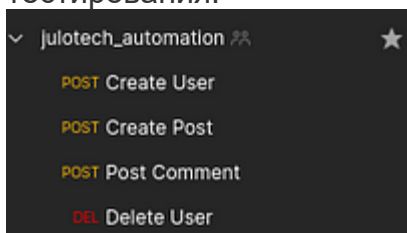


# Postman для эффективного тестирования API

Тестирование API является важной частью разработки программного обеспечения, но при выполнении вручную оно может отнимать много времени и включать в себя много повторяющихся задач. [Postman](#) является одним из наиболее широко используемых инструментов для тестирования API. Однако многие пользователи зачастую не используют его возможности автоматизации в полной мере, что приводит к снижению эффективности процесса тестирования API.

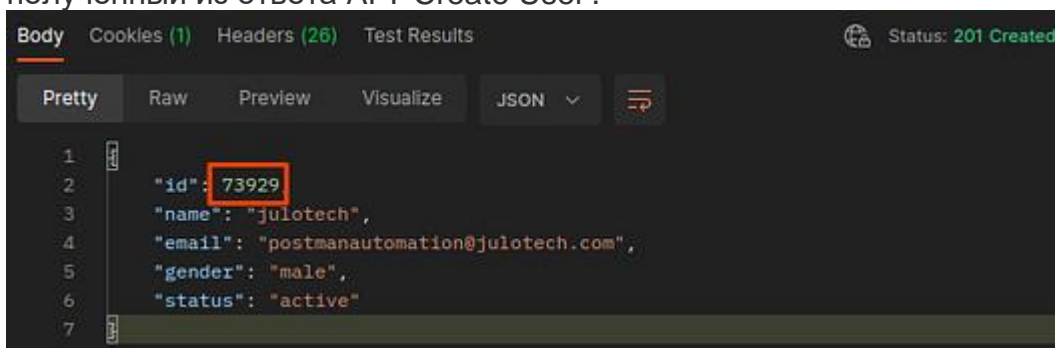
## Чувство безысходности от утомительных ручных шагов

В этом посте мы будем использовать коллекцию Postman Collection, представленную ниже. API, используемые в этой демонстрации, взяты с сайта [gorest.co.in](https://gorest.co.in), который предоставляет бесплатный фиктивный сервис REST API, предназначенный для тестирования.

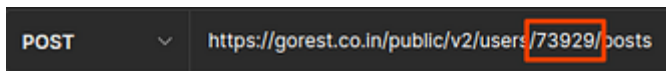


Названия запросов понятны и не требуют пояснений. Однако для контекста предположим, что эти API относятся к типичной системе онлайн-форума, которая включает функциональность для создания пользователей, сообщений, комментариев и удаления пользователей.

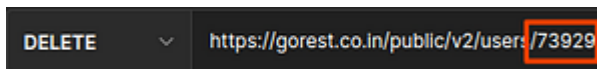
Интеграционное тестирование часто предполагает использование данных из ответа одного API в качестве параметра другого API. Например, чтобы протестировать API 'Create Post' и 'Delete User', необходимо добавить в URL ID пользователя, полученный из ответа API 'Create User'.



Получение user ID из успешного ответа API 'Create User'



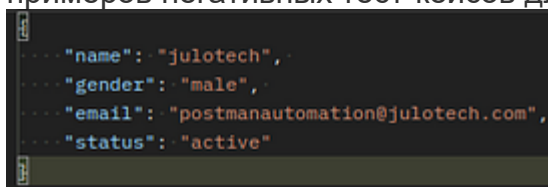
Размещение user ID в URL API 'Create Post'



Ввод user ID в URL-адрес API 'Delete User'

Аналогично, URL API 'Post Comment' требует Post ID, полученный из ответа API 'Create Post'. Эти шаги могут быть утомительными и отнимать много времени, особенно при тестировании реальной системы, которая обычно включает в себя больше четырех API.

Как опытные тестировщики, мы также хотим проводить негативные тесты на каждом API в дополнение к интеграционному тестированию. Давайте рассмотрим несколько примеров негативных тест-кейсов для API 'Create User'.



```
{
  "name": "julotech",
  "gender": "male",
  "email": "postmanautomation@julotech.com",
  "status": "active"
}
```

Тело запроса API 'Create User' (JSON)

Number	Test Case
1	Check if "name" is empty
2	Check if "gender" is other than "male" or "female"
3	Check if "gender" is empty
4	Check if "email" is in invalid format
5	Check if "email" is already taken
6	Check if "email" is empty
7	Check if "status" is other than "active" or "inactive"
8	Check if "status" is empty

Примеры негативных тест-кейсов для API 'Create User'

Восемь — именно столько раз нам пришлось вручную нажимать кнопку "Отправить", не говоря уже о том, что нам пришлось вручную изменять тело запроса между нажатиями.

