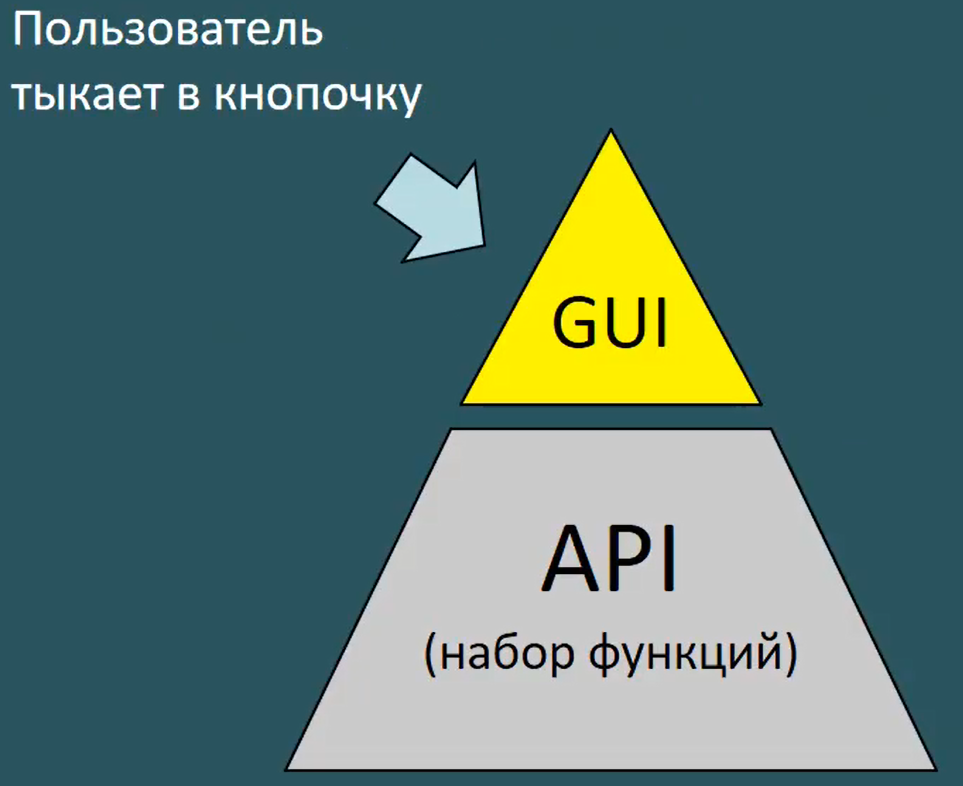
Лекция. Тестирование API

# Тестирование API

**API (Application Programming Interface) — программный интерфейс приложения.**

API является посредником между разработчиком приложений и какой-либо средой, с которой это приложение должно взаимодействовать. API упрощает создание кода, поскольку предоставляет набор готовых классов, функций или структур для работы с имеющимися данными.

Современные API часто принимают форму веб-сервисов, которые предоставляют пользователям (как людям, так и другим веб-сервисам) какую-то информацию. Обычно процедура обмена информацией и формат передачи данных структурированы, чтобы обе стороны знали, как взаимодействовать между собой.



API включает в себя:

* Операции, которые вы можете проводить
* Данные, которые подаются на входе
* Данные, которые получаем на выходе

По сути это функция или набор функции. Группируют их разработчики. Например, API для входа в систему: регистрация, авторизация; API создания и сохранения записи в реестре; API для отчетности: отчет1, отчет2, …отчет n.

Если захотим автоматизировать загрузку отчета, то увидим такую пирамиду тестирования



Тестирование через API - это тип тестирования программного обеспечения, который включает тестирование интерфейсов прикладного программирования напрямую и в рамках интеграционного тестирования, чтобы определить, соответствуют ли они ожиданиям функциональности, надежности, производительности и безопасности.

Т.е. тестирование через API – тестирование без графического интерфейса, но без unit тестов; это тестирование бизнес-логики; или это тестирование взаимодействия двух и более подсистем, которые общаются между собой по какому-то протоколу данных.

Обычно при обращении к веб API используются запросы HTTP. Существуют стандартные методы, которые могут содержаться в HTTP запросе в соответствии с CRUD (create, read, update, and delete, про CRUD <https://en.wikipedia.org/wiki/Create,_read,_update_and_delete>). Эти методы также называют HTTP глаголами:

* *GET. Наверное, самый популярный тип запроса. Используется для получения или чтения данных.*
* *PUT. Обычно используется для обновления ресурса.*
* *POST. Обычно используется для создания нового ресурса.*
* *DELETE. Удаляет данные.*
* *И другие*

Обычные GET запросы способен посылать веб-браузер. Для посылки других типов запросов могут потребоваться скриптовые языки или специальные инструменты.

Например, Postman, Jmeter, SoapUI, Advanced rest Client и другие.

