**Евразийский национальный университет имени Л.Н.Гумилева**

**Отчет**

**по Лабораторной работе №9**

|  |  |
| --- | --- |
|  | Выполнила:  Жайсанова Дания ИС-32  Проверила:  Жукабаева Т.К. |

**г. Нур-Султан**

**2022**

**Лабораторная работа №****9**

**Тема:** Api testing

**API**— это Application Programming Interface, или программный интерфейс приложения, с помощью которого одна программа может взаимодействовать с другой. API позволяет слать информацию напрямую из одной программы в другую, минуя интерфейс взаимодействия с пользователем.

API работает по такому же принципу — принимает ваш запрос, передаёт информацию системе, обрабатывает её и возвращает ответ.

API может быть внутренним, частным — когда программные компоненты связаны между собой и используются внутри системы. А может быть открытым, публичным — в таком случае он позволяет внешним пользователям или другим программам получать информацию, которую можно интегрировать в свои приложения.

Чтобы программам общаться между собой, их API нужно построить по единому стандарту. Одним из них является REST — стандарт архитектуры взаимодействия приложений и сайтов, использующий протокол HTTP. Особенность REST в том, что сервер не запоминает состояние пользователя между запросами. Иными словами, идентификация пользователя (авторизационный токен) и все параметры выполнения операции передаются в каждом запросе. Этот подход настолько прост и удобен, что почти вытеснил все другие.

Тестирование API проводят, основываясь на бизнес-логике программного продукта. Тестирование API относится к интеграционному тестированию, а значит в ходе него можно отловить ошибки взаимодействия между модулями системы или между системами. Для тестирования используют специальные инструменты, где можно отправить входные данные в запросе и проверить точность выходных данных. Одним из таких инструментов как раз и является Postman. Вот что он умеет:

* Составлять и отправлять запросы;
* Сохранять запросы в папки и коллекции;
* Параметризовать запросы;
* Добавлять к вызову API контрольные точки;
* Создавать разные окружения для одних и тех же запросов;
* Запускать коллекции с помощью Collection Runner и использовать их как автоматизированные тесты.

