**For what do I need Postman?** Postman can connect to the server through the API, and exchange the data. Postman allows to work with API through an interface, no need to write code or open terminal

**Две версии postman-a**: 1)- web и 2)-app. Если работать в web то с низу с права будет вкладка - Auto-select agent.

По умолчанию активна ф-ция auto-select, отключаем. У нас есть 3 варианта

1)- Brawser agent - озночает что запросы будут отсылатся на API с помощью браузера, и самая главная идея в том, что тут надо понимать, что браузер в принципе не соответствует своим функционалом для этой работы, у него есть определённые ограничения и он в принципе не заточен для работы с большинством API. Это в принципе и является причиной почему у нас в web версии постмана есть 3 разных варианта (brawser agent-один из них).

2)- Cloud agent - фактически использует специальные (secured) серверы самого постмана для запросов. Т.е. postman web отсылает запрос на серверы постмана, а сервер отправляет запрос на API, откуда мы получаем респонс. Cloud agent имеет также лимит в запросах

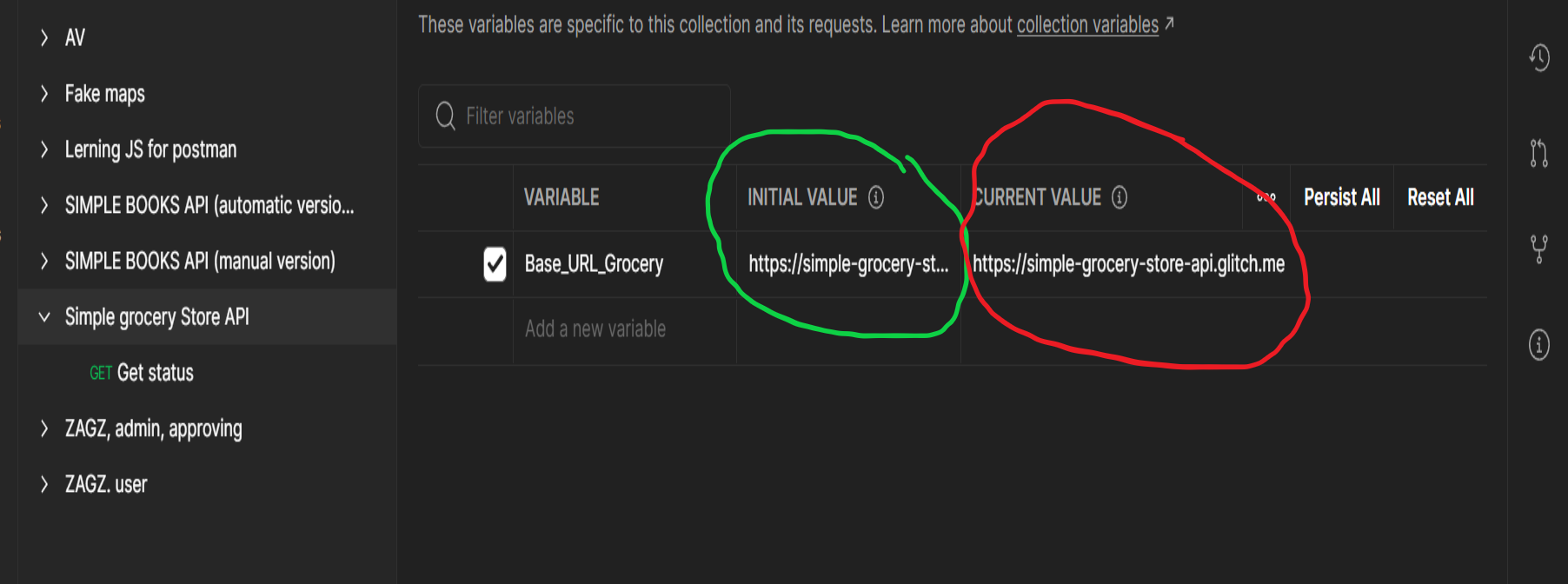
3)- Desctop agent - предполагает загрузку дополнительного софта для выполнения запросов

**Collection** - it's like a folder that stores multiple requests. We typically create collections` for the API or for specific case that we have. And it's not possible to save some request without specifying a collection.

**Variables-** всегда пишутся в квадратных скобках {{ }}, всё как в джава, записываем какое то значение (configuration) в переменную (что-то повторяющееся каждый риквест например URL, без END-point-a) а потом когда нам надо использовать опять это значение подставляем переменную. Это удобно если например надо сделать 20 запросов, или мы сделали 20 запросов, а потом ,что то поменялось и теперь надо отредактировать все эти запросы. В случае с переменной нам надо отредактировать только в одном месте-т.е. значение этой переменной.