С http протоколом используют SOAP протокол для отправки soap-сообщения и REST (Representational State Transfer, в переводе википедии «передача состояния представления») для отправки rest-сообщения. Важно понимать, что **REST – это не протокол и не стандарт, а архитектурный стиль**. SOAP используют HTTP как транспортный протокол, в то время как REST базируется на нем.

## SOAP

Тестирование SOAP с точки зрения техник тестирования ничем принципиально не отличается от работы с другими API, но для его проведения требуются предварительная подготовка (в плане теории протокола) и специальные инструменты для тестирования.

Чек-лист:

* XML
* XSD
* WSDL
* SOAP
* SoapUI

### XML

XML – язык разметки, схожий с HTML. Любое сообщение, отправляемое/получаемое через SOAP, – это XML-документ, в котором данные удобно структурированы и легко читаемы, например:

<?xml version="1.0"?>

<note>

<to>Юля</to>

<from>Наташа</from>

<heading>Напоминалка</heading>

<body>Не забудь выключить свет!</body>

</note>

XML-документ представляет собой обычный текстовый файл, в котором при помощи специальных маркеров создаются элементы данных, последовательность и вложенность которых определяет структуру документа и его содержание. Основным достоинством XML документов является то, что при относительно простом способе создания и обработки, они позволяют создавать структурированную информацию, которую хорошо "понимают" компьютеры.

Об XML подробно можно почитать <https://ru.wikipedia.org/wiki/XML> или <https://www.w3schools.com/xml/default.asp>.

Пример xml сообщения :

<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">

<soap:Header>

<wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing">EAEU://VN/CP/P.EE.02/P.EE.02.ACT.001</wsa:To>

<wsa:ReplyTo xmlns:wsa="http://www.w3.org/2005/08/addressing">

<wsa:Address>EAEU://RU/CP/P.EE.02/P.EE.02.ACT.002</wsa:Address>

</wsa:ReplyTo>

<int:Integration xmlns:int="urn:EEC:Interaction:v1.0">

<int:TrackID>urn:uuid:835a476b-41d6-11ea-8e2d-005056896c64</int:TrackID>

<int:AcceptTime>2020-01-28T14:00:17Z</int:AcceptTime>

</int:Integration>

<wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">int://CP/P.EE.02/0.0.1/P.EE.02.PRC.005/P.EE.02.TRN.005/P.MSG.PRS</wsa:Action>

<wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing">urn:uuid:4242a300-2163-4a56-8daf-3072a31b5ac3</wsa:MessageID>

<int:ProcedureID xmlns:int="urn:EEC:Interaction:v1.0">urn:uuid:24c33a15-ff05-4124-b3ca-250ba3c9ba60</int:ProcedureID>

<int:ConversationID xmlns:int="urn:EEC:Interaction:v1.0">urn:uuid:d09bf0a7-2026-4201-bb2f-85c7fc12b9d0</int:ConversationID>

<wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">urn:uuid:0833a7c7-7e2d-426d-8221-b69f108db904</wsa:RelatesTo>

</soap:Header>

<soap:Body>

<sgn:ProcessingReceipt xmlns:sgn="urn:EEC:signal:v1.0">

<sgn:SignalId>c5b37073-c23a-4fd0-8ac9-15a9cfcd1d1f</sgn:SignalId>

<sgn:DateTime>2020-01-28T14:00:17.076Z</sgn:DateTime>

</sgn:ProcessingReceipt>

</soap:Body>

</soap:Envelope>

Состоит из

<?xml version="1.0" encoding="UTF-8"?> - Объявление XML. Здесь указывается версия языка xml version="1.0". Объявление для версии 1.0 необязательно, поэтому можете увидеть xml без объявления, то это значит, что системы по умолчанию будут понимать, что это версия 1.0. Кроме версии XML, объявление может также содержать информацию о *кодировке* документа.

Header – заголовоки xml-сообщения. Это относится именно к конкретному примеру, в нем заполняется информация об отправителе и получателе.

Body – это тело самого сообщения.

Важно обратить внимание на [namespaces](https://www.w3schools.com/xml/xml_namespaces.asp) (метод разрешения конфликтов при описании элементов в XML) – в SOAP их использование необходимо. В XML имена элементов определяются разработчиком. Это часто приводит к конфликту при попытке смешать документы XML из разных приложений XML. При использовании префиксов в XML должно быть определено пространство имен для префикса.

Пространство имен может быть определено атрибутом xmlns в начальном теге элемента.

Объявление пространства имен имеет следующий синтаксис. xmlns: prefix = " URI ".

Почитать <https://www.w3schools.com/xml/xml_namespaces.asp>

В примере видим, что в <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"> был объявлен префикс soap.