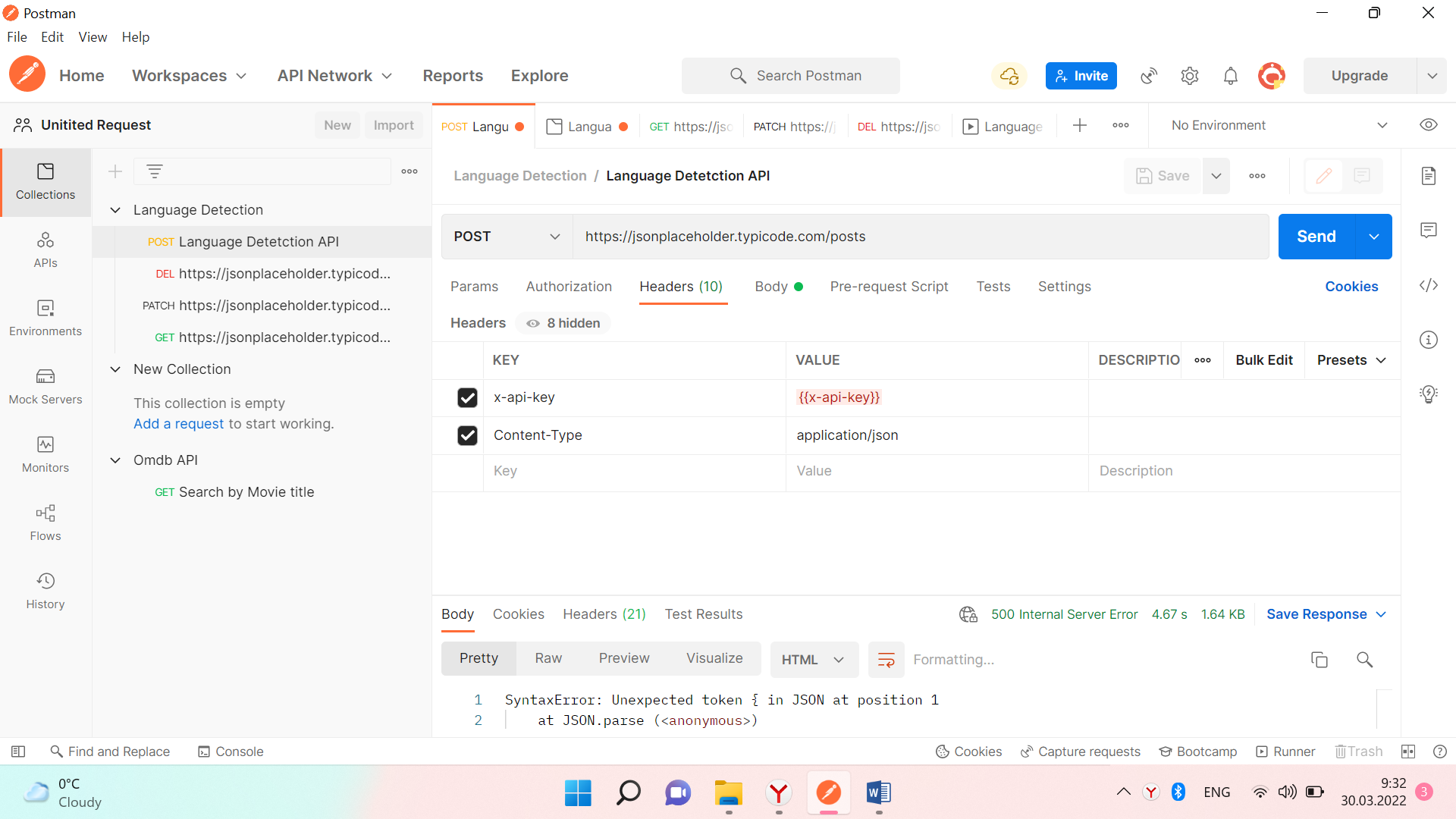
**Для чего нужен Postman?**

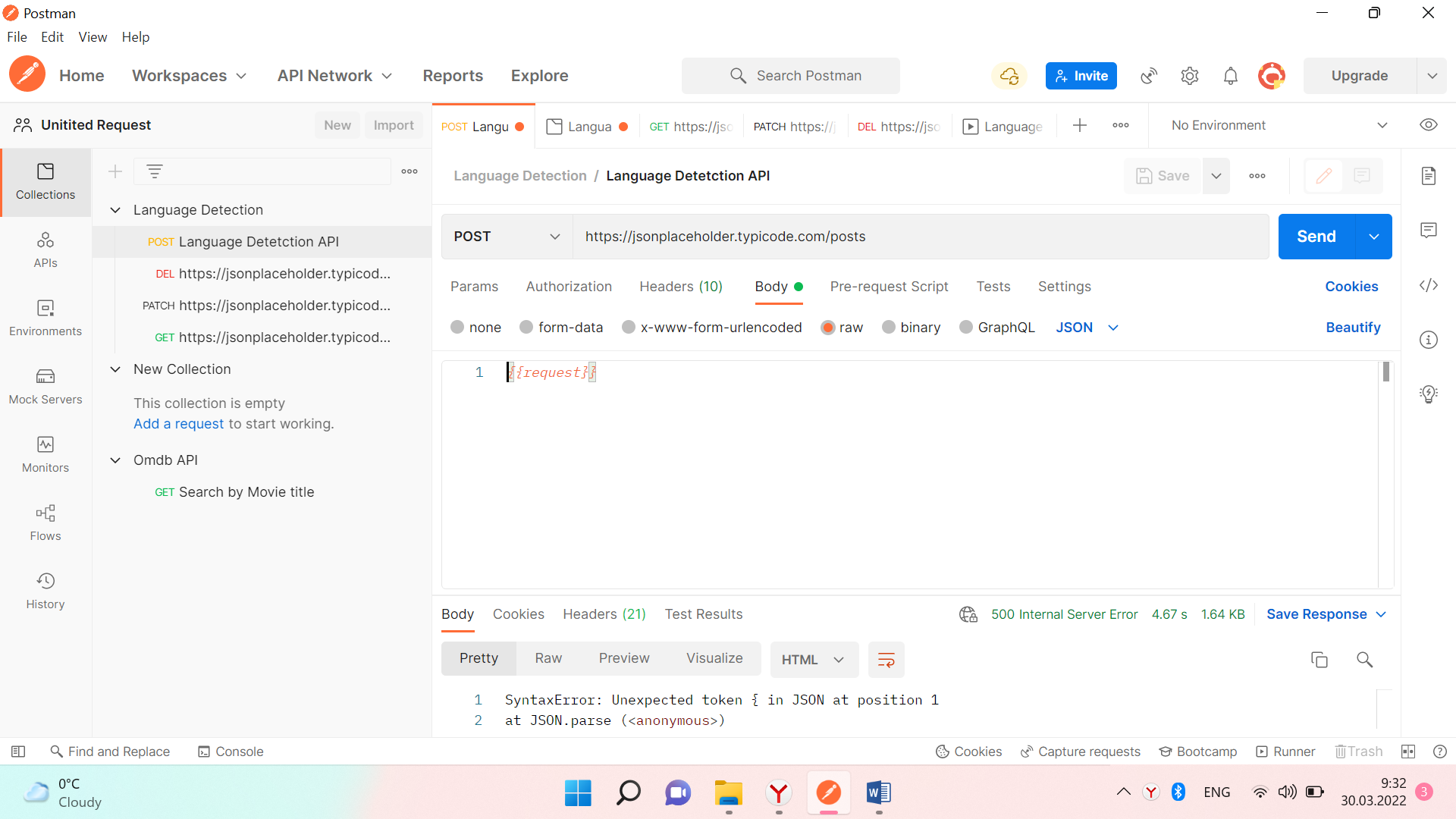
Postman предназначен для проверки запросов с клиента на сервер и получения ответа от бэкенда. Можно описать общение Postman с бэкендом в виде диалога: Postman: “Дай мне информацию по балансу именно этого пользователя”Backend: “Да, конечно, запрос правильный, получи информацию по балансу этого пользователя” Описанные выше ответы от бэкенда имеют свой код ошибки, которые приходят в ответе.