**Collection veriable** - создаётся в самой коллекции

Если посмотреть в консоли то видно, что на сервер “ушол” запрос корректный, без переменной, но со значением которое было в переменной. POSTMAN использует это внутри для своих нужд,, а на сервер передаёт ТОЛЬКО значения этих переменных.

Переменные можно устанавливать как бы локальные, относящиеся к определенной коллекции и соответственно “видимые” только внутри этой коллекции. Если попытаться воспользоваться такой переменной в другой коллекции будет ошибка.

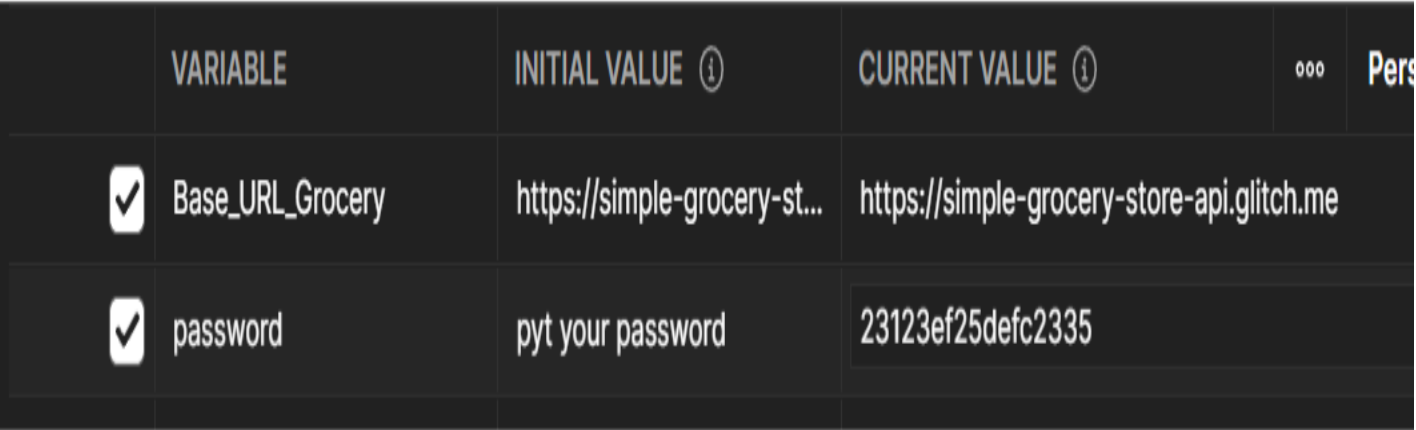
Переменные можно редактировать, для этого открой саму коллекцию, у каждой переменной есть два значения INITIAL и CURRENT. Этот функционал добавлен для работы в команде.

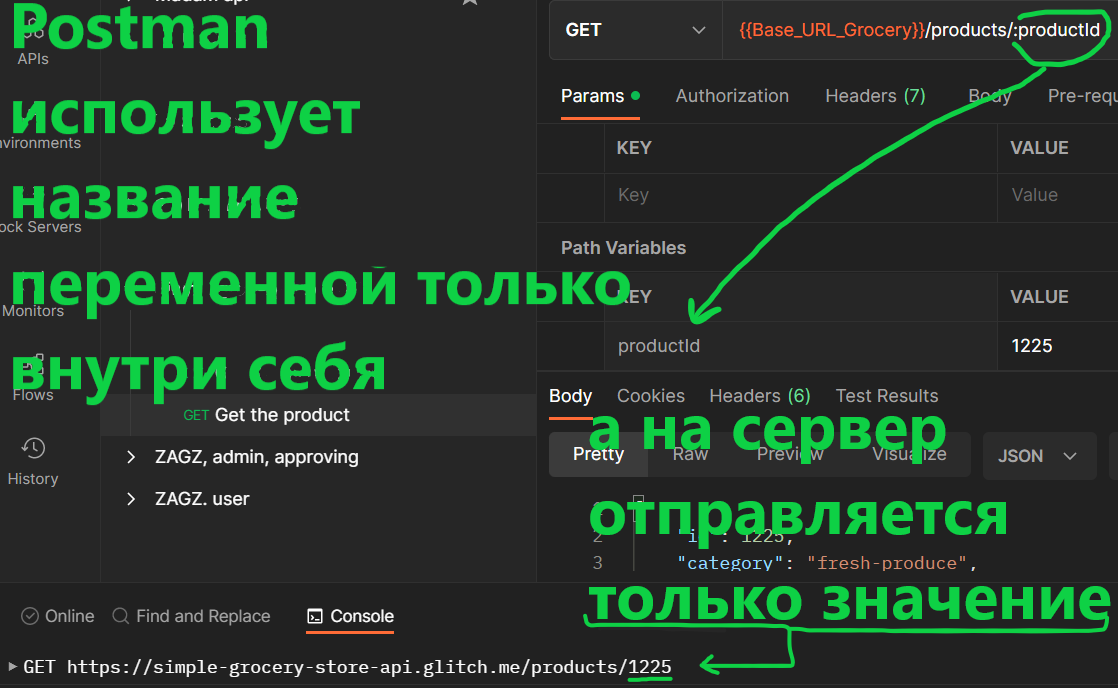
INITIAL VALUE- здесь указанно значение данной переменной, и это значение видно для всех пользователей кто работает с данной коллекцией в постмане.

