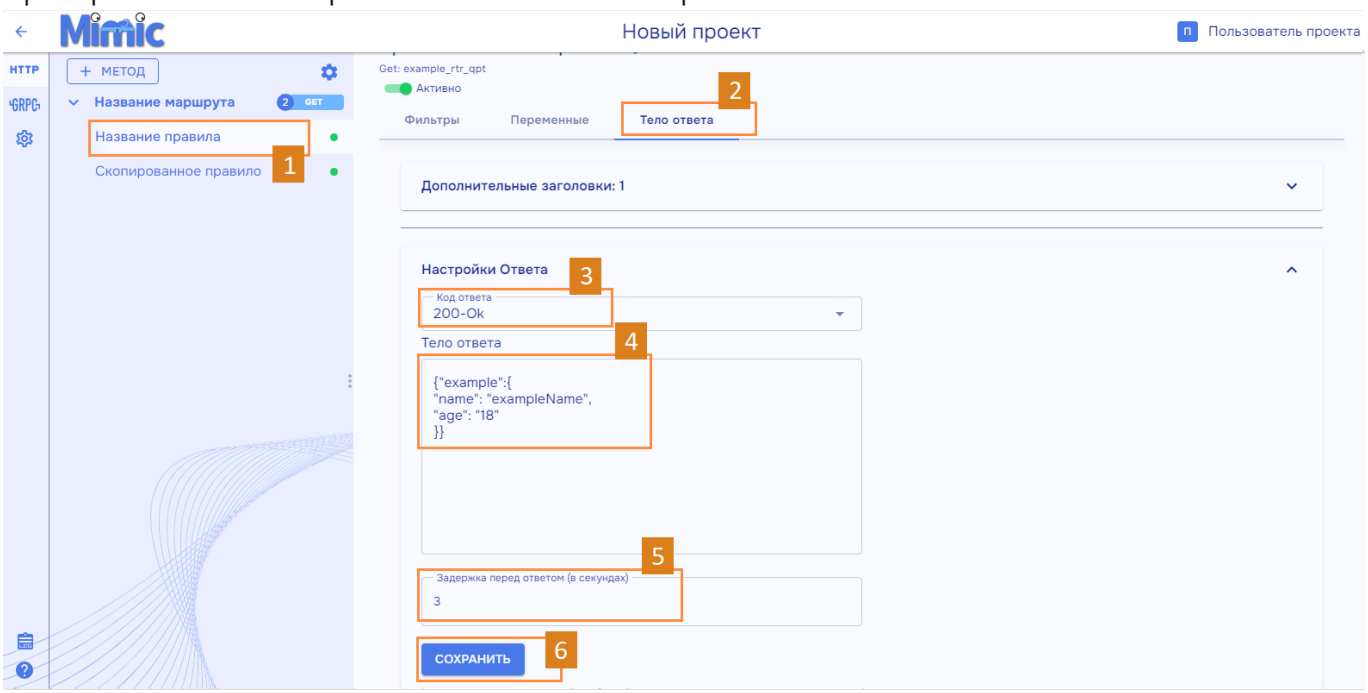
4. Указываем **Наименование** заголовка;

5. Указываем **Значение** заголовка;

6. Нажимаем ✓ для сохранения.

Установка кода и тела ответа

Пример заполнения настроек ответа в теле ответа правила:



Описание действий:

1. Переходим на страницу **Правила**;
2. Переходим во вкладку **Тело ответа**;
3. В блоке **Настройки ответа** выбираем из предложенного списка нужный **Код ответа**;
4. Заполняем **Тело ответа**;

5. Устанавливаем время **Задержки перед ответом** (в секундах). Если задержка перед выдачей ответа не нужна, то указываем 0;
6. Нажимаем кнопку **Сохранить**.

В ответе можно использовать значения переменных: (локальных, глобальных и маршрута). Для того чтобы их корректно обозначить и передать, необходимо:

Любую часть ответного сообщение можно заменить на переменную. Например, в ответе **"name": "Имя"** можно заменить текст "Имя" на значение переменной. Для этого, вместо заменяемого текста, нужно указать в двойных фигурных скобках имя ранее созданной переменной, например, **Var**. Получим **"name": "{{Var}}"**.

Для переменных, не являющихся локальными, в имени используется префикс:

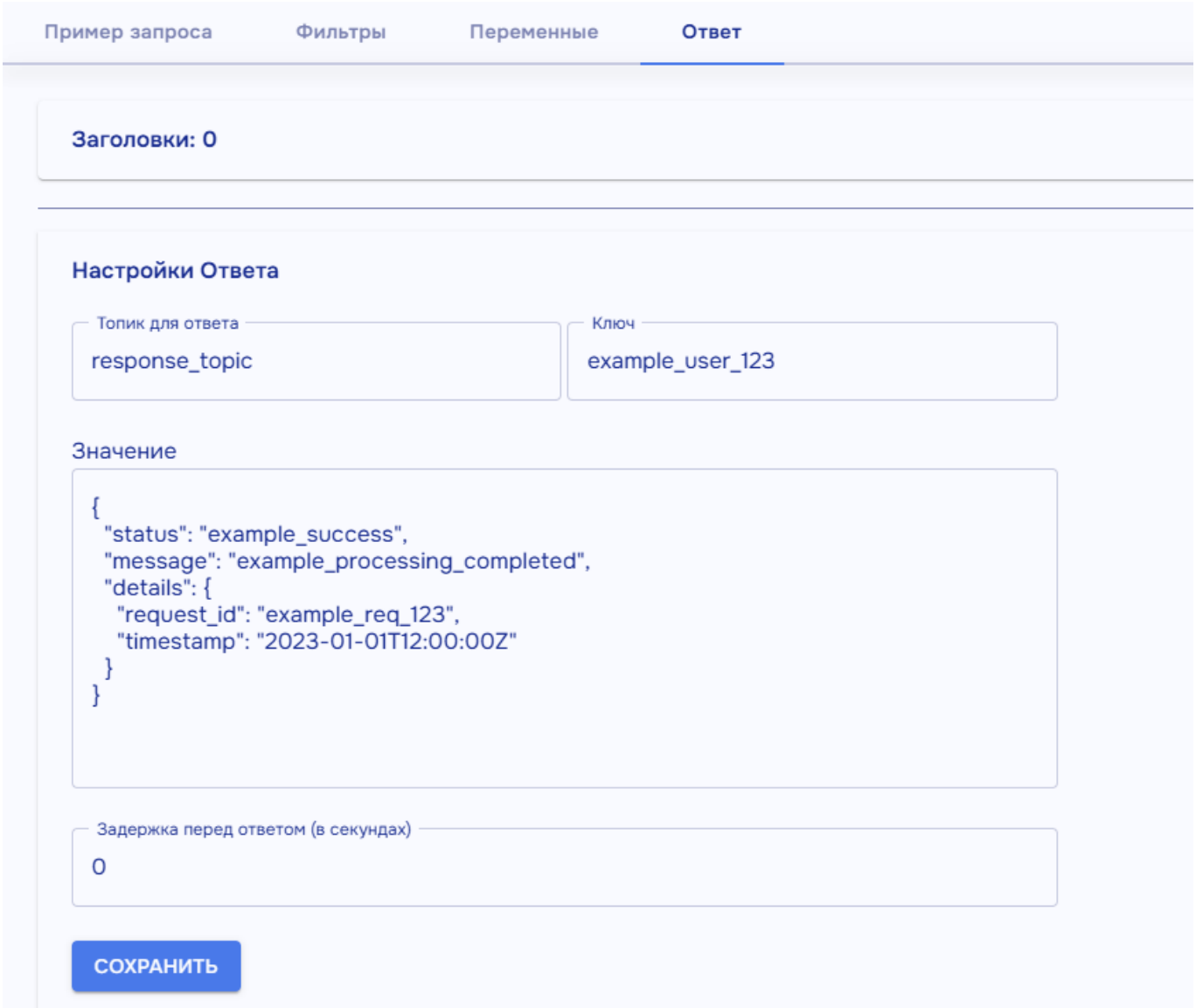
- **G:** - глобальные переменные
- **E:** - переменные маршрута

В данном примере можно увидеть, как используется глобальная переменная: **"age": {{G:age_request1}}**, где **age_request1** - название переменной, **G:** - указание на принадлежность к глобальности переменной.

Ответное сообщение для Kafka

Ответное сообщение Kafka - это настраиваемая структура данных, которая отправляется в указанный топик Kafka при срабатывании всех условий правила. Сообщение содержит ключ, значение и может

включать дополнительные заголовки для маршрутизации и обработки.



Топик для ответа - топик, в который будет отправлен ответ при срабатывании правила.

Ключ - поле, в котором можно задать значение для передачи в ключе kafka-события. Также возможно задать [{guid}], тогда для ключа сообщения ответа будет сгенерировано guid-значение.

Права пользователей

Создавать ответные сообщения и управлять ими могут Владелец проекта и Пользователь, просматривать - Читатель.

Управление правилами и ответами по умолчанию через API

> **Активировать или дективировать правило/ответ по умолчанию возможно посредством API.**

API (Application Programming Interface) — это интерфейс, позволяющий программам взаимодействовать и обмениваться данными друг с другом.