<sgn:ProcessingReceipt xmlns:sgn="urn:EEC:signal:v1.0">

<sgn:SignalId>c5b37073-c23a-4fd0-8ac9-15a9cfcd1d1f</sgn:SignalId>

<sgn:DateTime>2020-01-28T14:00:17.076Z</sgn:DateTime>

</sgn:ProcessingReceipt>

Здесь для всех реквизитов,входящих в ProcessingReceipt объявлен неймспейс :sgn

Если бы объявили так, то xml сообщение будет некорректным и не пройдет проверку структуры xml:

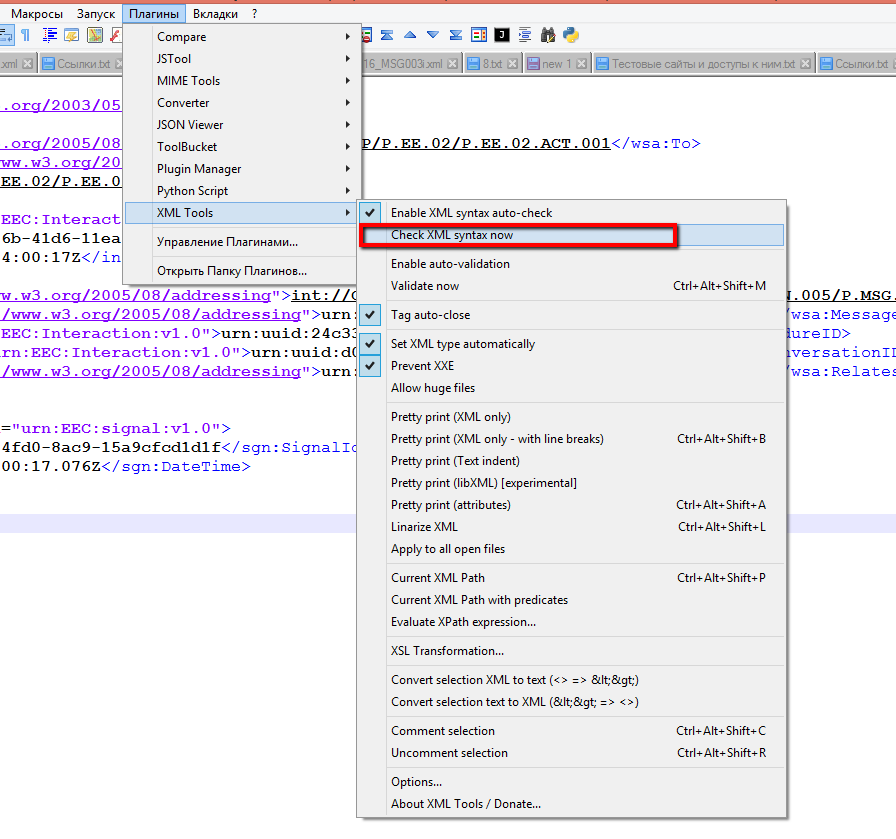
<sgn:ProcessingReceipt>

<sgn:SignalId xmlns:sgn="urn:EEC:signal:v1.0">c5b37073-c23a-4fd0-8ac9-15a9cfcd1d1f</sgn:SignalId>

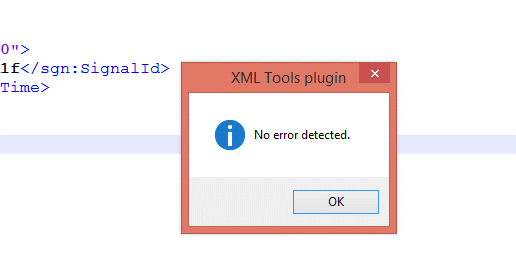
<sgn:DateTime>2020-01-28T14:00:17.076Z</sgn:DateTime>