CURRENT VALUE- это значение является основным с которым работает данный пользователь,т.е. я, и оно видно только мне

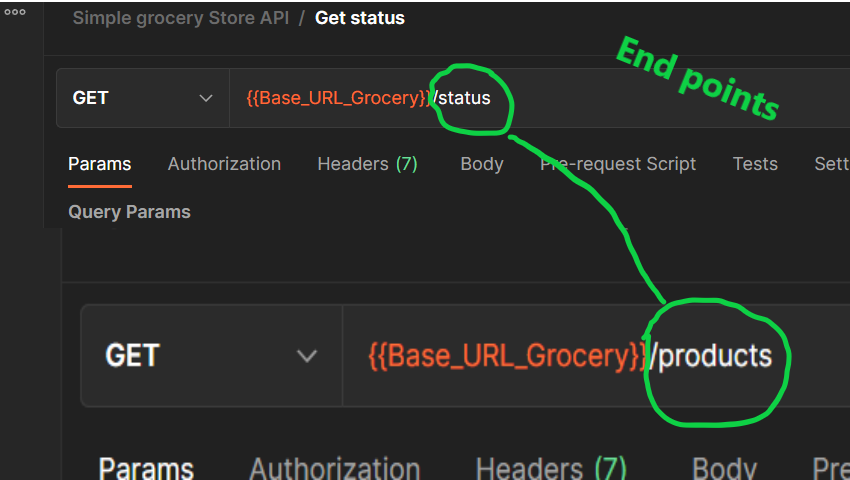
Это может быть удобно при работе с какими то паролями, в этом случае при использовании данной переменной можно будет защитить свои персональные данные. Смотри пример ниже.

Another benefit of using variables in Postman is that it allows you to store sensitive information, such as passwords or API keys, in a secure and centralized location. The only way to delete a Postman collection variable is by editing the collection itself.



**Path variable Path variables in an REST API endpoint are used to specify a unique identifier for a resource.**- эти переменные, судя из названия указываются в строке ввода после двоеточия. В POSTMAN во вкладке params автоматически создаётся отдельная графа “Path Variables” куда также автоматически (как правило когда копи-пэйстиш endpoint то там уже есть название path variable и когда подставляешь в строку ввода то название переменной копируется) вводится название path переменной, остаётся только ввести её значение (обязательно валидное). Значение как правило берутся из предыдущих риквестов. Функционал похож на Query params, тоже по сути своей path variable определяют какие то фильтры реквеста, тем самым дополняя имеющиеся Query params. А почему так называется? Path variable называются так потому что они отправляются на сервер в строке риквеста, если открыть консоль то увидим, что на сервер “ушол” риквест с endpoint-ом БЕЗ названия нашей path variable, а ТОЛЬКО значение!!! То есть значение переменной было подставлено в путь (path). Если в такой переменной не указать значение то получим ошибку, так как Postman подставит вместо значения, название переменной (placeholder), в свою очередь сервер получив такой реквест будет рассматривать полученное название как значение и выдаст ошибку, что-то типа 400Bad Request , “The product must be a number”.

**!!!! Path variables - are mandatory !!!!! They are part of the path, you can't remove them and get the same functionality**



**End point** - (technically speaking) the end of communication channel.))) Нармальными словами это что то вроде окончаний в URL которые (при одном и том же baseurl) обращаясь к одной и той же API, дают нам разные респонсы.