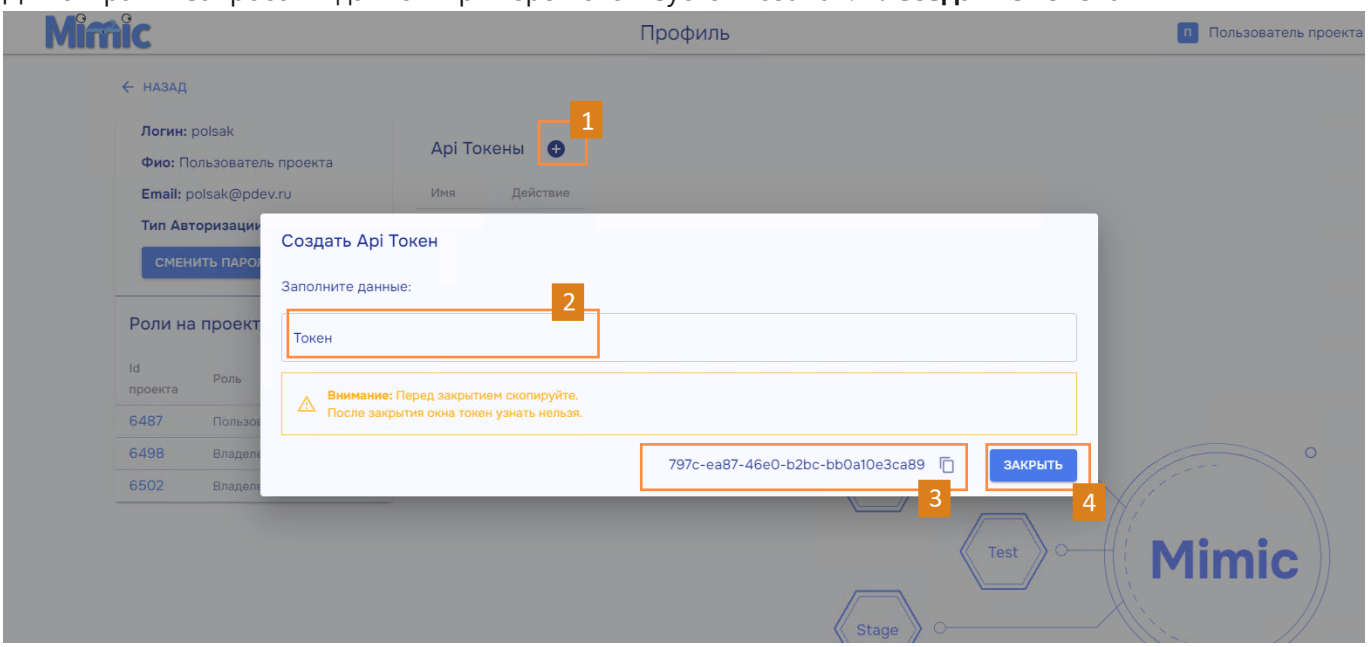
Для управления правилами/ответами по умолчанию посредством API, пользователю необходимо создать токен в системе Mimic.

> **Токен** — это уникальный идентификатор, который используется для аутентификации и авторизации пользователя при доступе к ресурсам или сервисам.

Для изменения активности правила/ответа по умолчанию требуется вызвать **HTTP-метод PUT rule/activity** с двумя параметрами: **id** (идентификатор правила) и **isActive** (активность правила: **true** - правило активно, **false** - правило не аквтивно). **Одновременно** может быть **активно любое количество правил**, однако **ответ по умолчанию** может быть активен **только один**.

Активация правила посредством API

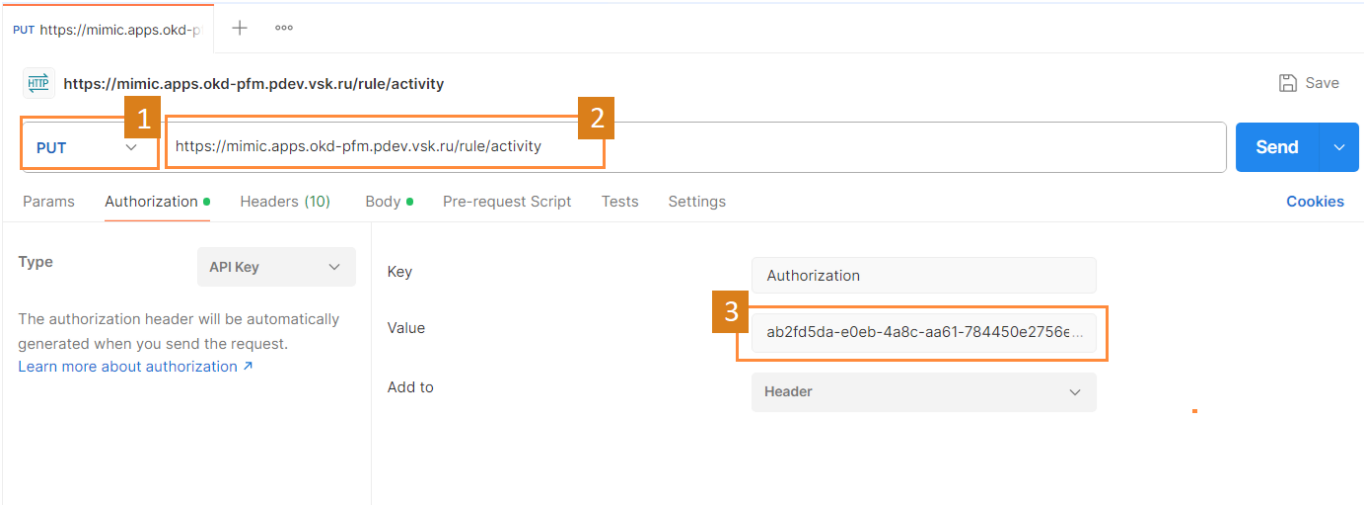
Для отправки запросов в данном примере используется Postman. **1. Создание токена**



Описание действий

1. В разделе **Мой профиль** нажимаем кнопку **Добавить Api токен**;
2. Вводим **Название** для токена и нажимаем кнопку **Сгенерировать**;
3. **Копируем токен**, который был сгенерирован системой. **После закрытия окна токен узнать нельзя**;
4. Закрываем окно;

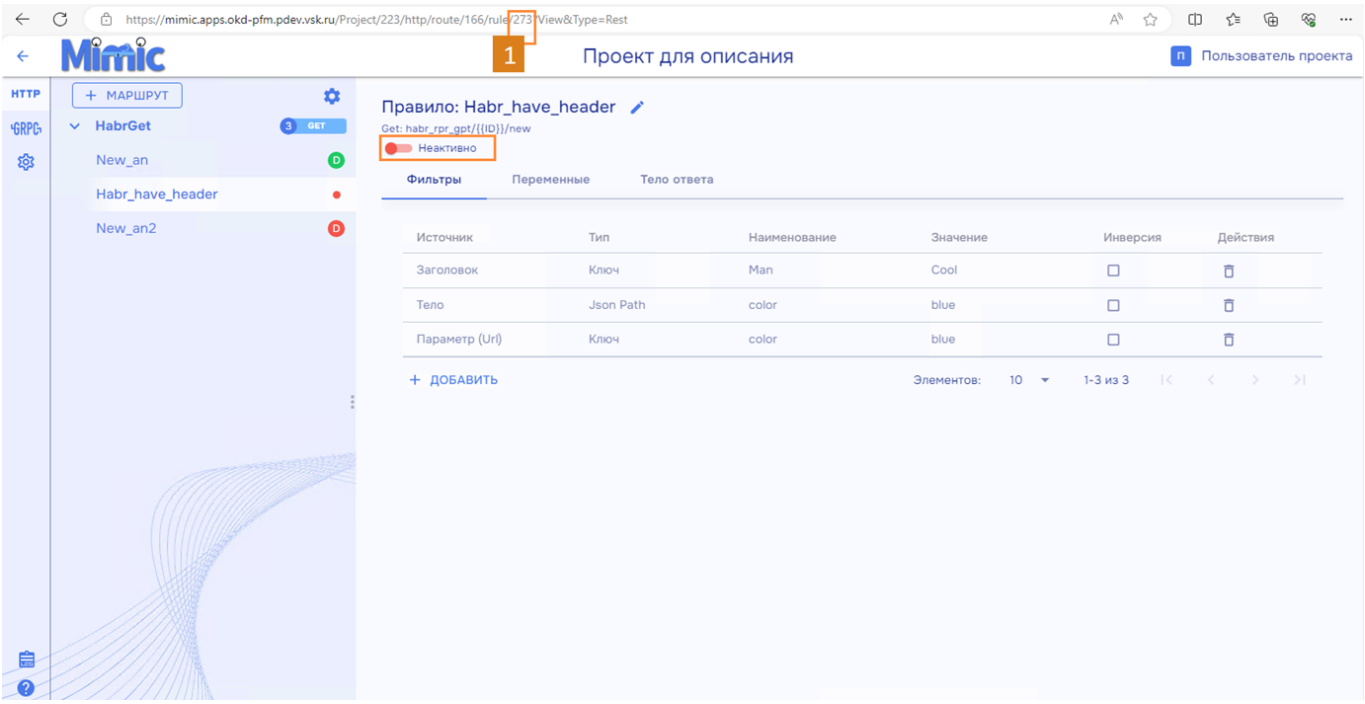
2. Заполнение адреса запроса и заголовка авторизации