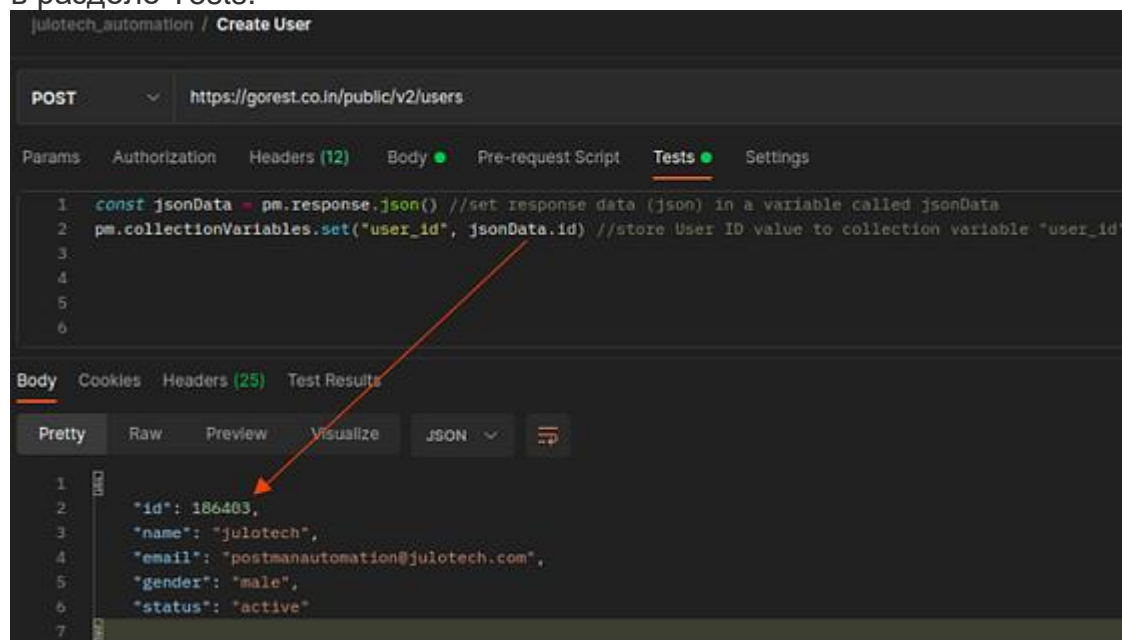
Давайте не будем больше медлить и начнем автоматизировать API. Я разделю процесс на две части: интеграция и управление данными.

## Автоматизация — интеграционный тест

### Шаг 1: Использование переменных

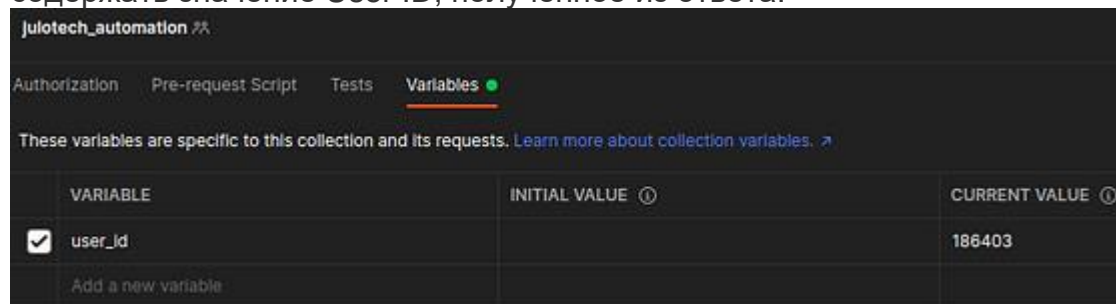
Мы можем упростить процесс, сохранив идентификатор пользователя из ответа API 'Create User' в переменной и затем используя его в URL API 'Create Post' и 'Delete User', вместо того, чтобы вручную копировать и вставлять его несколько раз.

Эту задачу можно решить, просто включив небольшой фрагмент кода на JavaScript в раздел Tests запроса. После отправки запроса Postman автоматически выполнит код в разделе Tests.



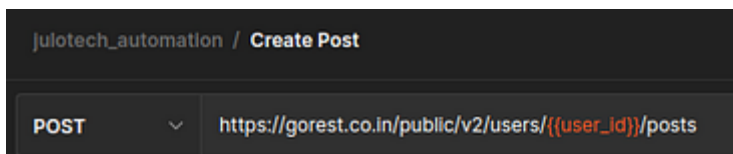
Сохраните User ID в переменной коллекции "user\_id"

Теперь, когда ID пользователя сохранен в переменной коллекции с именем "user\_id", давайте рассмотрим переменные коллекции. Переменная "user\_id" теперь должна содержать значение User ID, полученное из ответа.

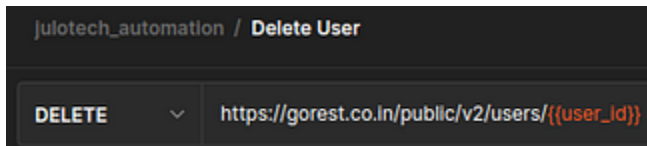


Переменные коллекции

Наконец, включите переменную в URL-адреса API 'Create Post' и 'Delete User'.