В первом случае — это ошибка с кодом 500 (Internal Server Error) внутренняя ошибка сервера, которая говорит о том, что сервер столкнулся с неожиданным условием, которое помешало ему выполнить запрос.

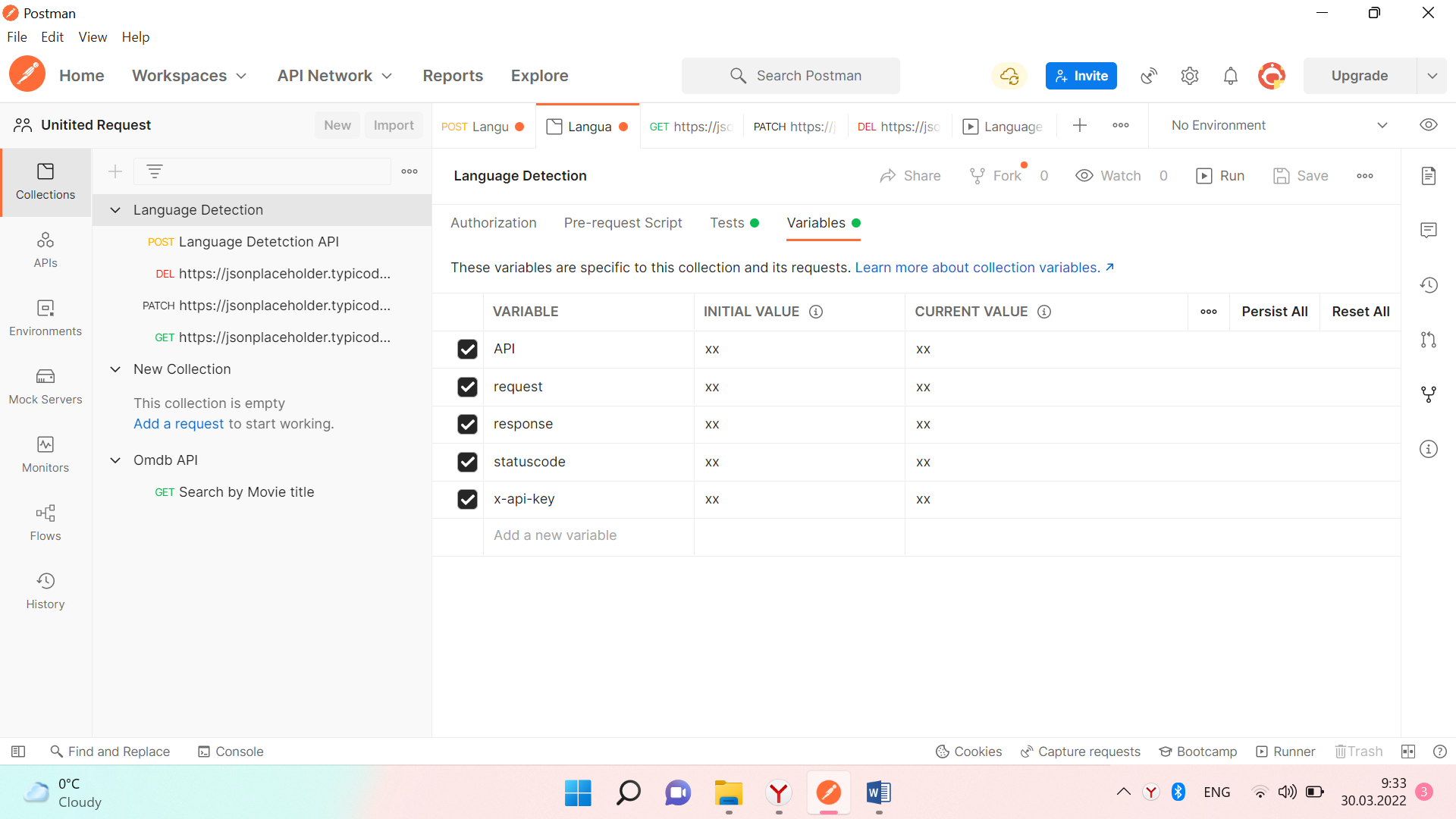
Во втором — 404 ошибка (Not Found) код ответа HTTP о том, что сервер не может найти данные по запросу, полученному от клиента. Именно для этого и предназначен Postman — для проверки запросов клиент → сервер по документации, чтобы убедиться, что всё работает на стороне бэкенда.