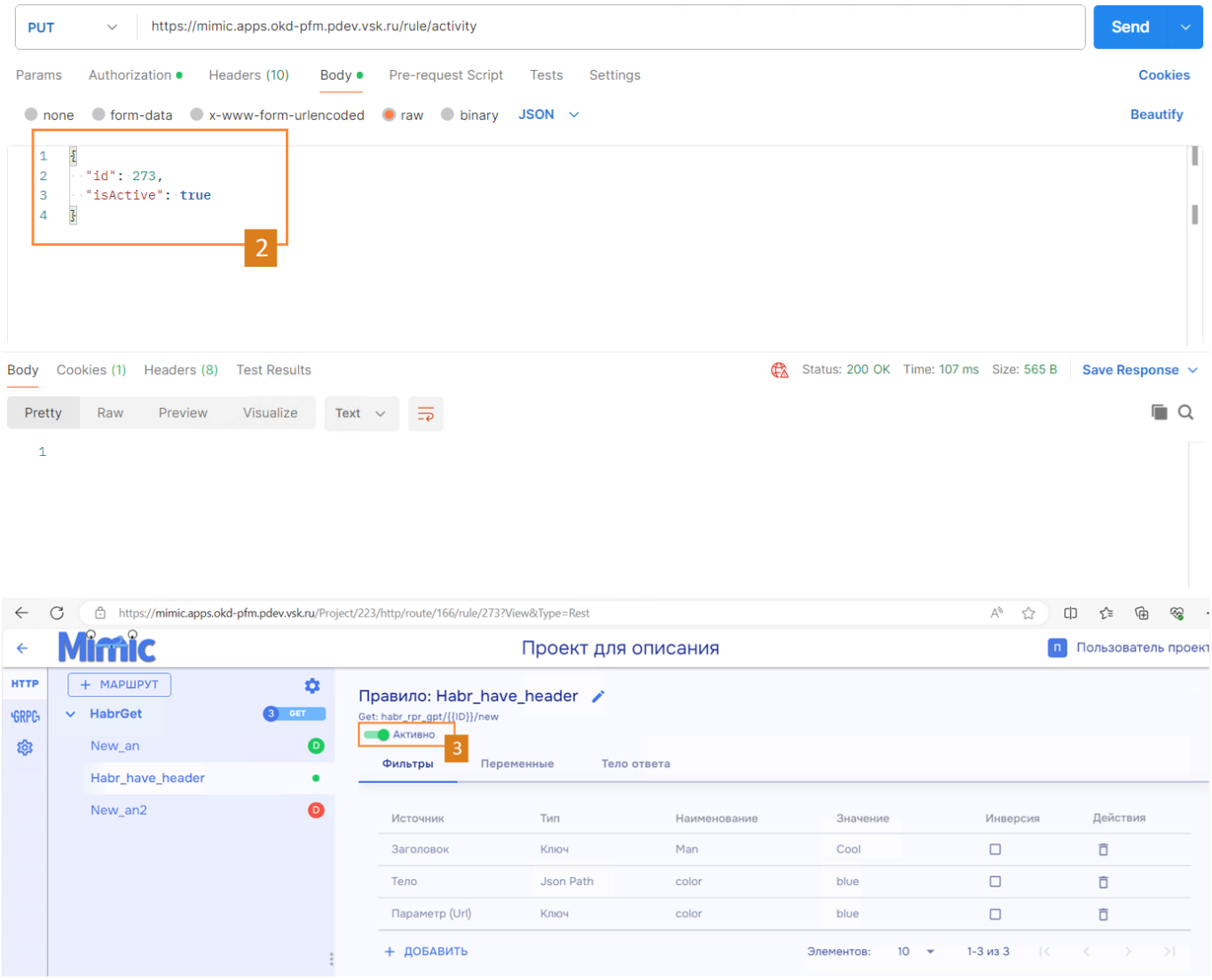


Описание действий

1. **Выбираем** глагол запроса **PUT**;
2. **Указываем адрес** для запроса **{host}/rule/activity**;
3. **Добавляем скопированный токен** в заголовок авторизации запроса.

3. Активация правила посредством Api





Описание действий

1. **Берем id правила** из URL (после rule/). В данном примере id правила - 273. **Обратим внимание, что в данный момент правило неактивно**;
2. **Подставляем id** правила, **признак** активности (true) запроса и отпавляем запрос; Пример запроса:

```
{
    "Id": 273,
    "IsActive": true
}
```

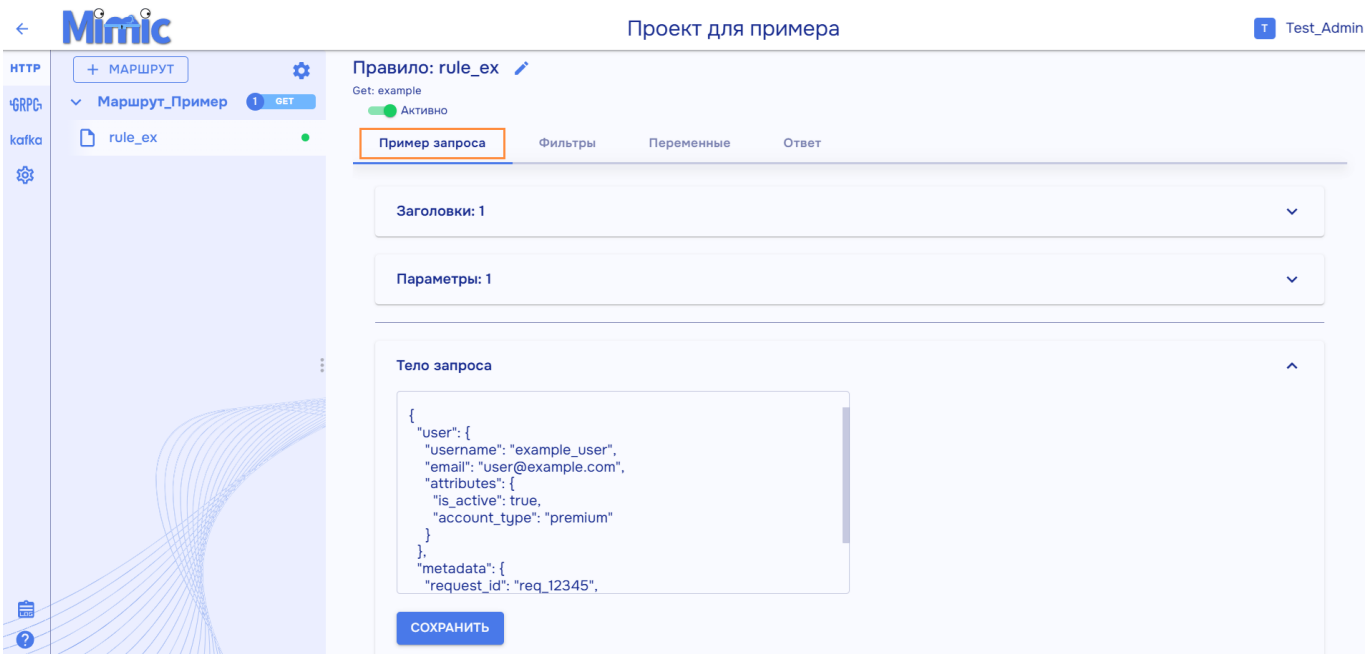
3. Заметим, что **в интерфейсе** Mimic **правило стало активным**.

Пример запроса

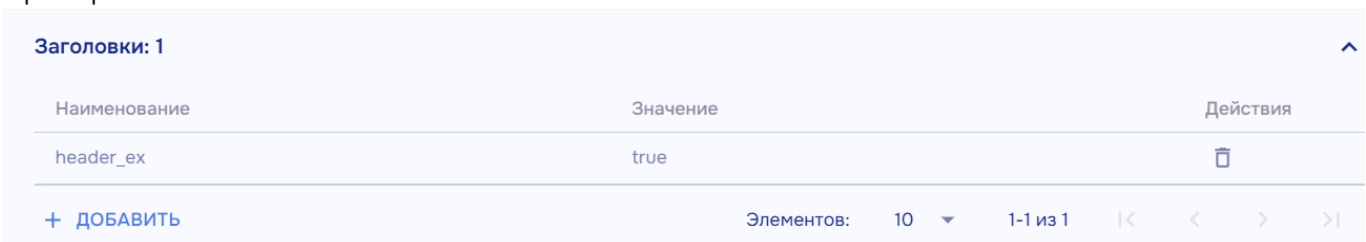
Данный раздел содержит **примеры запросов к указанному эндпоинту API для HTTP**, включая **обязательные заголовки, параметры и структуру тела запроса**. **Для Kafka** представлен **шаблон сообщения для корректной публикации в указанный топик Kafka**, а также **заголовки**. Приведенные на данной странице примеры демонстрируют **корректный формат данных для успешного взаимодействия с API** и могут быть использованы для тестирования интеграции или валидации запросов.

Все примеры заполняются пользователем вручную.

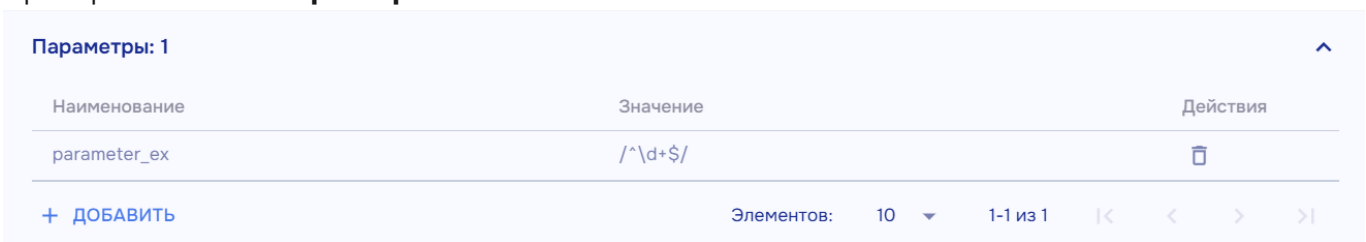
Страница "Пример запросов" для HTTP



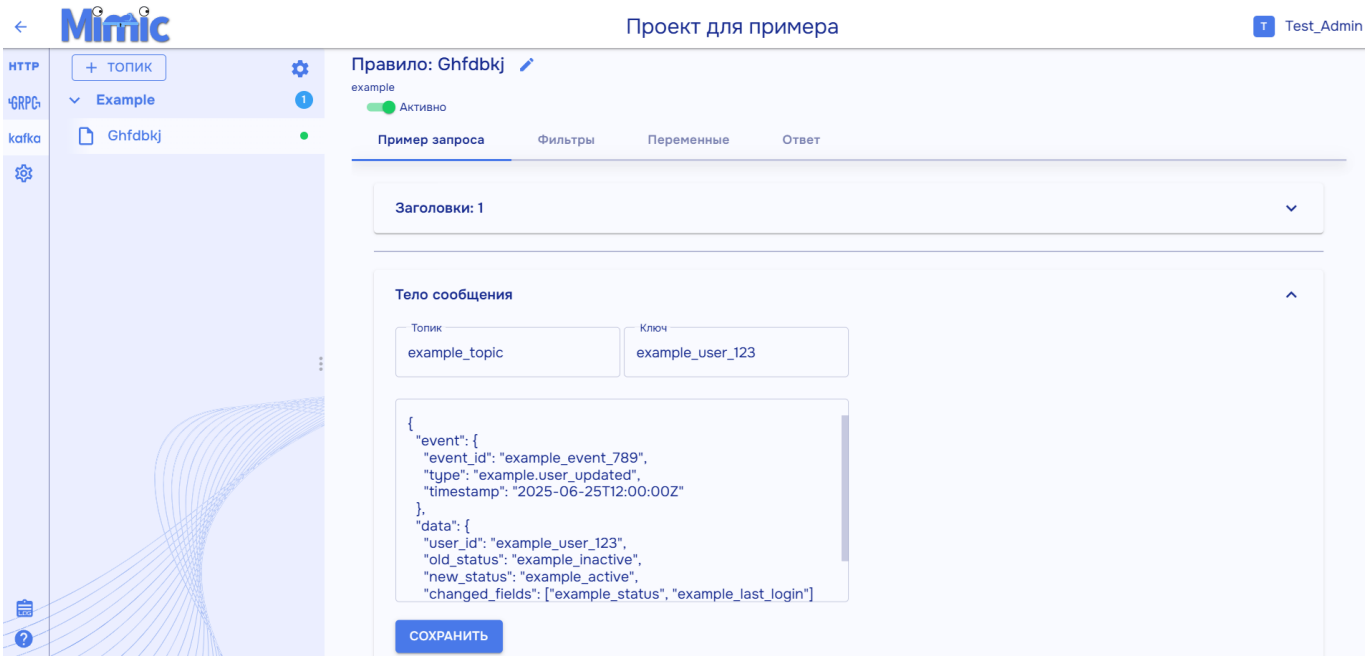
Пример заполнения **Заголовков**:



Пример заполнения **Параметров**:



Страница "Пример запросов" для Kafka



Права пользователей

Управлять примерами запросов могут Владелец проекта и Пользователь, просмотривать - Читатель.