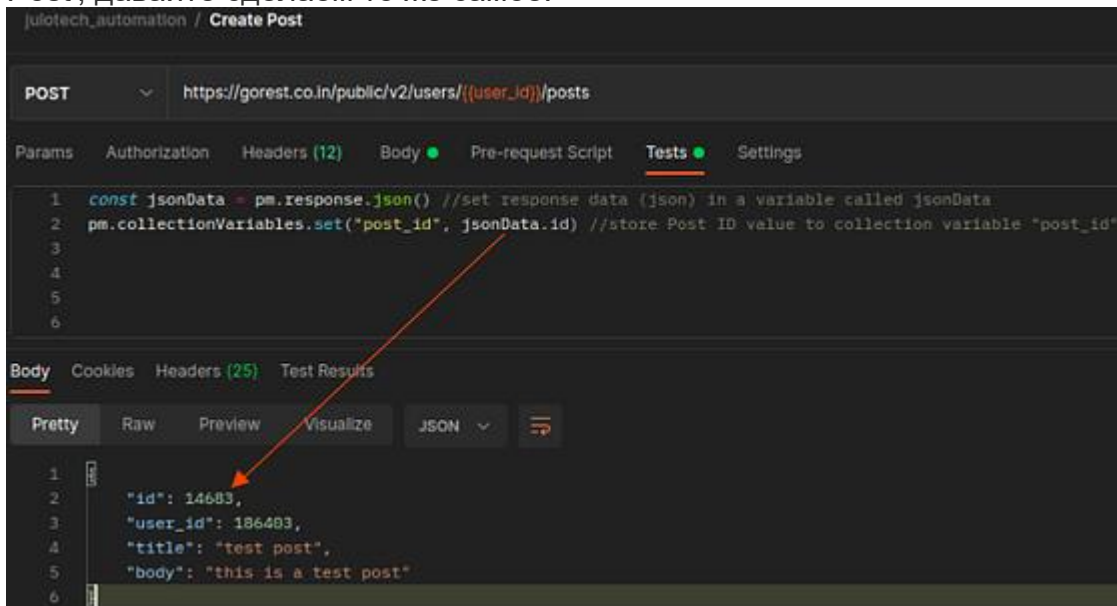


'Create Post' API URL

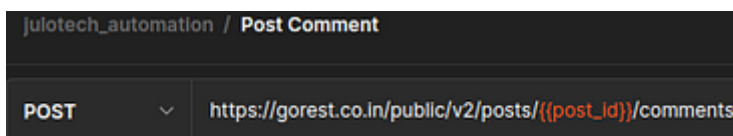


'Delete User' API URL

Поскольку URL API 'Post Comment' требует Post ID, полученный из ответа API 'Create Post', давайте сделаем то же самое.



Сохраните Post ID в переменной коллекции "post\_id"



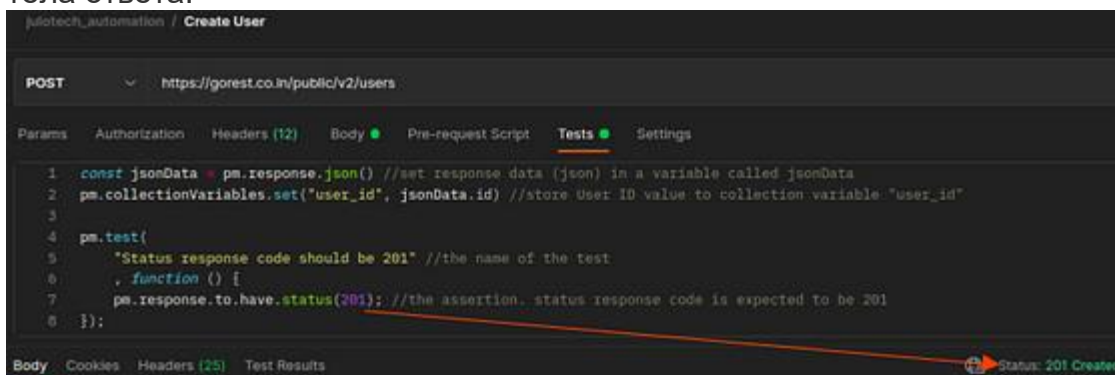
URL API 'Post Comment'

По завершении этого шага необходимость в ручном копировании и вставке данных между ответами API и URL будет устранена.

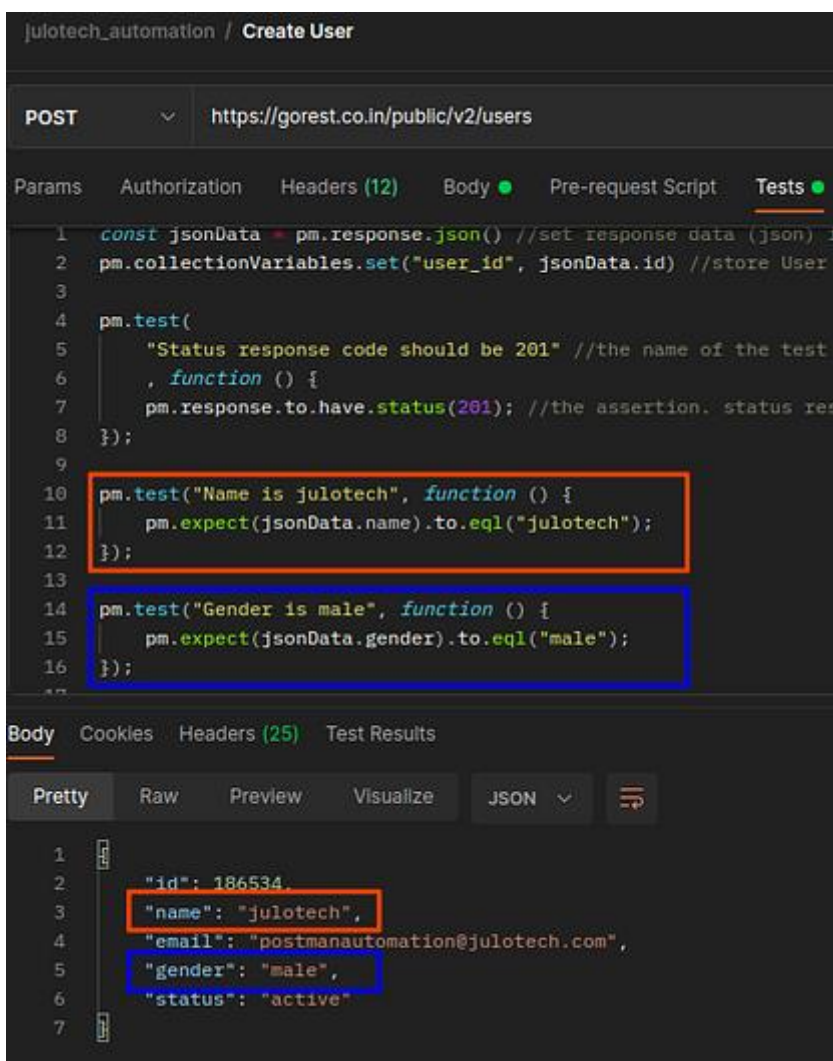
## Шаг 2: Создание утверждений (Assertions)