An endpoint is a specific URL that serves as the entry point for interacting with a particular resource (for example, one specific product) or set of resources (for example, a list of products). Endpoints are typically defined for each resource that the API exposes. In other words, each resource will have its own uniquely identifiable address.

In a REST API - we are getting resorsess, к примеру мы используем метод GET для получения List of product в какой то API. В этом случае то, что нам вернётся в респонсе это ресурсы, поэтому URL (uniform resource locator) так и называется, т.е. это локатор ресурсов

**shortcut** Чтобы просмотреть доступные быстрые клавиши нужно перейти по “settings” - “shortcuts”.

ctrl+s - сохранить риквест (вместо того чтобы всегда нажимать “save”)

ctrl+enter - выполнить риквест.

ctrl+t - новая вкладка

**Beautify** - когда мы в body запроса пишем / вставляем какой то JSON, то благодаря редактору Postman-a, нажав на эту клавишу справа от поля ввода, редактор сам всё отредактирует и делает код читабельным

**Authentication**

When we are working with API and, for example, creating data or updating an existing one? we need to provide some kind of authentication. Some API needs to know who is creating data, and, in order to tell API who we are, we need to register. The purpose of registration is to obtain an access token (it is a temporary password). Once we have got access token the easiest way to use it, is to go to Autorisation, and there is a list of different kinds of autorisation types. Надо выбрать тот который требуется API и внести значение полученное при регистрации или если это значение записано в переменную внести название переменной и Postman далее сам подставит значение. Реквесты где требуются авторизационные токены чаще всего предусматривают наличие информации в body.

Теперь постараемся разобраться что на самом деле произошло. Postman это всего лишь UI toll, интерфейс, и все эти вкладки типа authorization существуют исключительно для удобства. Когда мы внесли запись токена, а затем отправили риквест, на самом деле в реквест был добавлен header. Postmann added header for us.

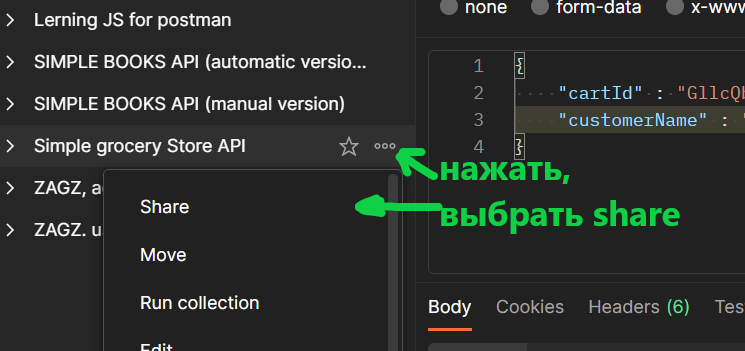
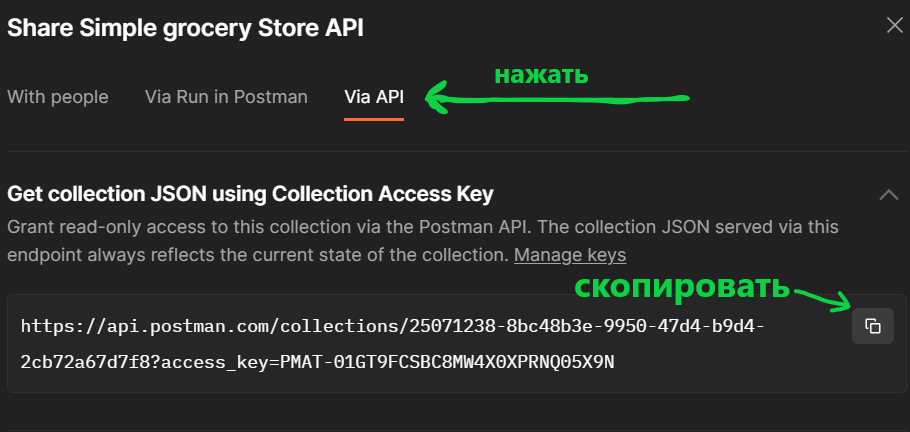
**Random data in requests**

Для того чтобы воспользоваться переменными с случайными значениями подставленными постманом надо воспользоваться синтаксисом “{{$ и дальше выбрать из предложенного}}”.