Ответы по умолчанию

> **Ответ по умолчанию** - это ответ, который выдается системой, когда не сработало ни одно из настроенных на маршруте правил или эти правила отсутствуют.

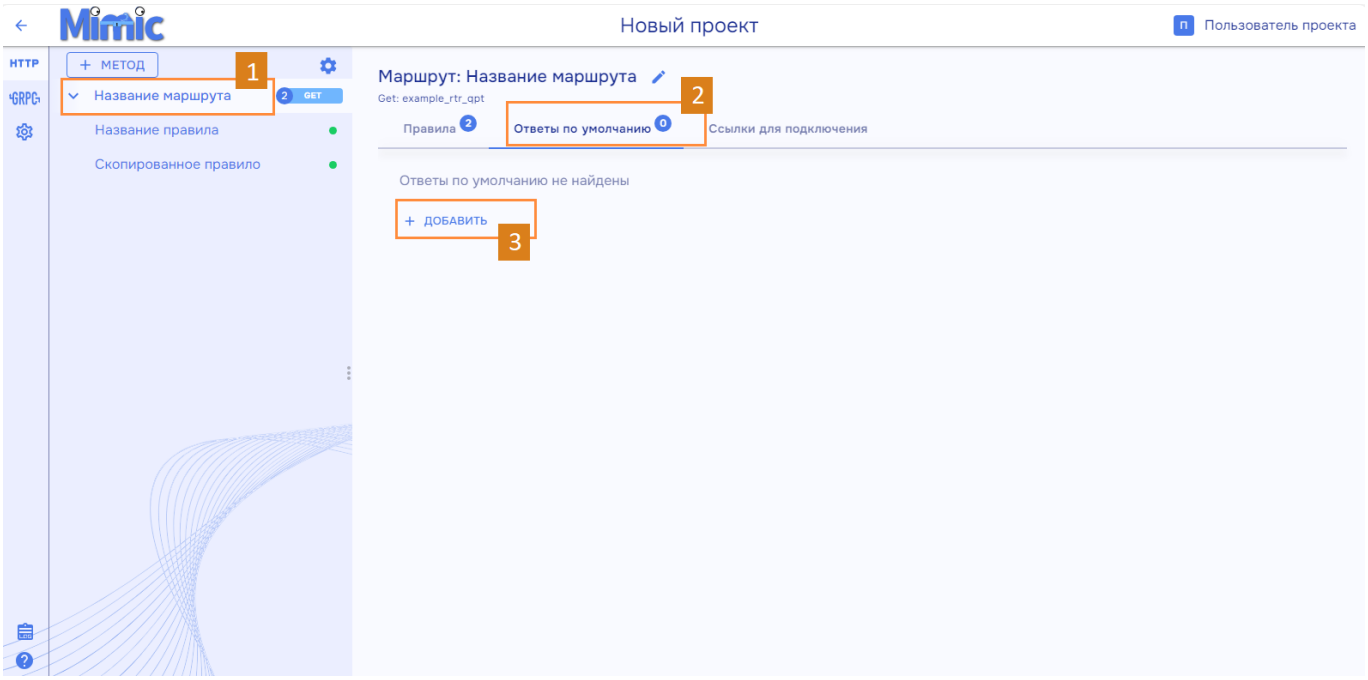
Логика выдачи ответа по умолчанию:

- Если нет правил или ни одно правило не сработало, то выдаем ответ из активного правила по умолчанию
- Если хотя бы одно правило сработало, то тело ответа будет браться из него

> Если есть активное правило по умолчанию, то проксирование работать не будет!

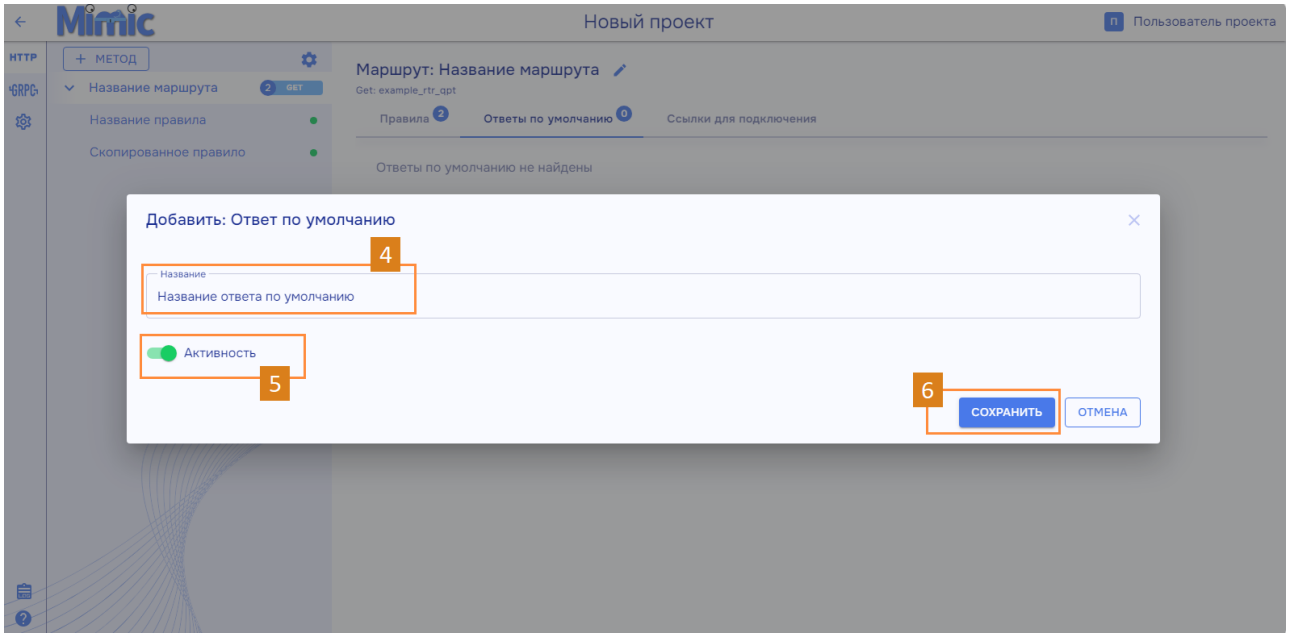
Настройка ответа по умолчанию

Пример добавления правила по умолчанию в маршруте у HTTPS (добавление метода в gRPC происходит аналогично) **Описание действий:**



1. Переходим на страницу необходимого **Маршрута**;
2. Раскрываем блок **Ответы по умолчанию**;

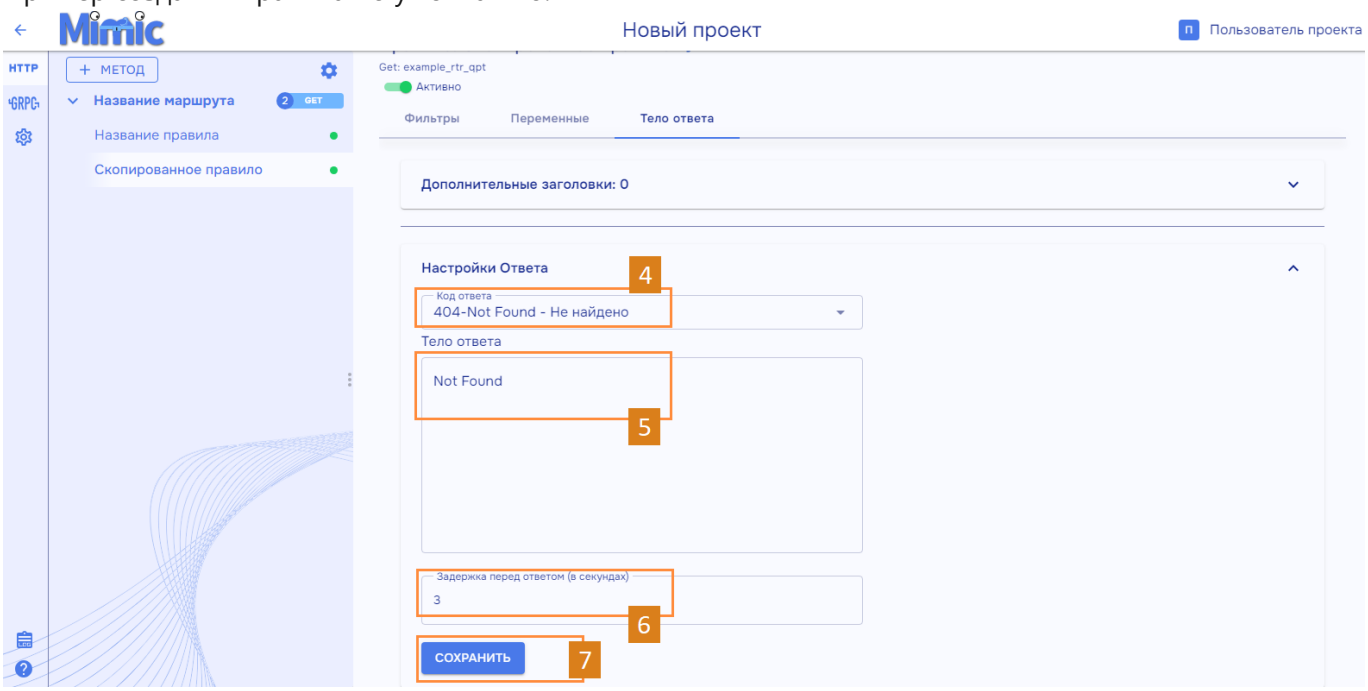
3. Нажимаем кнопку **Добавить**;



4. Указываем **Название правила** - может быть любым, используется только для удобства поиска в интерфейсе;
5. **Активность** ответа по умолчанию переключаем на зеленый, если это правило нужно использовать сейчас, и оставляем красным, если временно использовать не нужно. Также менять активность ответа по умолчанию возможно посредством API;
6. **Сохраняем** ответ по умолчанию нажатием соответсвующей кнопки.