Утверждения являются важным компонентом автоматизации, поскольку они позволяют определить успех или неудачу теста.

Аналогичным образом, утверждения могут быть включены в раздел Tests запроса. Поскольку в одной статье невозможно охватить все возможные типы утверждений, мы сосредоточимся на процессе добавления утверждений для код-статуса ответа и тела ответа.

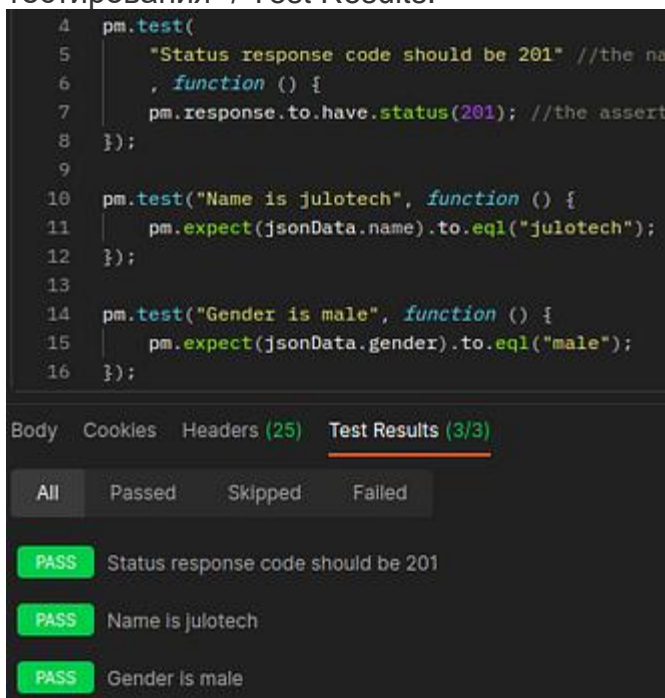


Утверждение кода ответа состояния



Утверждение тела ответа

Давайте сделаем проверку: отправим запрос и посмотрим на раздел "Результаты тестирования" / Test Results.

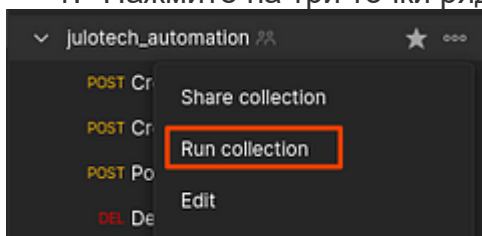


Отлично! Теперь, когда мы применили тот же процесс к остальным API (которые не будут рассматриваться в этой статье), давайте перейдем к последнему шагу.

### Шаг 3: Использование Postman Collection Runner

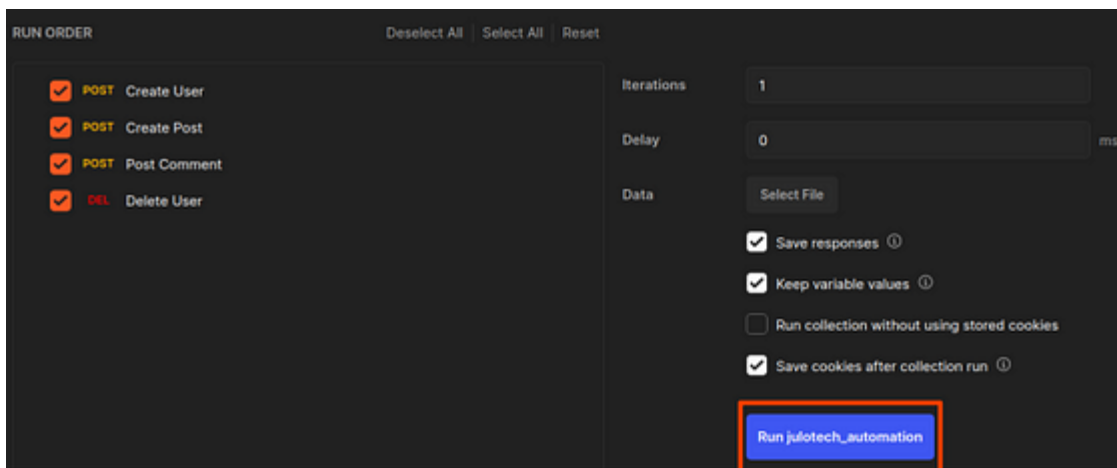
Хотя мы выполнили шаги 1 и 2, нам все еще нужно вручную нажать кнопку «Отправить» для каждого API. [Postman Collection Runner](#) автоматизирует этот процесс, запуская все API в коллекции. Помните, что он запускает их последовательно, поэтому перед запуском runner-а убедитесь, что порядок API в коллекции правильный.

1. Нажмите на три точки рядом с названием коллекции и выберите Run collection.