**Headers**

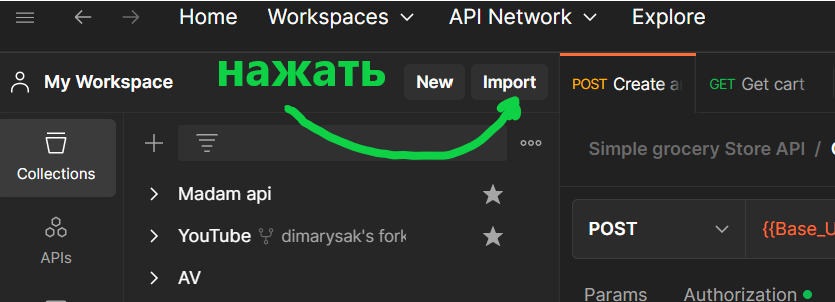
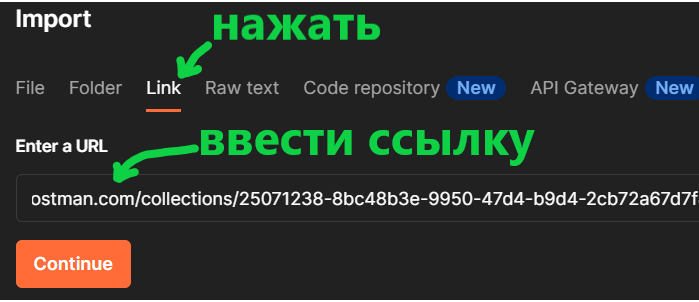
По большому счёту это пары key : value, подставляются постманом автоматически, служат как бы говоря серверу какую инф. в себе несут и как обрабатывать тело запроса. Также headers есть и в респонсе, гдеони в свою очередь говорят какая инф. находится в теле и как с ней работать/отображать.

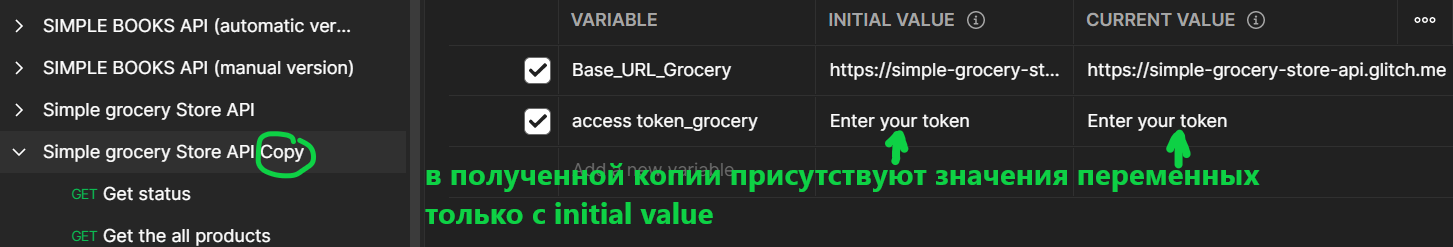
**Exporting and Importing a Postman collection**

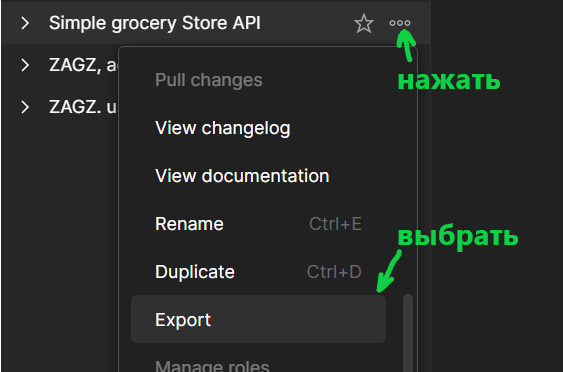
Там ещё надо нажать сгенерировать ключ, прежде чем копировать ссылку! Затем можно поделится ссылкой.

Для того, чтобы загрузить ссылку надо:

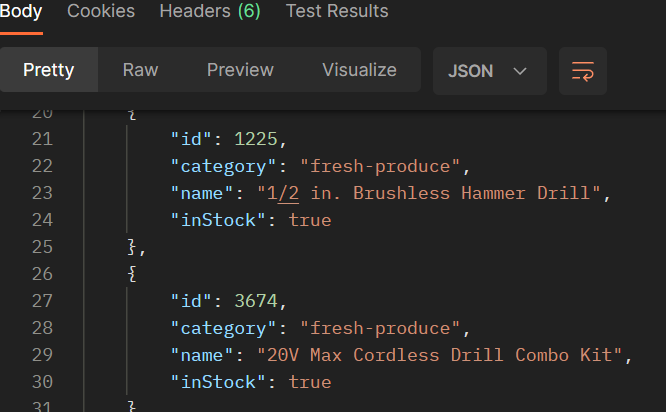


Если при создании переменных “зряча” указывать initial value (то что будет видно если мы расшарим коллекцию) и current value (значение которое останется только у нас в системе) то в копии мы не “засветим” значения своих личных данных.