Заполняем переменные и формируем тело ответа

Пример создания правила по умолчанию:



Описание действий:

1. Если требуется добавляем **Переменные** во вкладке **Переменные**;
2. Переходим во вкладку **Тело ответа**;
3. Если требуется добавляем **Доболнительные заголовки**;

4. В разделе **Настройки Ответа** выбираем **Код ответа**;
5. Формируем **Тело ответа**;
6. Заполняем **Задержку перед ответом**;
7. Нажимем кнопку **Сохранить**.

Процесс настройки **переменных и тела ответа** для правила по умолчанию не отличается от аналогичного при создании обычного правила и подробно описан в разделах:

- Создание переменных
- Формирование тело ответа

> **Права пользователей**
>
> Создавать ответы по умолчанию и управлять ими могут Владелец проекта и Пользователь, просмотривать - Читатель.

Журнал запросов

> **Журнал запросов** представляет собой список всех входящих запросов, которые отпраляются в систему.

Журнал содержит следующую информацию о каждом запросе:

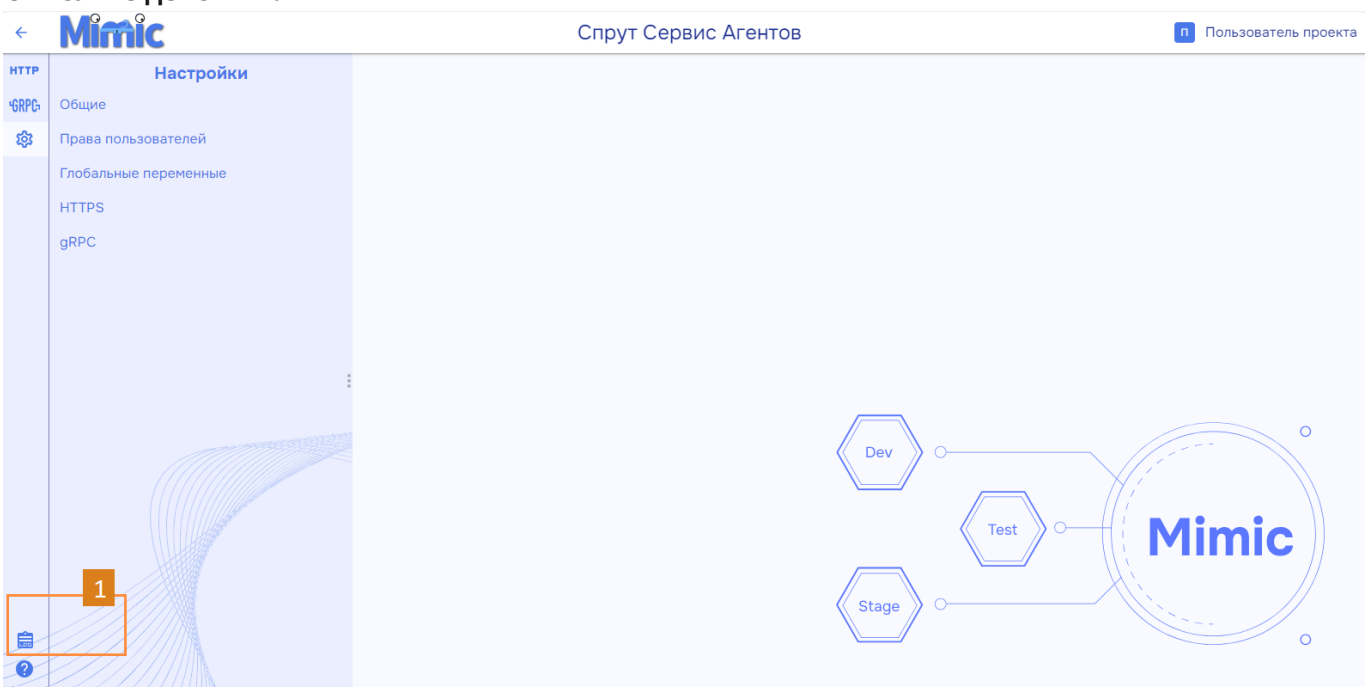
- Дата
- Время
- Стенд (Dev/Test/Stage/Prod)
- Проект
- Тип запроса (Get/Post/Put/Patch/Delete)
- Endpoint
- Правило

Обращение к Журналу запросов может осуществляться как с главной страницы Mimic, так и с конкретного проекта, маршрута или правила - для удобства навигации и поиска необходимой информации. В каждом случае будет использоваться определенный фильтр: например, при доступе из проекта будет применен фильтр по проекту.

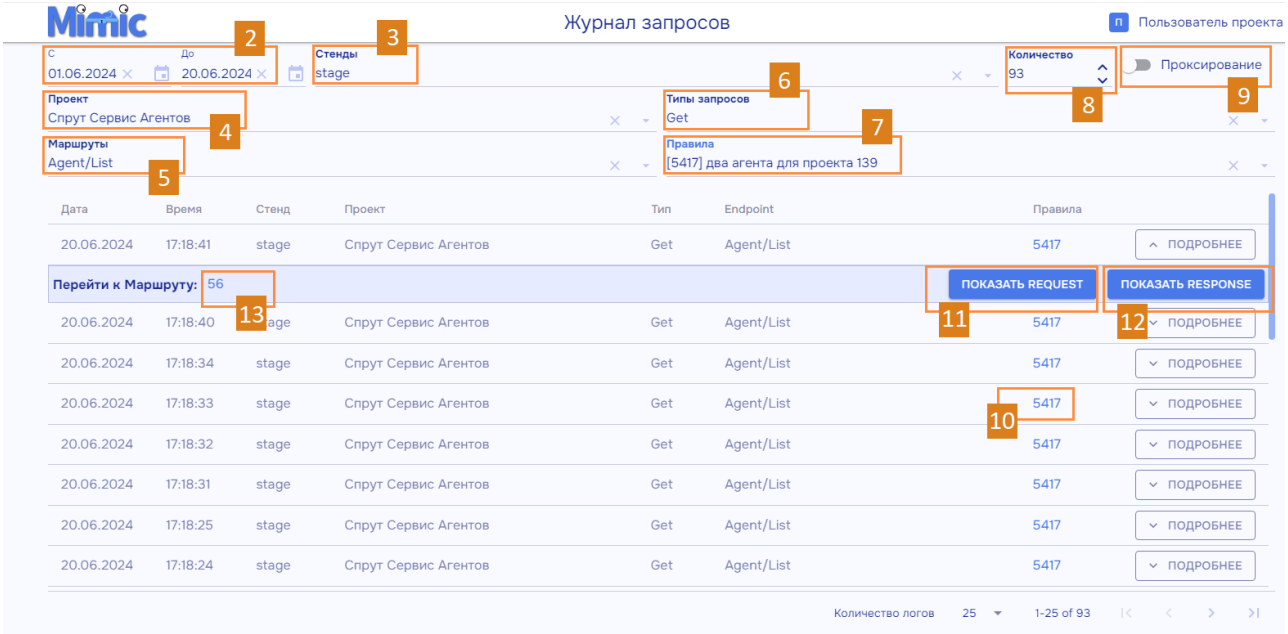
Для доступа из Mimic и проектов - кнопка .

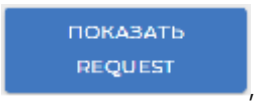
Пример работы с Журналом запросов с главной страницы Mimic:

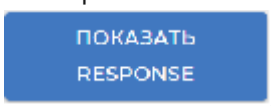
Описание действий:



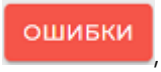
1. **Переходим в Журнал запросов** посредством нажатия на иконку  в боковом меню проекта;



2. Указываем необходимый **диапазон дат** (по умолчанию дата текущая);
3. Выбираем нужный **Стенд**
4. При необходимости меняем **Проект** (по умолчанию проект, из которого был осуществлен переход в Журнал);
5. Выбираем **Маршруты**;
6. Выбирем интересующие **Типы запросов**;
7. Выбираем **Правила**;
8. При необходимости меняем **Количество запросов**;
9. При необходимости переводим чекбокс **Проксирование** в активное положения для просмотра запросов, по которым не сработало правило и был выполнен поход напрямую в сервис;
10. Если в запросах **сработало Правило**, то возможно просмотреть это правило, нажав на активный номер соответствующего правила;
11. В запросах, по которым было найдено правило и сформирован ответ Mimic, есть возможность просмотра **Исходящего запроса** посредством нажатия на кнопку 

, где указывается информация о: проекте, маршруте, задержке ответа, заголовках и теле запроса (Ответы от проксируемых запросов просматривать нельзя);
12. В каждом запросе есть возможность посмотра **Входящего ответа** посредством нажатия на кнопку 

, где указывается информация о: проекте, маршруте, коде ответа, параметрах, теле ответа и дополнительных заголовках;
13. Также существует возможность просмотра **сработавшего Маршрута** посредством нажатия на активный номер соответствующего маршурта.

Если при обработке запроса **произошла ошибка**, то соответствующая строка маршрута подсвечивается красным цветом, и в описании запроса появляется кнопка Ошибки 

, при нажатии на которую можно ознакомиться с текстом ошибки.