**ШАГ 1.** Создайте новый запрос в Postman и используйте фигурные скобки для определения полей, которые использовались в файле CS





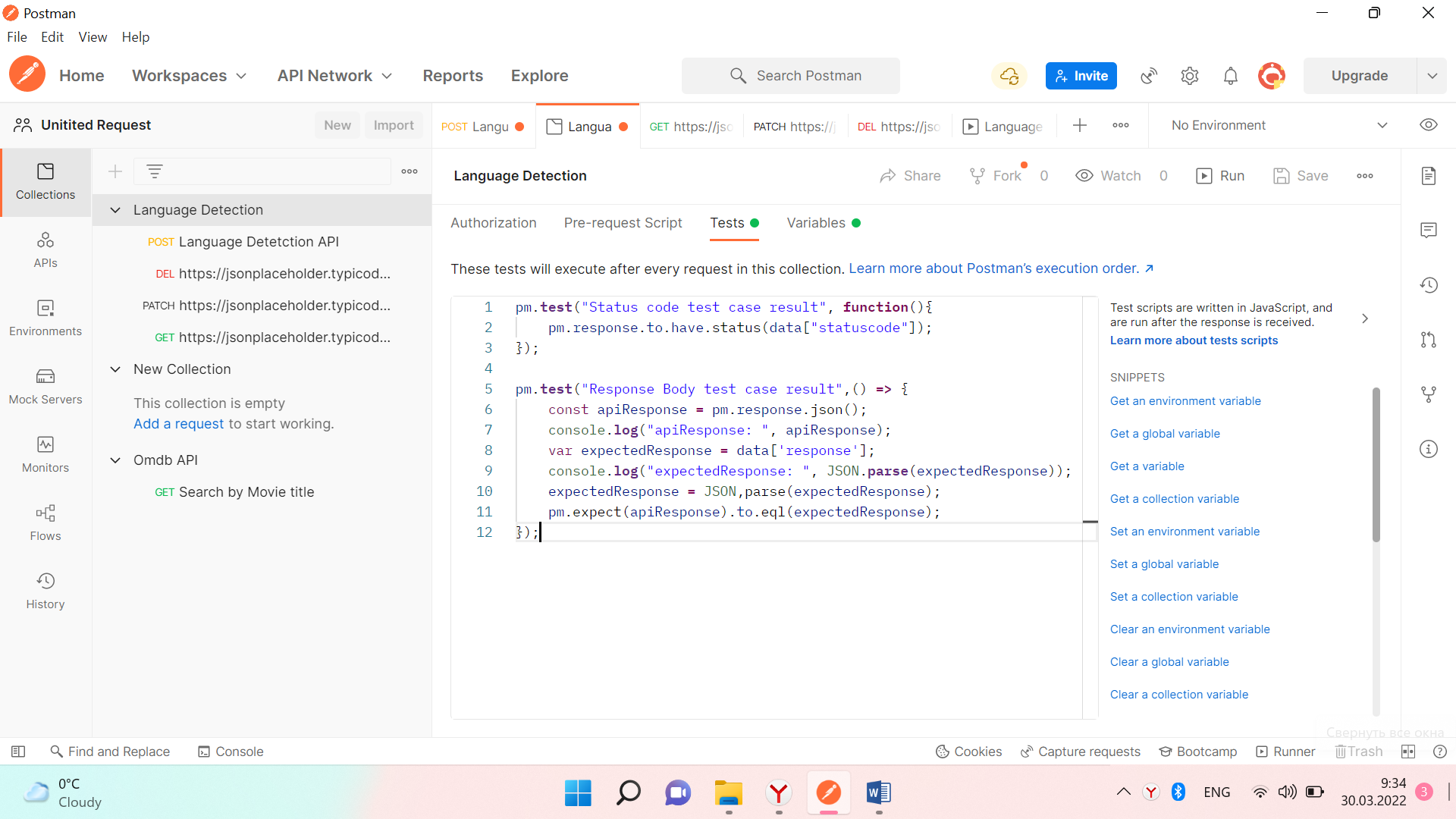
**Шаг 2.** Сохраните запрос API, создав новую коллекцию