Также можно export данную коллекцию себе на комп в виде JSON file, т.е. сохранить данный файл, а затем также поделится с кем то или оставить себе ). 

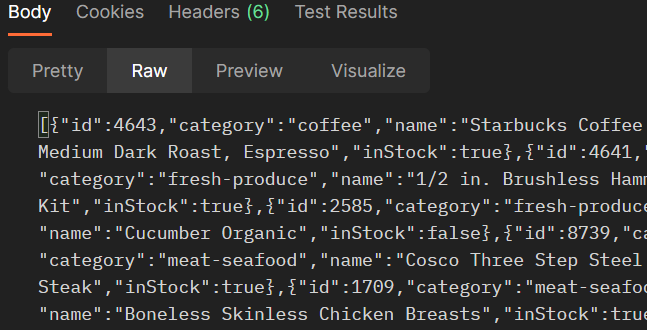
Если получил такой файл то загрузить в постман можно через вкладку **import** / **File** и далее всё тоже как и при “шаринге”).

**Visualizing responses in Postman**

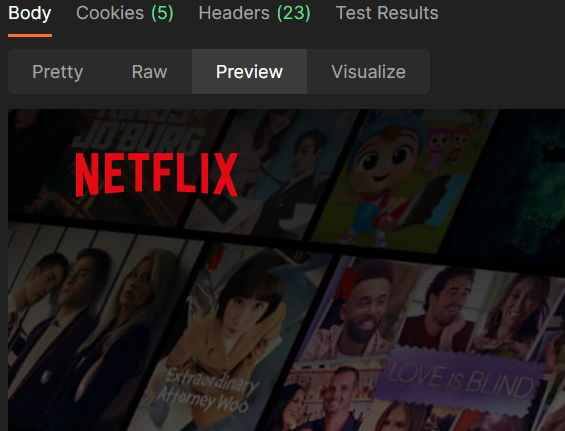


Респонс приходит на POSTMAN и POSTMAN даёт нам на выбор как мы хотим “видеть” этот респонс. В частности вкладка Pretty показывает всё “красиво” в разных цветах названия переменных и их значения. Возможность свернуть выбранные объекты ипр.

Но это не то как эти данные выглядят на самом деле. Это просто ф-ция POSTMAN-a.

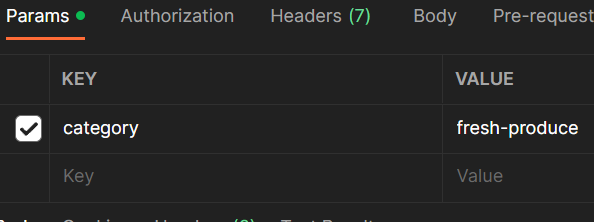


Вкладка Raw это то, как на самом деле выглядит JSON, который мы получаем в респонсе, поэтому и вкладка Raw. Raw JSON. Сырой JSON, т.е. необработанный.



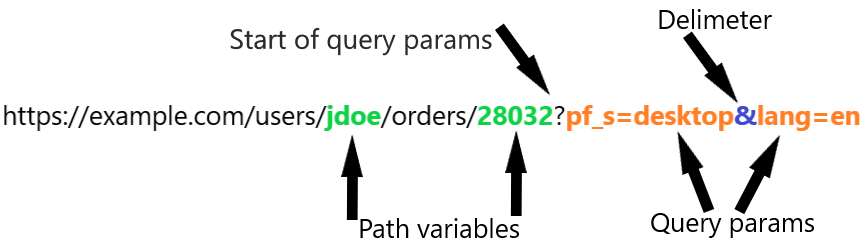
Preview - имеет значение когда мы работаем с другим форматом данных, не с JSON. Например если зделать запрос на какойто сайт например netflix.com, который имеет HTML, то вкладка Pretty- постарается отразить всё красиво несмотря на то, что это HTML, а не JSON. Во вкладке RAW будет совсем неразбериха, хотя постман сделает всё возможное. А вот вкладка Preview отображает контент максимально как браузер (но с меньшими возможностями, POSTMAN инструмент для работы с API в первую очередь)

**Query** **Parameters**

Речь идёт о Query params.