> **Права пользователей**

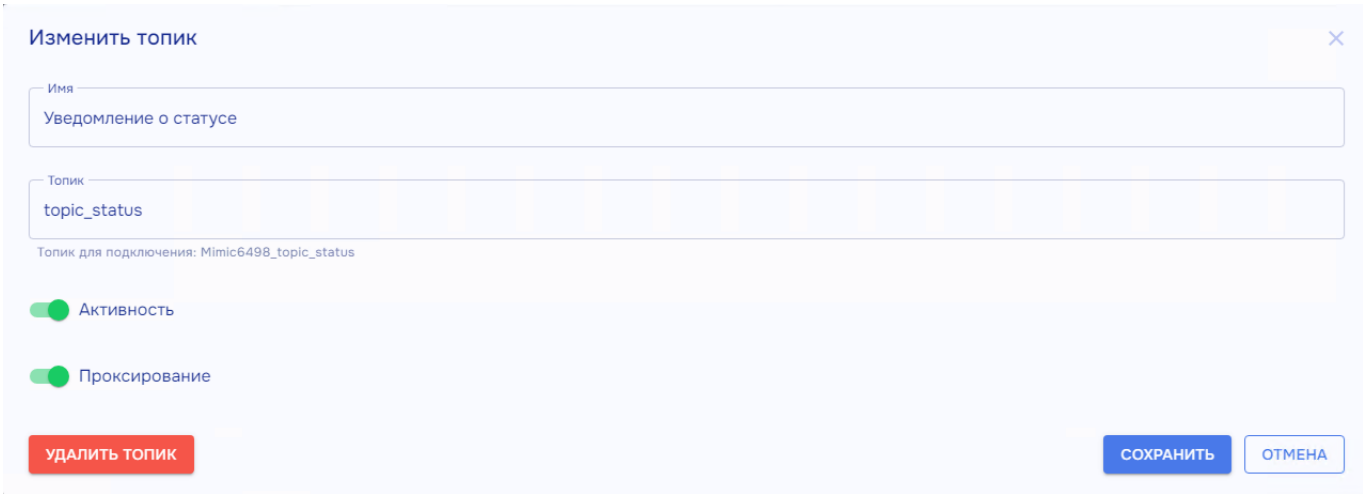
> Просматривать журнал запросов могут пользователю с любой ролью в проекте.

Топики (Kafka)

> **Топик** - это базовая сущность мокирования Kafka, в рамках которой пользователь может делать настройки для нужного отображения ответа по созданным правилам.

Страница топика включает в себя следующие блоки: **Редактирование топика**, **Правила**, **Ответы по умолчанию**, **Проксирование**, **Настройки Kafka**.

Редактирование топика позволяет изменить **имя топика в системе Mimic**, **название** топика, **активность** и **проксирование**.



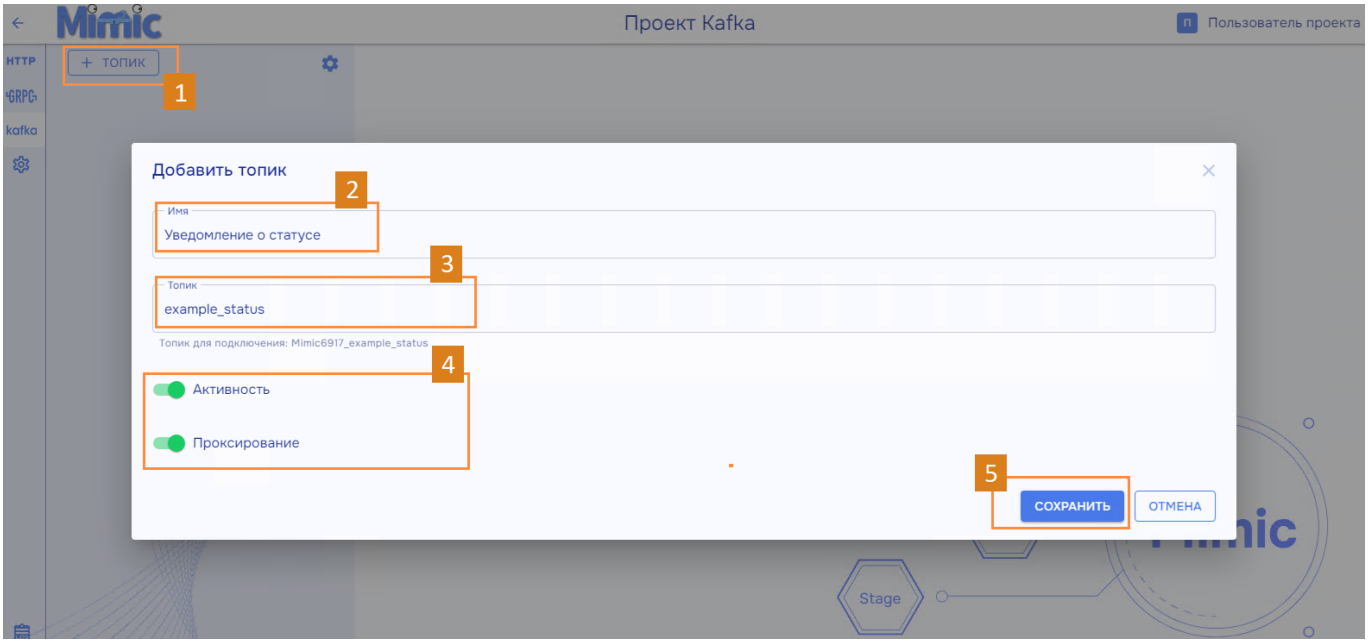
Добавление топика

Добавление топика может быть выполено ***двумя способами***: ***1)*** путем добавление нового топика с заполнением всех соответствующих полей; ***2)*** путем добавления нового топика на основе уже существующего - копирование топика.

> Если у проекта отсутствуют топики, то создание нового возможно только первым способом - с заполением всех полей.

В остальных случаях возможен второй способ, то есть создание топика на основе уже существующего (копирование) с его дальнейшим редактированием. При копировании топика все его правила также будут скопированы.

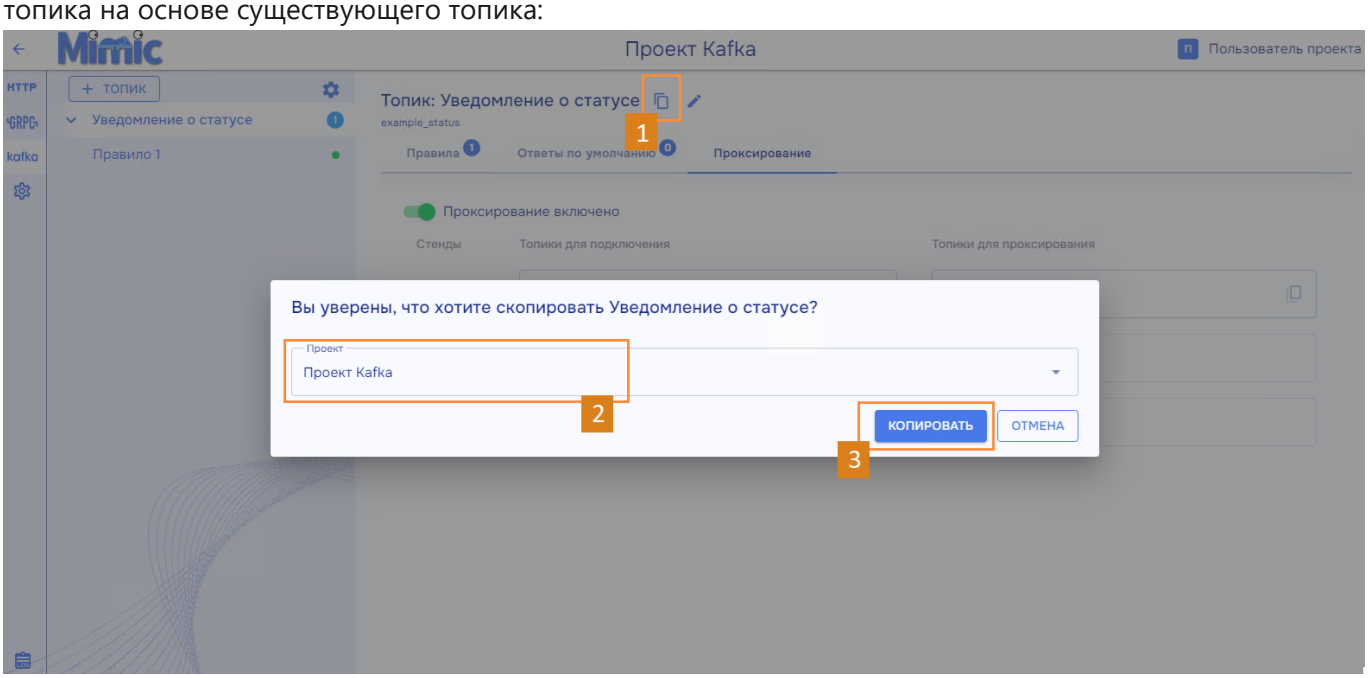
1. Добавление нового топика с заполнением полей Пример добавления топика:



Описание действий

1. Нажать кнопку **Добавления топика**, располагающуюся в верхней части меню топиков;
2. Указать **Имя топика в системе Mimic**;
3. Указать **Топик**;
4. При необходимости переключить **Активность** и **Проксирование** в нужное состояние;
5. Нажать **Сохранить**.

2. Добавление топика на основе существующего (копирование топика) Пример добавления топика на основе существующего топика:

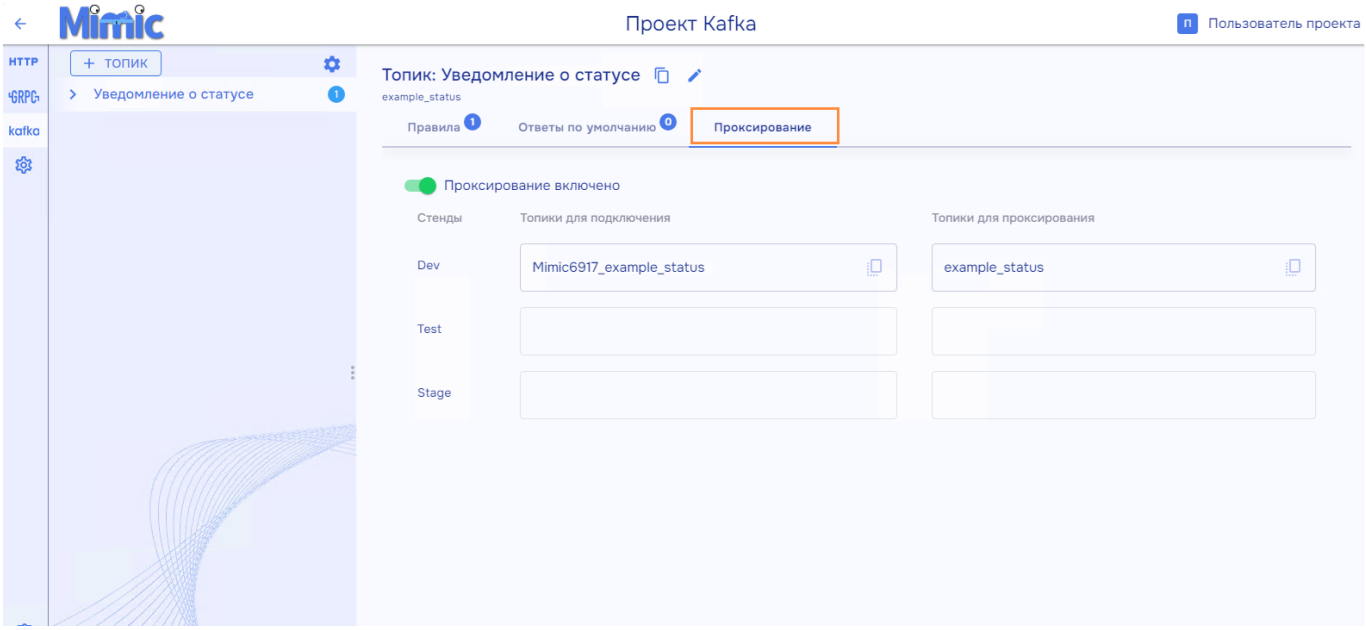


Описание действий

1. На странице необходимого для копирования топика нажимаем кнопку **Копировать** ;
2. Выбираем **Проект**, в который необходимо скопировать данный топик;
3. Нажимаем кнопку **Копировать**.