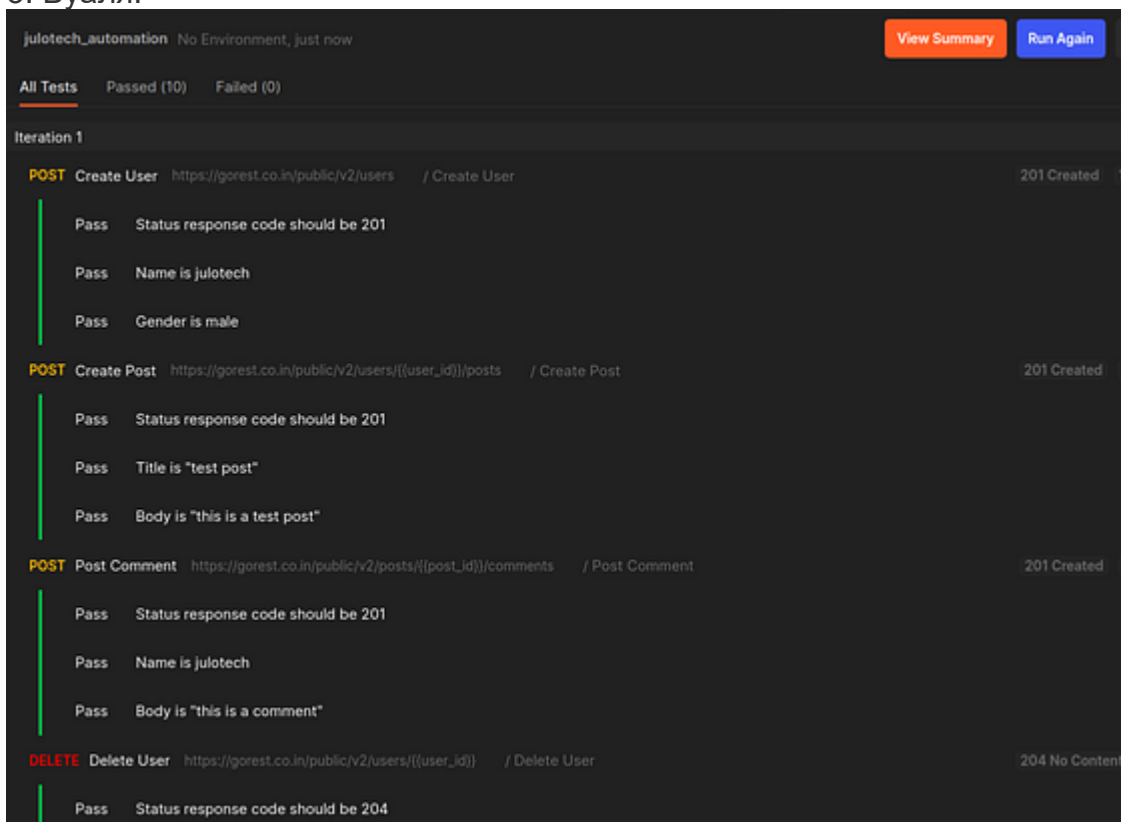


2. Нажмите кнопку "Run <имя коллекции>".





### 3. Вуаля!



Теперь интеграционный тест автоматизирован, что позволяет просто и без особых усилий выполнять его в любое время.

## Автоматизация — тест, основанный на данных

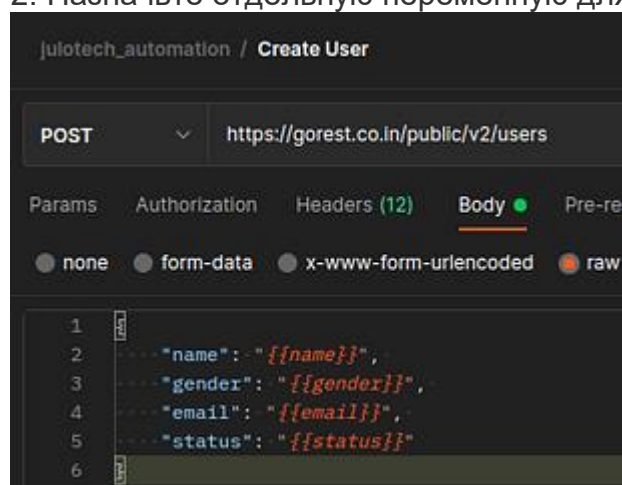
### Шаг 1: Настройка запросов

Цель этого процесса — создать наборы тестовых данных в CSV-файле, сохранить их в переменной коллекции, а затем отправить запрос определенное количество раз, исходя из количества строк (как тест-кейсов) в файле.

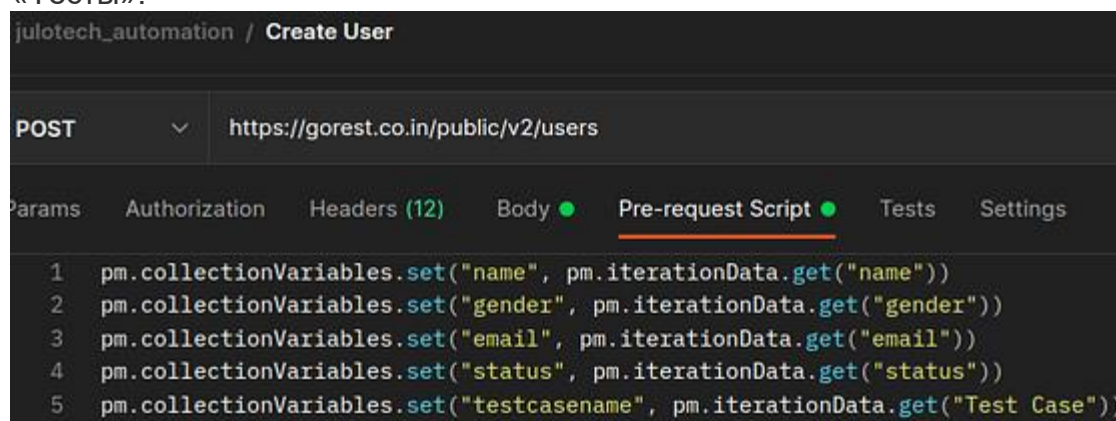
### 1. Создайте CSV-файл и введите необходимые тестовые данные