Parameters - с параметрами всегда работают с документацией, заранее нельзя предусмотреть какие параметры валидные (доступные), а какие нет, это полностью зависит от разработчика API, также как и с значениями того или иного параметра. В Postman-е для этого существует специальный UI с вкладкой Params, где можно редактировать параметры запроса и делать их активными или на время отключать при выполнении запроса, здесь же редактируются и path variables.

Основное назначение **Query** **Parameters** это фильтрация получаемых данных.

в поле KEY ввести название параметра, в поле VALUE - соответственно значение, а галочкой слева можно активировать/деактивировать участие данного параметра в запросе.

**“?” “=”**  **“&”**

Если посмотреть в строку ввода то при этом после End point-a в URL ставится **“?”** знак вопроса и пишется название параметра затем

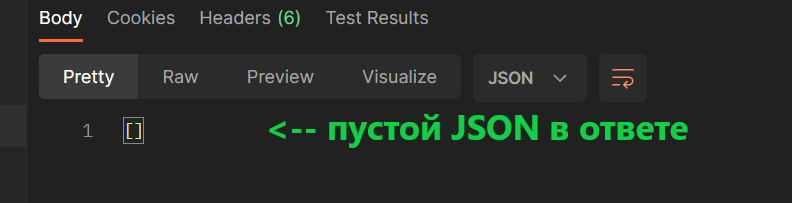
**“=”** знак равно и указывается значение параметра. Всё, что идёт до знака вопроса НЕ МОЖЕТ быть query params. Если указывается несколько query parameters то они перечисляются через амперсанд **“&”**

Пример: {{Base\_URL\_Grocery}}/products??category=fresh-produce**&**results=5

В вкладке Params это делать намного удобней, не надо набирать вручную в строке поиска и помнить про синтаксис, также активизировать/деактивизировать при необходимости проще.

К примеру у нас есть риквест с определенным набором Query Parametersи мы хотим создав новый риквест перенести все, или часть данных параметров в наш новый реквест, можно сделать это быстро, а не таскать по очереди, для этого справа в “шапке” таблицы для редактирования есть вкладка “Bulk edit” нажав на неё можно скопировать нужные пары (ключ значение), зайдя в нужный риквест обратно-же нажать на Bulk edit и вставить скопированное ранее. Затем нажать Key-value edit.

**Важно!** При написании имени параметра необходимо соблюдать табуляцию (заглавные/прописные буквы), так как данное поле является key sensitive. category и Сategory это с точки зрения Postman-a два разных параметра. Тоже самое касается и значений параметров, они должны быть написаны идентично как в документации. Most APIs are sensitive about how query parameters are written. So generally-speaking, query parameters are case-sensitive. Some APIs may handle query parameters as case-insensitive because it is more convenient and less error-prone. But this is not a rule. You should always check the API documentation to see how the query parameters are written.

Как особенность применения параметров, в респонсе можно получить пустой ответ, при этом название параметра, значение, могут быть корректными и статус код будет (200 ok), дело в том что мы получили пустой JSON, такое бывает. И это тоже важная информация говорящая, что в этой категории товаров нет, но технически всё работает исправно.

This is a representation of an empty list of data objects in JSON. This is not considered an error and the status code should be 200 OK.

“Подгружать” параметры можно только те которые указаны в документации. К примеру я получил какой то общий список продуктов и хочу отфильтровать. Я вижу, что у каждого объекта JSON-а есть поля, но это ошибка думать что по этим полям(по любому из полей) можно сделать фильтрацию и получить какую то специфическую группу продуктов (объектов), если разработчиком не предусмотрено такое поведение, то это Error.

Также если пытатся к примеру получить лист продуктов с большим/меньшим числом чем то что заявлено в документации.