Проксирование

В блоке **Проксирование** для каждого стенда указан топик для подключения и топик для проксирования. Также на данной странице возможно **отключение проксирования** с помощью флага активности.



Топик для подключения - это замещающий топик, к которому будет подключен сервис.

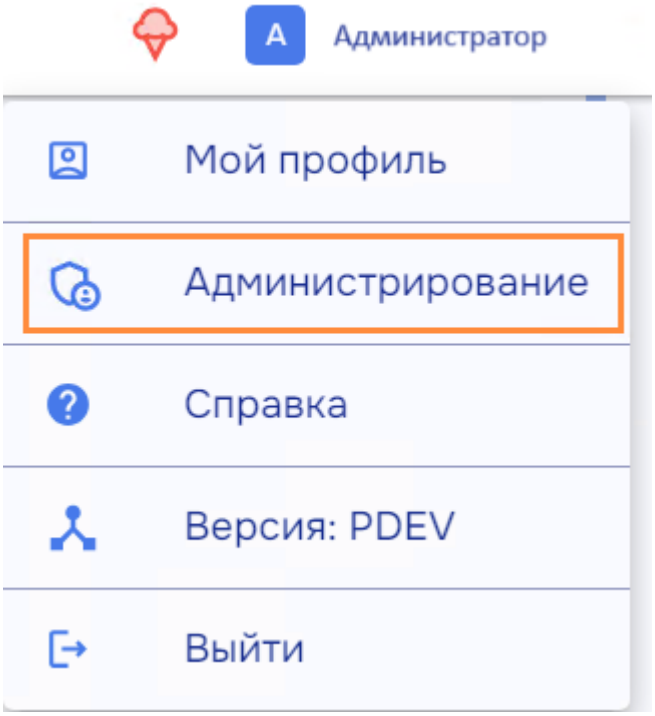
Топик для проксирования - это реальный топик, в которое будет отправлено сообщение, если не сработало ни одно правило, нет активных ответов по умолчанию и включено проксирование.

Права пользователей

Создавать топики и управлять ими могут Владелец проекта и Пользователь, просматривать - Читатель.

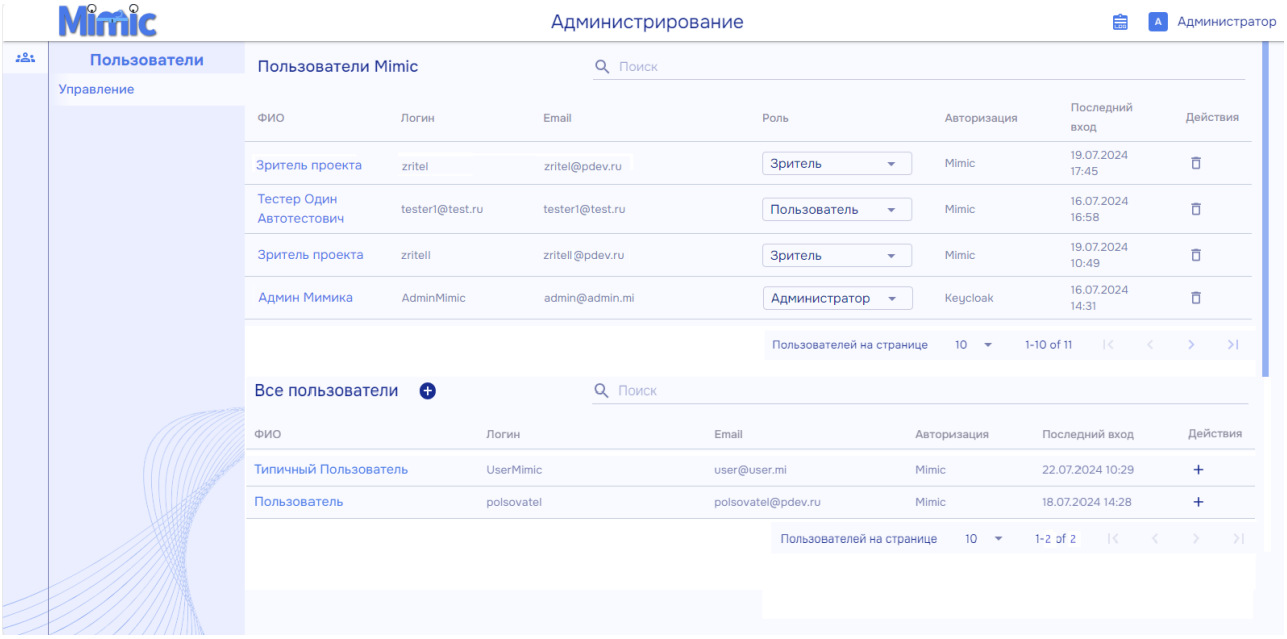
Администрирование

Для перехода к функциям администратора необходимо использовать интерфейс **Меню профиля (аватар)** → **Администрирование**. **Администрирование позволяет:** управлять пользователями платформы, их ролями, а также добавлять новых пользователей.



Интерфейс раздела **Пользователи** разделен на **два блока**:

1. **Глобальные роли**. Пользователи с глобальными ролями, чьи полномочия распространяются на все проекты платформы.
2. **Все пользователи**. Все пользователи, зарегистрированные в системе.



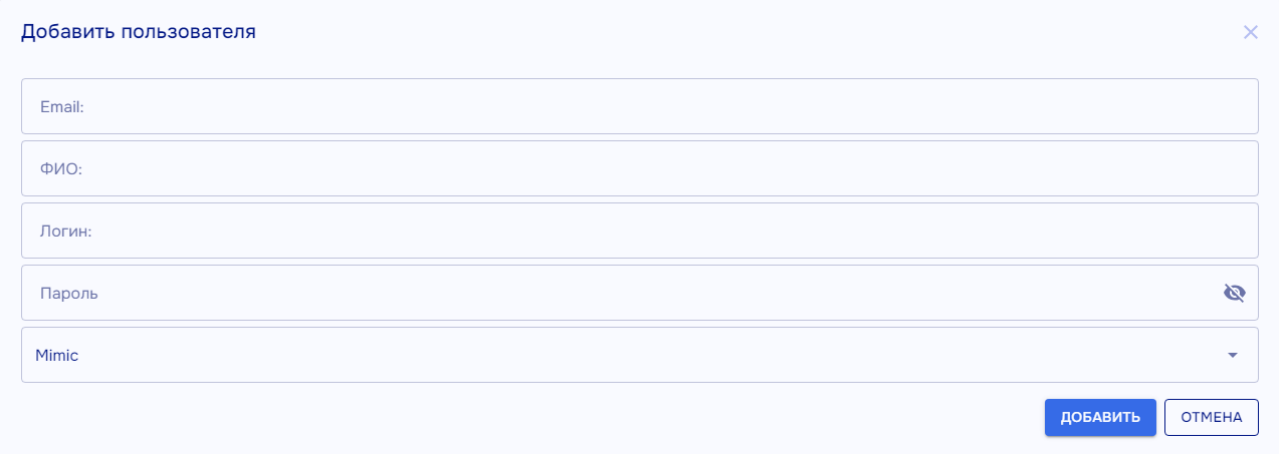
По каждому полю **доступна сортировка**, для этого необходимо нажать на соответствующее поле в интерфейсе системы. Введите текст в **поле поиска**, чтобы увидеть только тех пользователей, чьи ФИО

или Email содержат этот текст. По умолчанию список разбит на страницы, на каждой из которых может быть отображено до 10, 25, 50, 100 (в зависимости от выбора) пользователя. Используйте **стрелки внизу списка** для переключения между страницами.

Добавление пользователя в систему Mimic

Если пользователь производит **вход через Oauth2-авторизацию**, например KeyCloak, то этот пользователь **автоматически попадает в Список пользователей** при **первом входе в систему**. Иначе необходимо добавить **новую учетную запись**:

1. Нажмите на свой аватар и перейдите в раздел **Администрирование**;
2. В списке **Все пользователи** нажмите на иконку добавления ⊕;
3. **Введите необходимые данные** и нажмите на кнопку **Добавить**;



   Email - электронный адрес пользователя; ФИО - фамилия, имя и отчество пользователя; Логин - произвольный логин пользователя (латиницей); Пароль - пароль, который будет передан пользователю (может состоять из латинских, кириллических букв, цирф и символов); Mimic/Keycloak - способ авторизации пользователя (при добавлении пользователя администратором всегда "Mimic");
4. **Передайте логин и пароль** пользователю.

Глобальные роли

В данном разделе описывается управление **глобальными ролями**, которые влияют на права пользователей при использовании платформы Mimic.

Описанный интерфейс администрирования **доступен только пользователям с глобальной ролью "Администратор"**. Данные пользователи обладают возможностью управления всеми пользователями и их ролями.

Глобальная роль пользователя распространяется на все проекты, существующие на платформе Mimic, если в отдельном проекте не указана иная роль. **Не все пользователи имеют глобальную роль**, для работы с проектом пользователю назначается **проектная роль**. Про проектные роли подробнее в Проектные роли.