Number	Test Case	name	gender	email	status
1	Check if "name" is empty		male	postmanautomation@julotech.com	active
2	Check if "gender" is other than "male" or "female"	julotech	asd	postmanautomation@julotech.com	active
3	Check if "gender" is empty	julotech		postmanautomation@julotech.com	active
4	Check if "email" is in invalid format	julotech	male	postmanautomationjulotech.com	active
5	Check if "email" is already taken	julotech	female	postmanautomation123@julotech.com	active
6	Check if "email" is empty	julotech	male		active
7	Check if "status" is other than "active" or "inactive"	julotech	male	postmanautomation@julotech.com	asd
8	Check if "status" is empty	julotech	female	postmanautomation@julotech.com	

### 2. Назначьте отдельную переменную для значения каждого атрибута



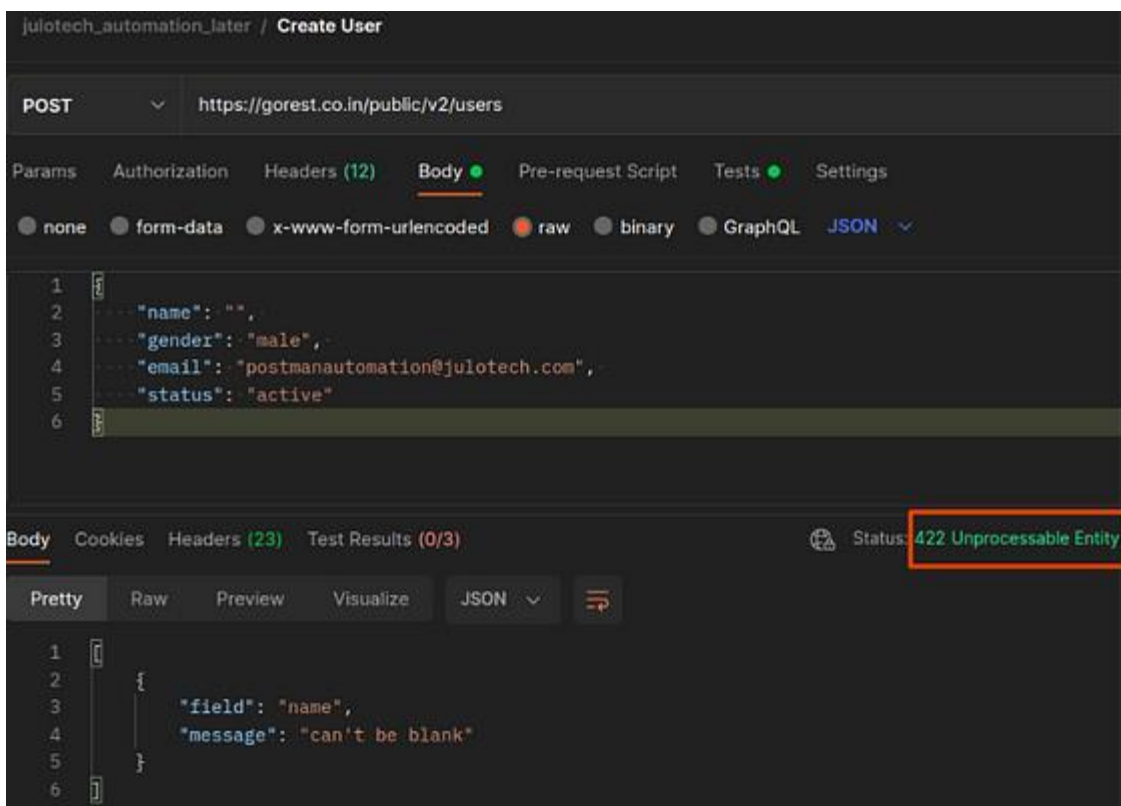
3. В разделе Pre-request Script добавьте код, чтобы загрузить данные теста из CSV-файла и сохранить их в переменных коллекции. Обратите внимание, что Postman автоматически выполнит этот код перед отправкой запроса, в отличие от раздела «Тесты».