**Шаг 3:** Создайте переменные в Коллекции, нажав кнопку редактирования в Коллекции со всеми полями, упомянутыми в файле CSV

**Шаг 4:** Теперь нам нужно написать тестовые примеры для API на вкладке «Тесты» в консоли Postman. Postman использует библиотеку **pm** и Javascript для запуска и тестирования кода, написанного на вкладке «Тесты».

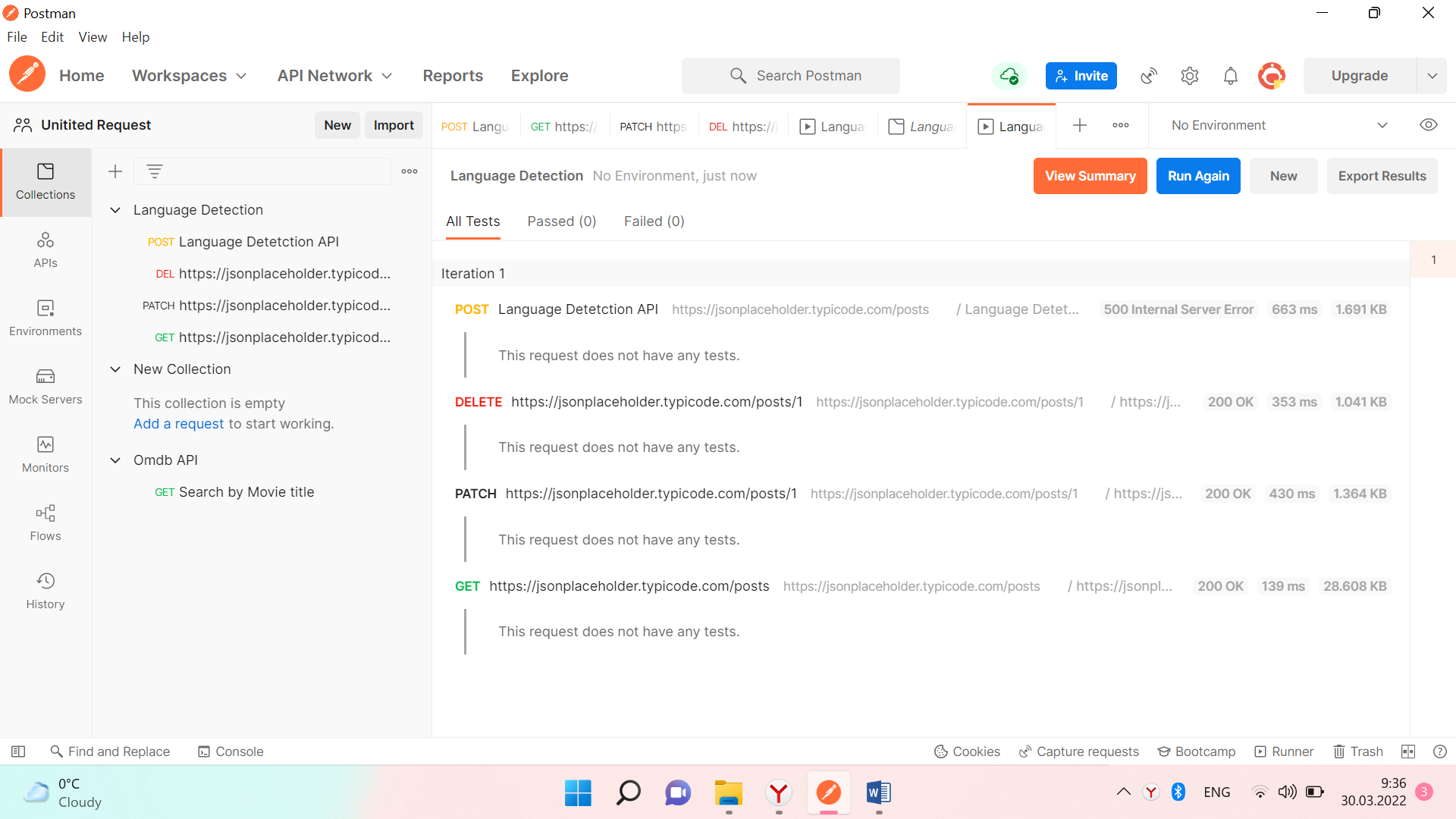
Здесь мы использовали два простых тестовых случая:

1. Чтобы проверить, совпадает ли код состояния ответа API с нашим кодом состояния ответа API в файле CSV.
2. Чтобы проверить, совпадают ли данные тела ответа API с данными в CSV-файле.



Поскольку мы использовали 7 итераций данных в файле CSV, почтальон будет запускать все запросы один за другим и проверять написанные нами тестовые примеры.

Итак, после завершения выполнения вы можете увидеть подробный отчет о тестовых случаях, которые не прошли и не прошли в консоли.



Это лучший и самый простой способ использовать инструмент Postman для автоматизации нескольких тестовых случаев для API.

Запрос — обращение от одного сервиса к другому сервису для получения или отправки информации. Запросы можно делать к любому стороннему сервису, который их поддерживает.

**Таблица запросов API:**

|  |  |
| --- | --- |
| Запрос | Описание |
| GET | **GET-запросы используются для получения данных от API** |
| POST | **POST ө это запросы используемые для отправки новых данных API** |
| PUT | **PUT-запросы используются для обновления уже существующих данных** |
| PATCH | **PATCH-запросы (как и PUT) используются для обновления уде существующих данных. Разница в том, что с помощью PATCH запросов можно обновить несколько записей за раз** |
| DELETE | **DELETE-запросы используются для удаления существующих данных** |
| HEAD | **Метод HEAD практически идентичен МЕТОДУ GET, за исключением отсутствия органа ответа. Другими словами, если GET /users возвращает список пользователей, то HEAD /users сделает тот же запрос, но не получит обратно список пользователей.** |
| OPTIONS | **Запросы OPTIONS более слабо определены и используются, чем другие, что делает их хорошим кандидатом для тестирования на наличие неустранимых ошибок API. Если API не ожидает запроса OPTIONS, хорошо поставить тестовый случай, который проверяет неисправное поведение. В двух словах, запрос OPTIONS должен возвращать данные, описывающие, какие другие методы и операции поддерживает сервер по заданному URL-адресу.** |
| TRACE | **Запросы TRACE используются для вызова удаленного теста с зацикливанием приложения по пути к целевому ресурсу. Метод TRACE позволяет клиентам просматривать любое сообщение, полученное на другом конце цепочки запросов, чтобы они могли использовать эту информацию для тестирования или диагностических функций.** |
| CONNECT | **Метод запроса CONNECT используется клиентом для создания сетевого подключения к веб-серверу по определенному протоколу HTTP. Хорошим примером является SSL-туннелирование. В двух словах, запрос CONNECT устанавливает туннель к серверу, идентифицированному определенным URL-адресом.** |

**Заключение:**

Итак, мы познакомились с базовыми возможностями инструмента Postman:

* Составили и отправили запросы;
* Задали параметры в запросах;
* Создали и переключались между окружениями;
* Написали базовые тесты;
* Создали и отредактировала коллекции;
* Выполнили запуск тестов.

Это только малая часть полезных и интересных функций Postman, при помощи которых можно тестировать API.