</sgn:ProcessingReceipt>

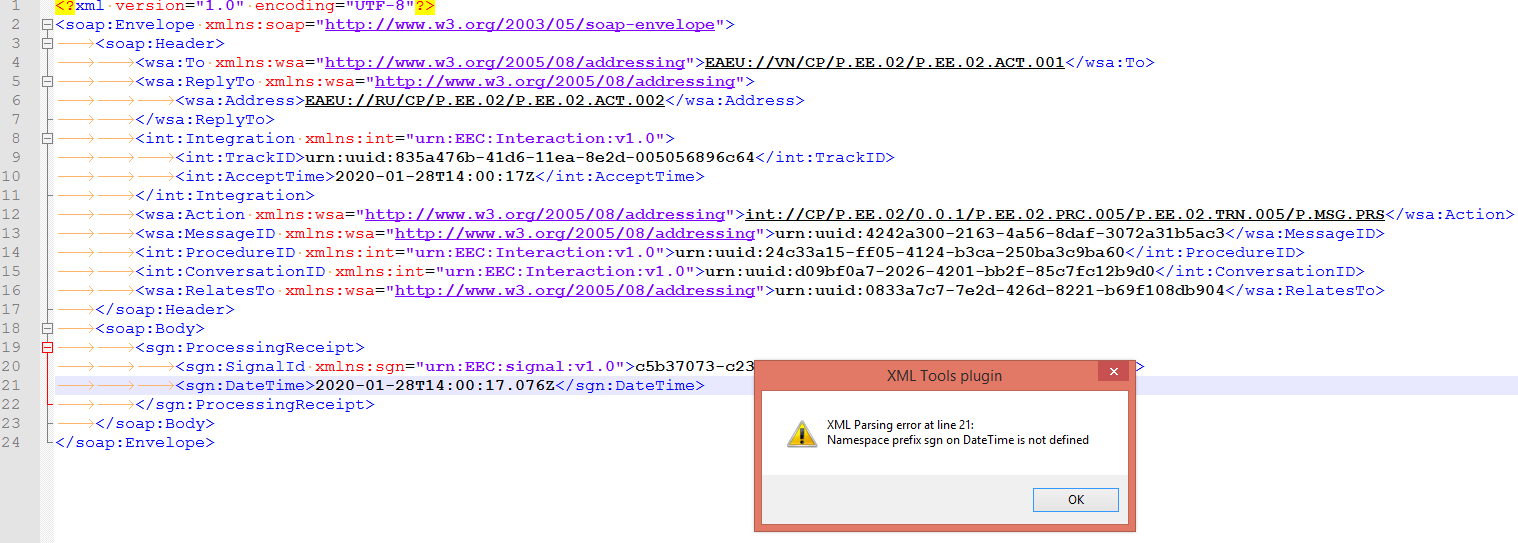
Проверить синтаксис xml легко в notepade с помощью плагина XML Tools



Ошибки отсутствуют



Если нарушить структуру как указано на примере выше и проверить xml, то получим:



### XSD

При работе всегда удобно иметь стандартизированное описание возможных XML-документов и проверять их на корректность заполнения. Для этого существует XML Schema Definition (или сокращенно XSD). Две главные фичи XSD для тестировщика – это описание типов данных и наложение ограничений на возможные значения. Например, элемент можно сделать необязательным для заполнения и ограничить его размер 255 символами с помощью XSD:

...

<xs:element name="body" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="255"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

...

, где

minOccurs="0" указывает на необязательность

restriction base="xs:string" – тип данных строка

maxLength value="255" – ограниченное кол-во символов - до 255.

XSD – это сила и мощь. Чем подробнее описан XSD, тем меньше головной боли доставит вам тестирование сервиса. С помощью выстроенной схемы сервис сам сможет валидировать полученные данные и возвращать пользователю ошибку. Следовательно, первым делом при тестировании SOAP вы должны проверить документацию – XSD. Довольно часто встречаются ошибки в XSD в виде некорректно прописанных ограничений или случайно закравшихся кириллических символов. Подробнее прочитать про XSD можно опять же на [w3schools](https://www.w3schools.com/xml/schema_intro.asp) и [codenet (по-русски)](http://www.codenet.ru/webmast/xml/part6.php" \t "_blank).

Есть инструменты позволяющие валидировать xml сообщение на соответствие xsd схеме вручную. Например, Notepade плагин XMLTools

Пример xml сообщения 

В body

<doc:ResourceStatusDetails xmlns:ccdo="urn:EEC:M:ComplexDataObjects:v0.4.4" xmlns:csdo="urn:EEC:M:SimpleDataObjects:v0.4.4" xmlns:doc="urn:EEC:R:ResourceStatusDetails:v0.4.4">