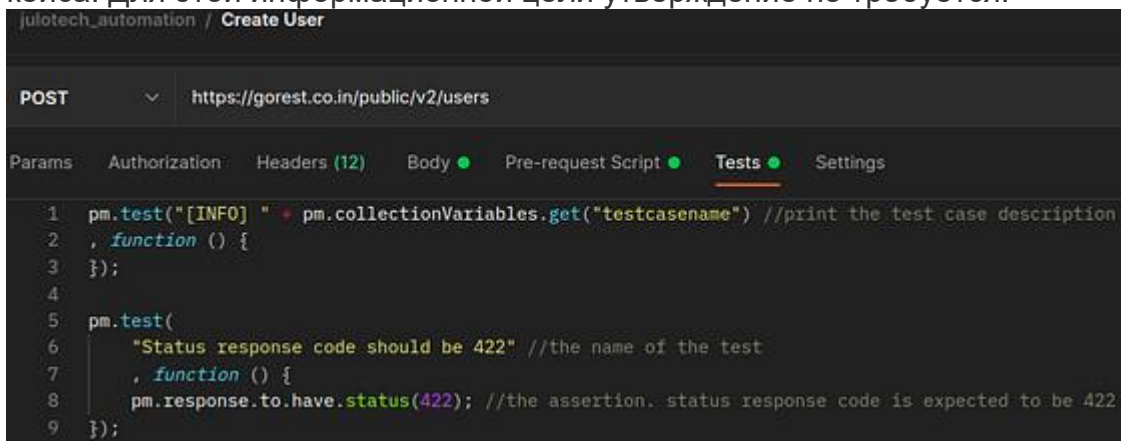
## Шаг 2: Создание утверждения

На рисунке показан результат тест-кейса «проверить, пусто ли имя».



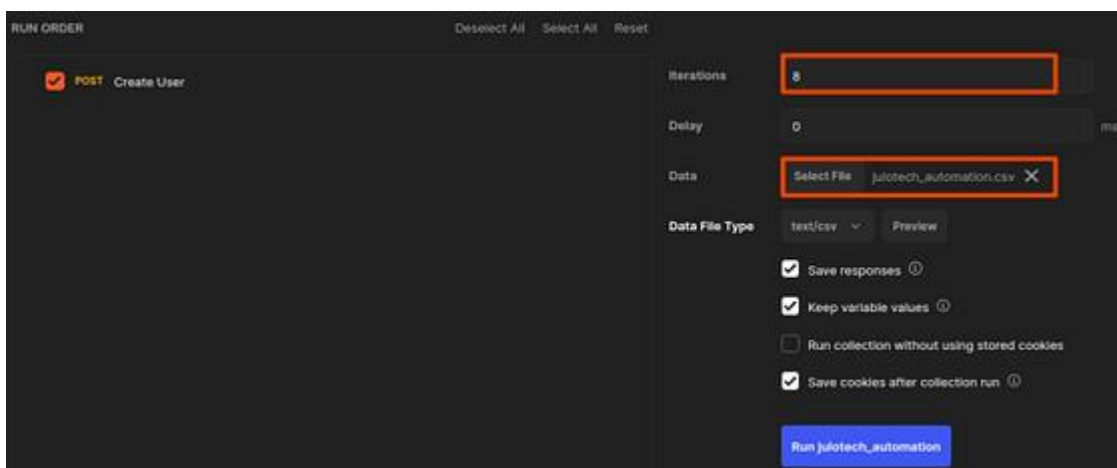


Поскольку все тест-кейсы будут возвращать ответ со статус-кодом 422, будет целесообразным определить это значение в качестве ожидаемого ответа. Кроме того, нам понадобится еще один «тест» для отображения описания каждого тест-кейса. Для этой информационной цели утверждение не требуется.