Учетной записи, зарегистрированной в Mimic, может быть присвоена **одна из трех глобальных ролей**:

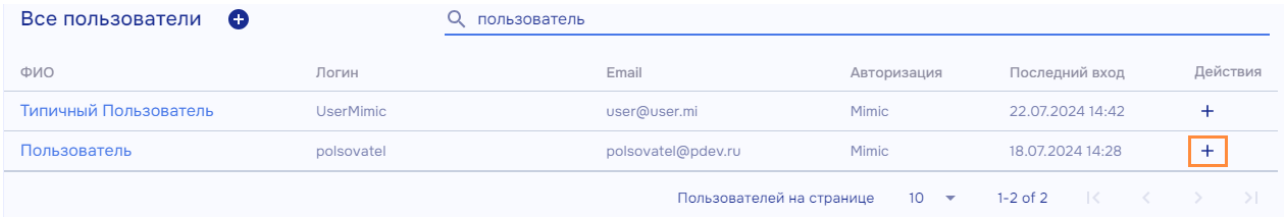
- **Администратор**. Может просматривать и вносить изменения во все проекты платформы Mimic, добавлять пользователей в систему, назначать глобальные роли.

- **Пользователь**. Может просматривать и вносить изменения во все проекты платформы Mimic.
- **Зритель**. Может просматривать все проекты платформы Mimic.

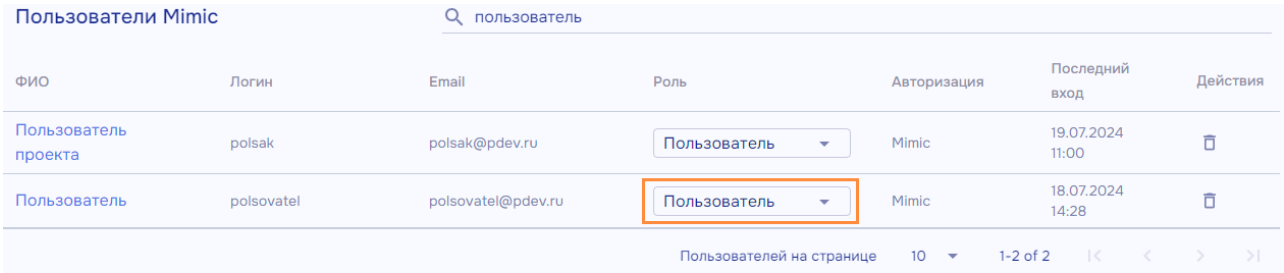
Назначение глобальной роли пользователю

Чтобы добавить пользователя в **Глобальные роли** (пользователи с глобальной ролью):

1. **Нажмите на свой аватар** и перейдите в раздел **Администрирование**;
2. В таблице **Все пользователи** найдите необходимого пользователя;
3. **Нажмите на кнопку** + в соответствующем поле строки пользователя;



4. В таблице **Глобальные роли** найдите необходимого пользователя и **измените глобальную роль** на желаемую (по умолчанию роль: Пользователь).



Если учетная запись пользователя еще не заведена, то сначала необходимо добавить пользователя в систему!

Изменение глобальной роли учетных записей

Чтобы **преобразовать учетные записи** между глобальными ролями "Администратор", "Пользователь" и "Зритель":

1. **Нажмите на свой аватар** и перейдите в раздел **Администрирование**;
2. В списке **Глобальные роли** найдите **необходимого пользователя**;
3. **Выберите желаемую роль** в выпадающем списке поля **Роль**.

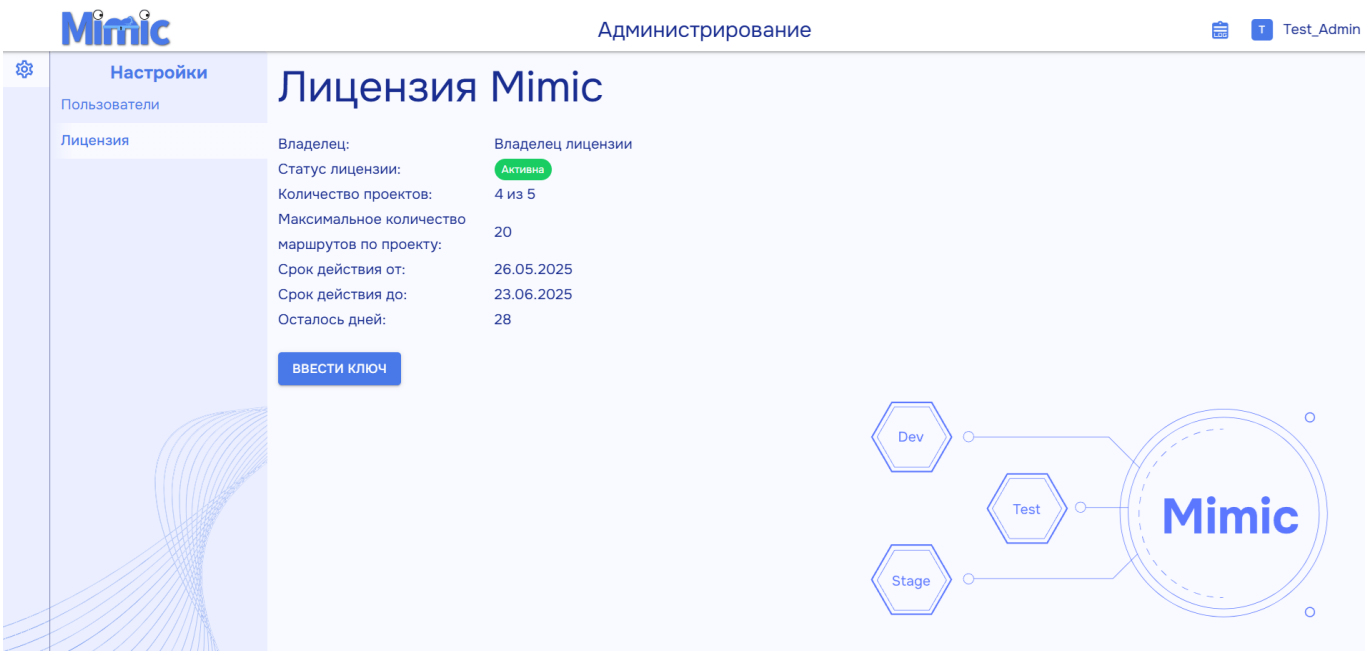
Для удаления учетной записи из **Глобальные роли** (пользователи, имеющие глобальную роль) необходимо нажать на **значок корзины** в соответствующей строке. **После этого учетная запись будет доступна в списке Все пользователи (не имеющие глобальную роль).**

Лицензия

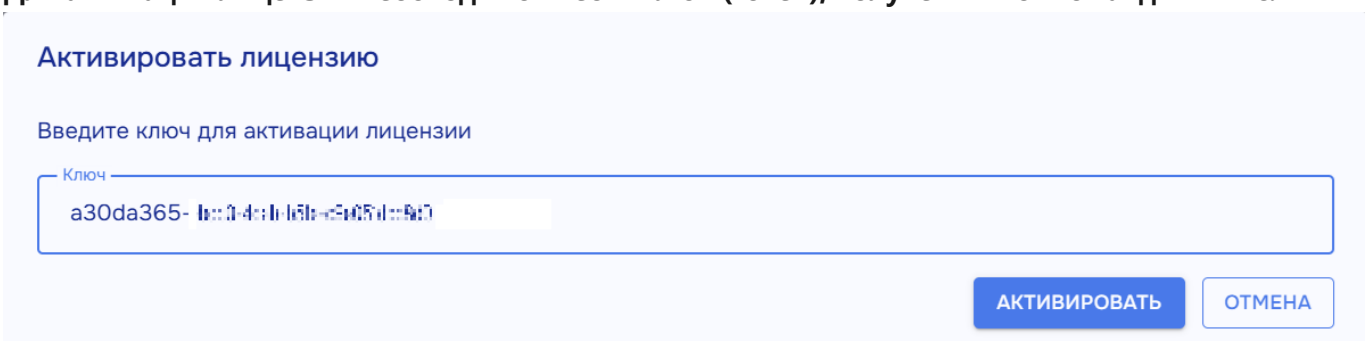
> **Лицензия** - это ключ, который предоставляет пользователю или компании право на использование системы Mimic в соответствии с установленными условиями.

Условия лицензии: 1. Устанавливает ограничение на срок использования продукта Mimic. По истечению срока действия лицензии создание новых объектов в системе будет невозможно, работа с существующими данными (запросы, просмотр) останется доступной. **2. Ограничивает максимальное количество проектов.** Попытка создания проекта сверх установленного лимита будет заблокирована системой. В случае если продленная лицензия будет рассчитана на меньшее количество проектов, то доступ к существующим проектам будет ограничен. Чтобы возобновить работу, необходимо удалить лишние проекты, приведя их количество в соответствие с новой лицензией.

3. Ограничивает максимальное количество маршутов на каждый проект. Данное ограничение действует независимо для каждого проекта в рамках лицензии. Например, при лимите в 10 проектов и 10 маршрутов система будет поддерживает до 100 маршрутов суммарно. Создание дополнительных маршрутов сверх установленных лимитов блокируется автоматически. **4. Не ограничивает количество пользователей в системе.**



Для активации лицензии необходимо ввести ключ (токен), полученный от команды Mimic:



Список изменений

Изменения в версии 1.5.3 (текущая)

- Исправлено некорректное отображение интерфейса в Mozilla Firefox
- Добавлена возможность настройки периодической очистки логов по проекту
- Добавлена возможность ручной очистки логов по фильтрам

Изменения в версии 1.5.2

- Добавлена возможность имитировать запросы через топики Kafka
- Добавлено управление правилами через API
- Добавлена возможность копирования маршрута

Изменения в версии 1.5.0

- **Добавлена авторизация и ролевая модель в систему.** Теперь пользователи видят только свои проекты и могут выдавать доступы участникам. Доступы владельцев проектов будут выданы согласно таблице: https://confluence.vsk.ru/pages/viewpage.action?pageId=350194475
- **Добавлена возможность добавлять проекты в избранное**
- **Добавлена возможность создания GRPC соединения по рефлексии** Создание методов GRPC для имитаций происходит из фиксированного списка методов полученных из соединения GRPC
- **Добавлена возможность доступа к журналу запросов по API**
- **Увеличено тело ответа в правиле до 200 000 символов**