<ccdo:EDocHeader>

<csdo:InfEnvelopeCode>P.MM.02.MSG.006</csdo:InfEnvelopeCode>

<csdo:EDocCode>R.007</csdo:EDocCode>

<csdo:EDocId>ed4a2391-adb3-49c2-af2d-15b58c7f63a3</csdo:EDocId>

<csdo:EDocRefId>d5bffb3c-4900-4c5c-b794-1e035a4bac0a</csdo:EDocRefId>

<csdo:EDocDateTime>2020-03-12T23:00:04.962Z</csdo:EDocDateTime>

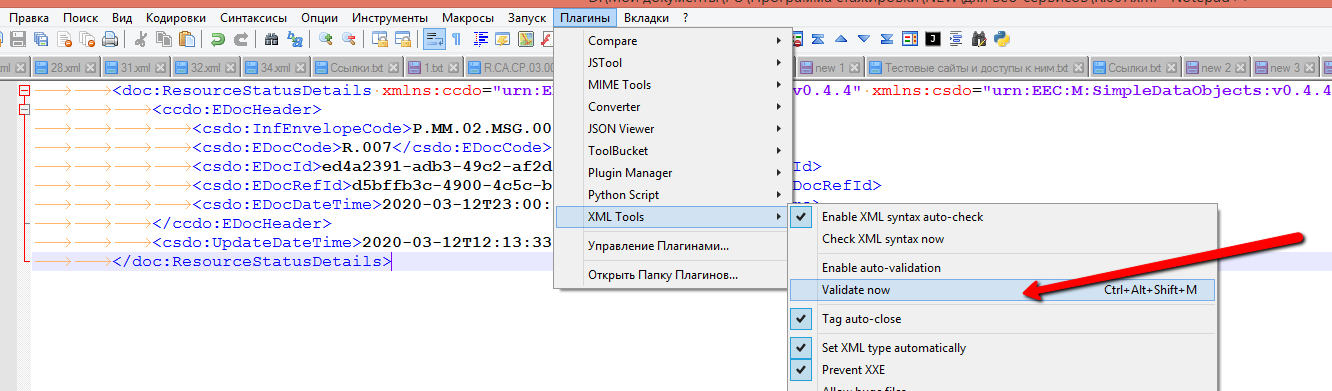
</ccdo:EDocHeader>

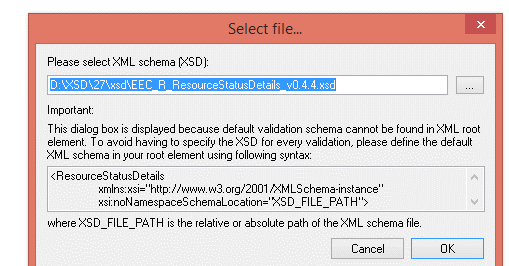
<csdo:UpdateDateTime>2020-03-12T12:13:33.41Z</csdo:UpdateDateTime>

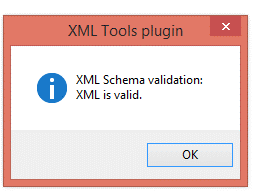
</doc:ResourceStatusDetails>

Уберем лишние реквизиты, не относящиеся к xsd 

Корневой элемент doc:ResourceStatusDetails xmlns:doc="urn:EEC:R:ResourceStatusDetails:v0.4.4"