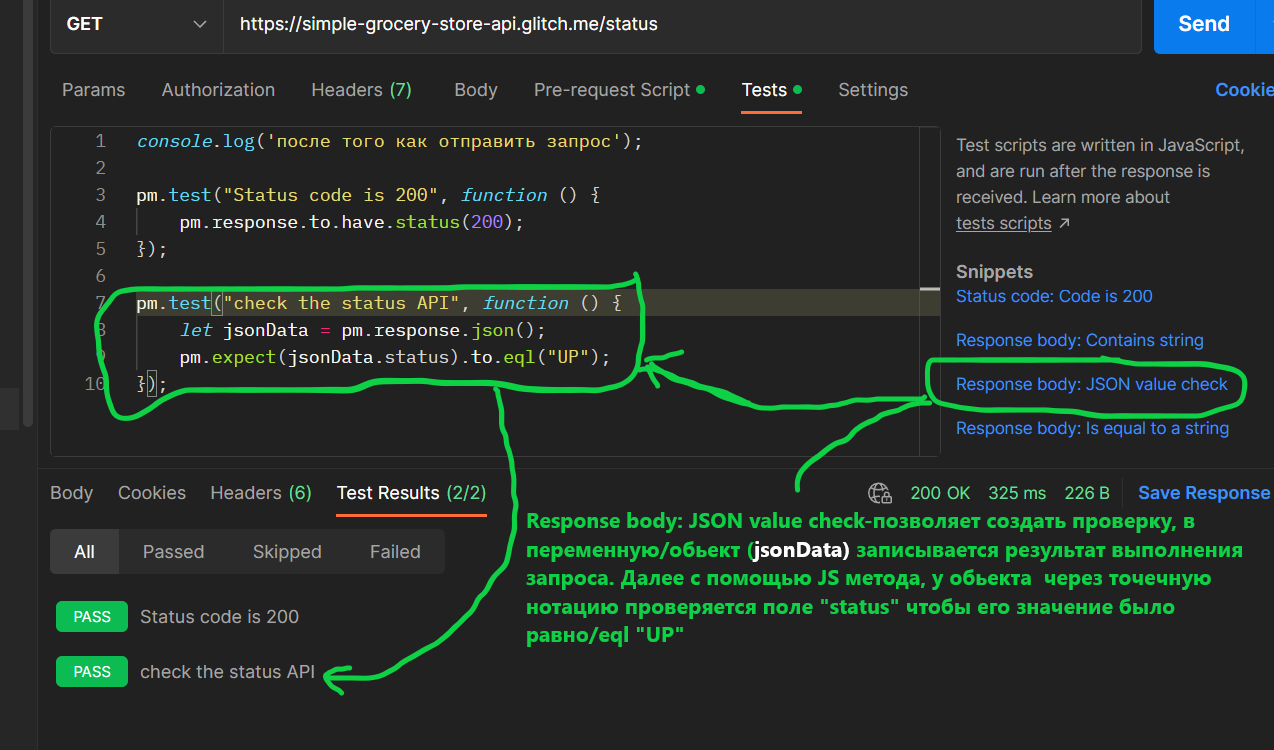
Также если API не предусмотрено получение в одном запросе сразу нескольких значений одного параметра (пытаемся получить один список с элементами у которых значение поля “category” равно dairy или eggs: products?category=dairy&category=eggs) то ничего не будет, поведение API нельзя запрограммировать, это делает разработчик, мы проверяем работает ли оно следуя документации. Every API is designed and built differently. This is why it is essential to study the API documentation and to understand what is the intended usage of the API. Each API works differently. Just by looking at the response body, we can't tell which parameters does the API accept.

Query parameters are present only in the request. The response typically is an object with properties. You may find query parameters that have the same name as a property from the body, but this is not mandatory.

**!!!!!!!!!** Работая с значениями параметров стоит отметить, что если в документации описан к примеру тип значения int, но при этом мы вводим символы(char) или String+int или float(2.5) и получаем от сервера какой то внятный респонс со статусом кода 200 то это является основанием чтобы задать вопрос разработчику о корректности такого поведения (ведь это баг). Так же к этому относится когда сервер возвращает неправильное количество элементов, к примеру по документации в респонсе количество продуктов может быть запрошено 1-20, а мы запрашиваем 0, и получаем 20, или запрашиваем 21, а получаем 20, это всё потенциальные баги.

**Pre-request-script** - вкладка где можно писать JS который **будет выполнен до того как риквест будет отправлен на сервер.**

**Tests** - вкладка где можно писать JS который **будет выполнен после того как риквест выполнится.** The point of the tests to automate the way we check if a specific endpoint is working? but instead of verifying that with our own eyes, we write the test and let Postman do it for us.

****

**ENVIRONMENT**

Обеспечивает набор переменных видимых для одной взятой коллекции и для определённых целей. В рамках одной коллекции можно иметь сколько хочешь таких окружений, каждая из которых будет иметь свой уникальный сет переменных. но одновременно к коллекции можно применять только одно окружение. Положительно отражается на уменьшении глобальных переменных .

выв