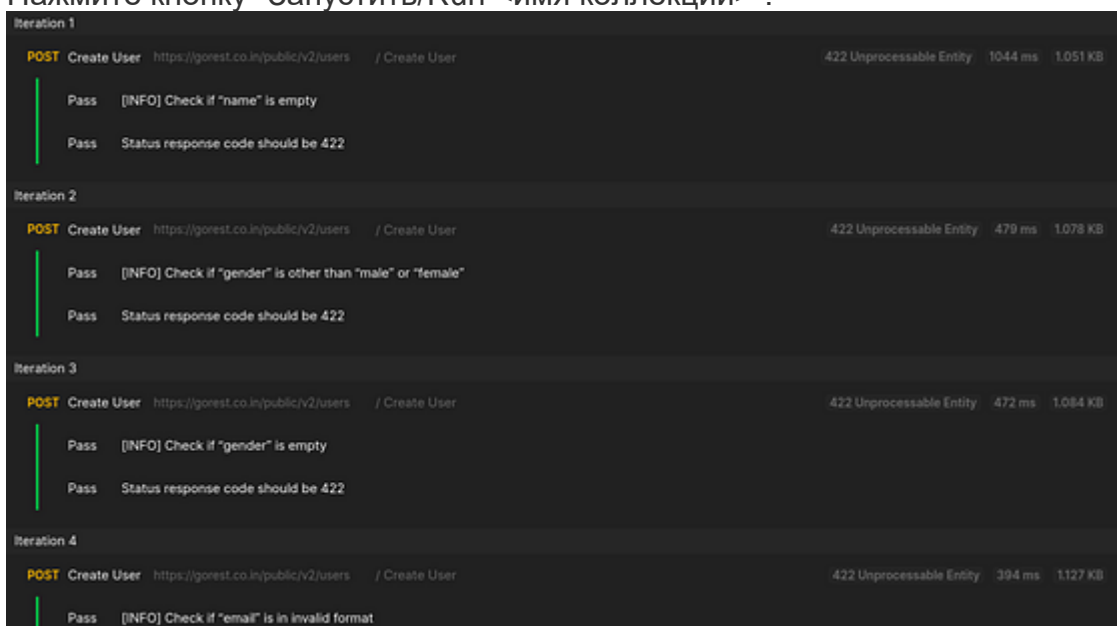


### Шаг 3: Загрузите CSV-файл и запустите коллекцию

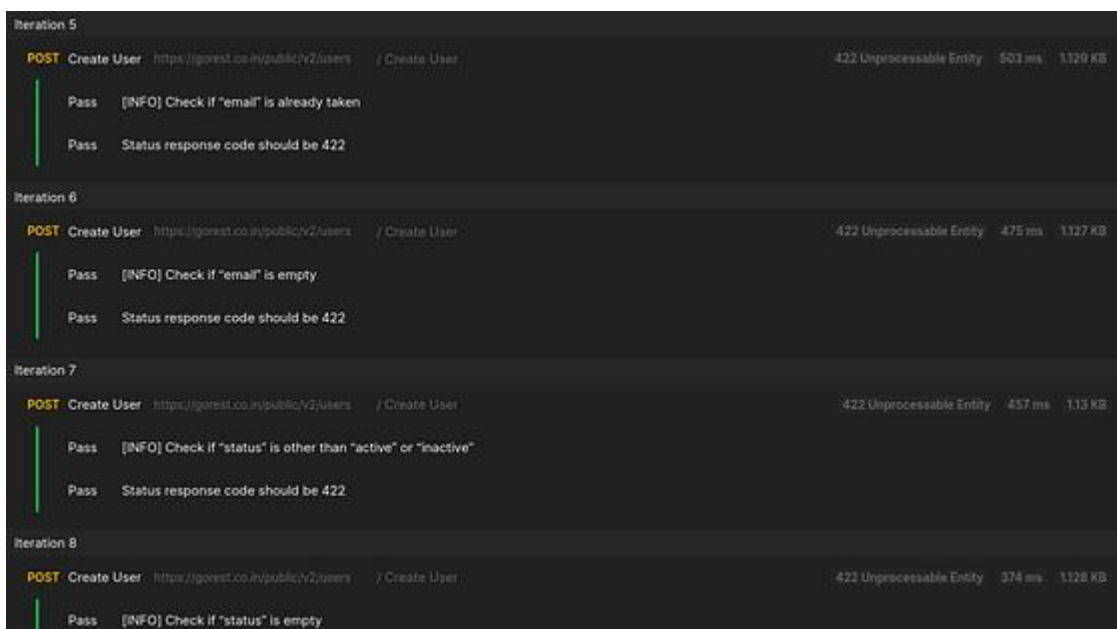
Перед запуском коллекции загрузите CSV-файл, содержащий тестовые данные. Postman автоматически определит количество итераций.



Нажмите кнопку "Запустить/Run <имя коллекции>".

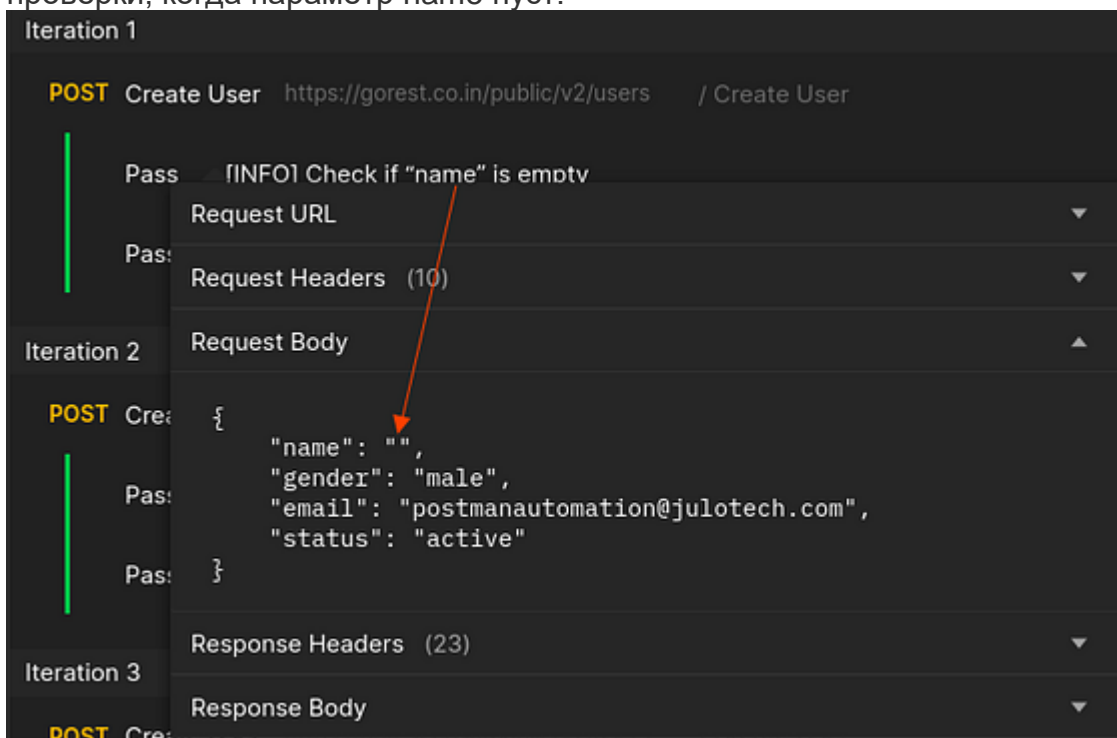


Результаты выполнения тест-кейсов с 1 по 4



Результаты выполнения тест-кейсов с 5 по 8

Из результатов видно, что все утверждения прошли. Мы можем для проверки углубиться в детали каждого утверждения, как в приведенном ниже примере проверки, когда параметр name пуст.



Тест-кейс 1

Отлично! Мы видим, что все отправленные запросы соответствуют тестовым данным из CSV-файла. Автоматизированный Data Driven тест завершен!