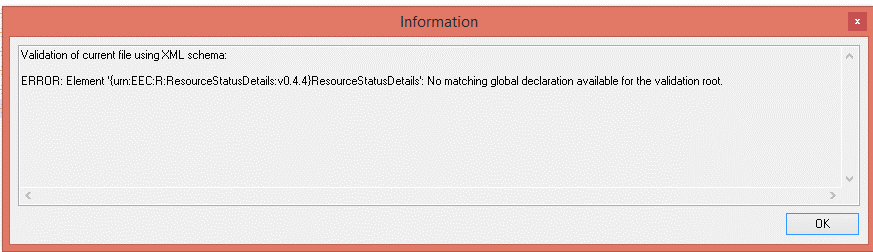






Если «сломать» xml или при валидации указать другую схему – выйдет ошибка

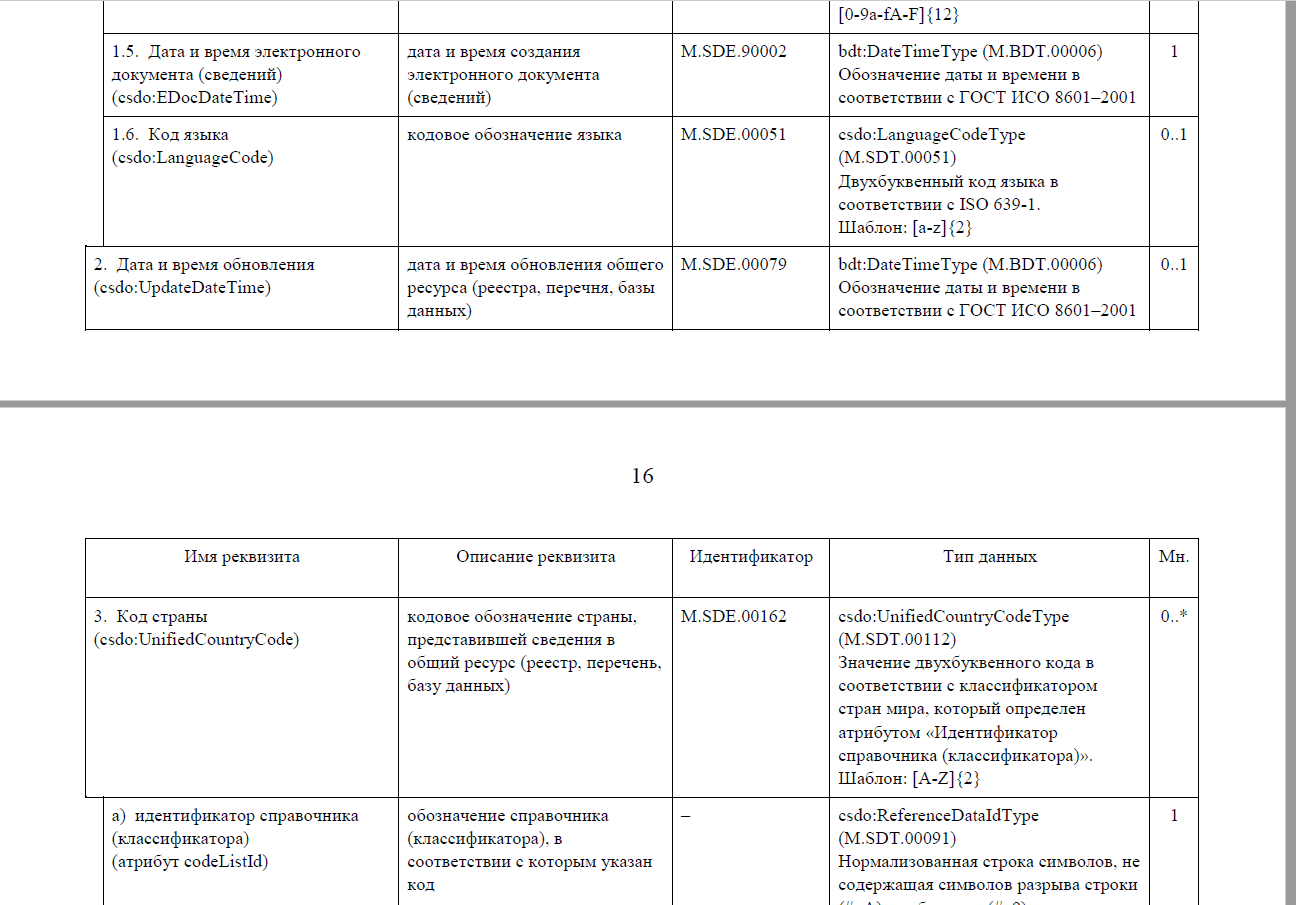
Например:



Иногда схемы электронных документов описывают в документации (но не всегда), что упрощает понимание и тестирование сервисов.

Например, схема документа R.007 – пример xml выше описана так.

Пример куска такого описания



Отдельно можно открыть описание полностью.

Обозначения множественности описаны в документе.

Здесь на скриншоте:

1 – обязательный элемент, единичный

0..\* - необязательный, множественный

0..1 – необязательный, единичный

Так же здесь можно увидеть какой тип данных у соответствующего реквизита, может быть регулярное выражение

csdo:LanguageCode - Двухбуквенный код языка в соответствии с ISO 639-1.

Шаблон: [a-z]{2}

Это значит, что если указать язык RU, то уже данные в xml не будут соответствовать схеме.

csdo:EDocDateTime - Обозначение даты и времени в соответствии с ГОСТ ИСО 8601–2001.

Это тоже очень важный и интересный реквизит для тестирования. Ранее при знакомстве с техниками тестирования мы говорили о том, что есть стандарты заполнения каких-либо полей, и это необходимо учитывать при тестировании.

Вот этот тот же случай. Здесь при тестировании мы должны ознакомиться с ГОСТ ИСО 8601–2001. Какие форматы даты и времени бывают с учетом указания часового пояса и без. Даты и время в формате Периода.

Ознакомиться с ГОСТ ИСО 8601–2001 необходимо для того, чтобы иметь представление, какое это время, что оно означает, в чем разница

2020-03-12T12:13:33.41Z

Или

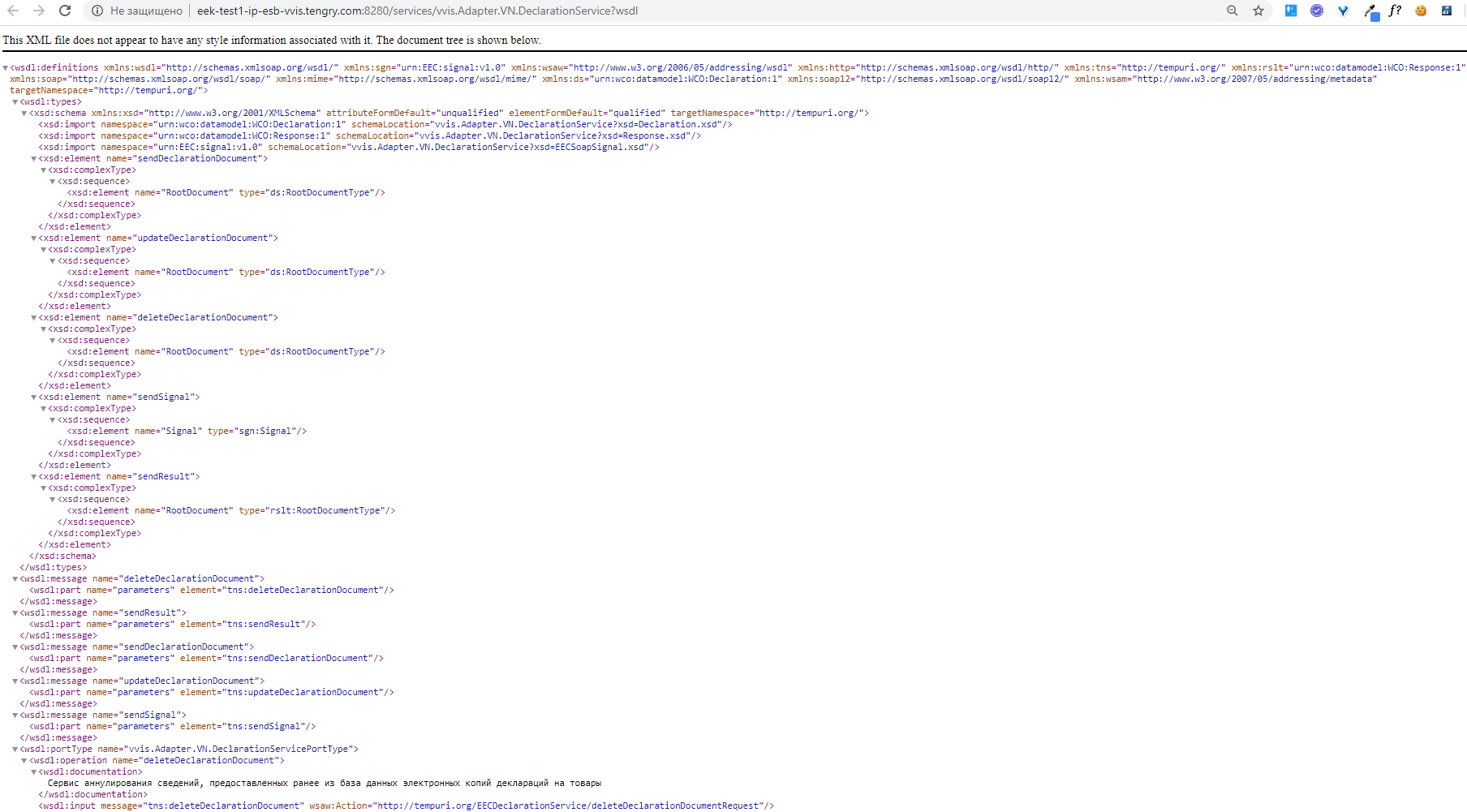
2020-03-12T12:13:33.41+03:00

Или почему нецелесообразно использовать формат 2020-03-12T12:13:33 в ПО, которое будет использоваться в разных часовых поясах.

### WSDL

[Web Services Description Language (WSDL)](https://www.w3schools.com/xml/xml_wsdl.asp) – это язык на основе XML, который используется для описания веб-сервисов. В WSDL-документе содержится информация о местонахождении сервиса и доступных методах (операциях); для каждого метода определяются параметры отправляемого и получаемого сообщения. Обратите внимание на то, что XSD может быть «встроен» внутрь WSDL-документа. WSDL-документ – это «капелька магии» в мире SOAP.

<http://eek-test1-ip-esb-vvis.tengry.com:8280/services/vvis.Adapter.VN.DeclarationService?wsdl>



### SOAP

У SOAP  есть две основные версии протокола: 1.1 и 1.2. Для начинающего тестировщика различия между этими версиями не так принципиальны. Важно лишь знать, какую версию использует ваш сервис (в некоторых случаях применяются сразу обе версии, и тогда объем тестирования возрастает). SOAP задает формат сообщений, которыми обмениваются клиент и сервер, в нем же описываются подробности обработки приложениями конкретных фрагментов сообщений. Например, определенные элементы в заголовке позволяют создавать приложения, в которых сообщения сначала проходят через несколько промежуточных «станций», а только потом достигают конечного получателя.

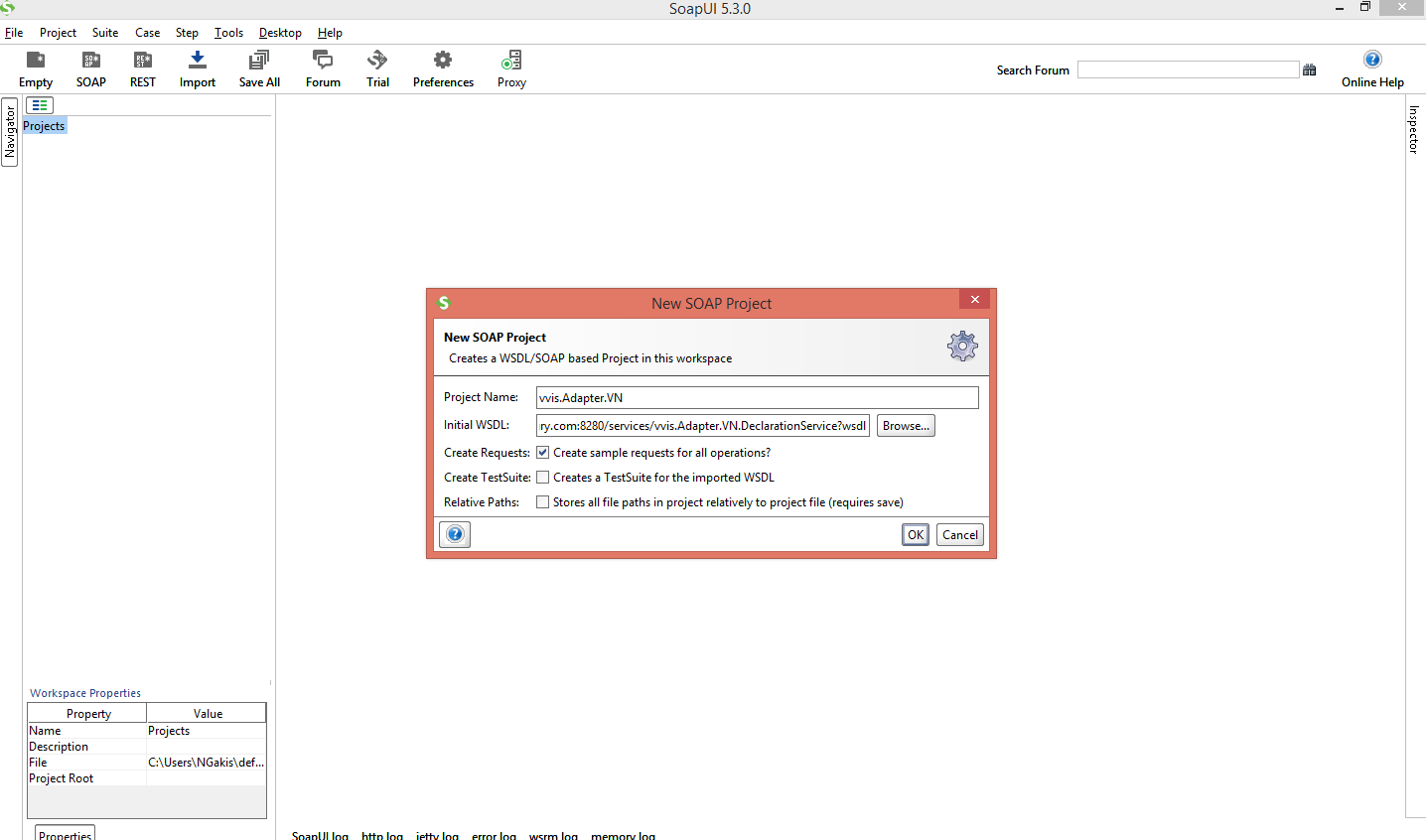
Прочитать о формате SOAP сообщений можно на [w3schools](https://www.w3schools.com/xml/xml_soap.asp).

### SOAP UI – инструмент для тестирования

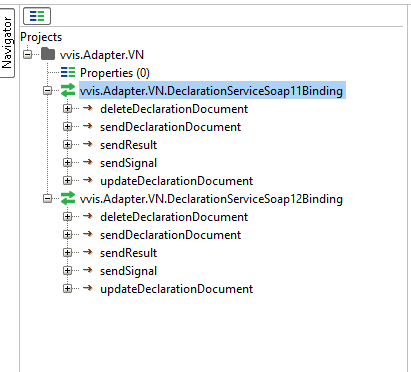
Тестирование SOAP практически всегда подразумевает использование [SoapUI](https://www.soapui.org/" \t "_blank). Прочитать про использование этого инструмента можно в разных источниках ([источник 1](http://www.proghouse.ru/programming/20-soapui), [источник 2](http://www.guru99.com/webservice-testing-beginner-guide.html)), но эффективнее всего будет ознакомиться с [официальной документацией](https://www.soapui.org/getting-started.html) или <https://www.soapui.org/resources/tutorials/soap-sample-project.html?utm_source=soapui&utm_medium=starterpage>. У этого инструмента довольно широкий функционал, который дает возможность писать автоматизированные тесты в том числе.

Для начального уровня достаточно уметь отправлять запросы. Для этого необходимо научиться создавать проект на основе WSDL. SoapUI может сгенерировать все необходимые запросы для вас; вам останется лишь проверить правильность их заполнения и нажать кнопочку «Send». После выработки навыков создания корректных запросов вы должны овладеть искусством формирования некорректных запросов, вызывающих появление ошибок.

Итак, в SoapUI необходимо создать новый проект и указать ссылку на wsdl

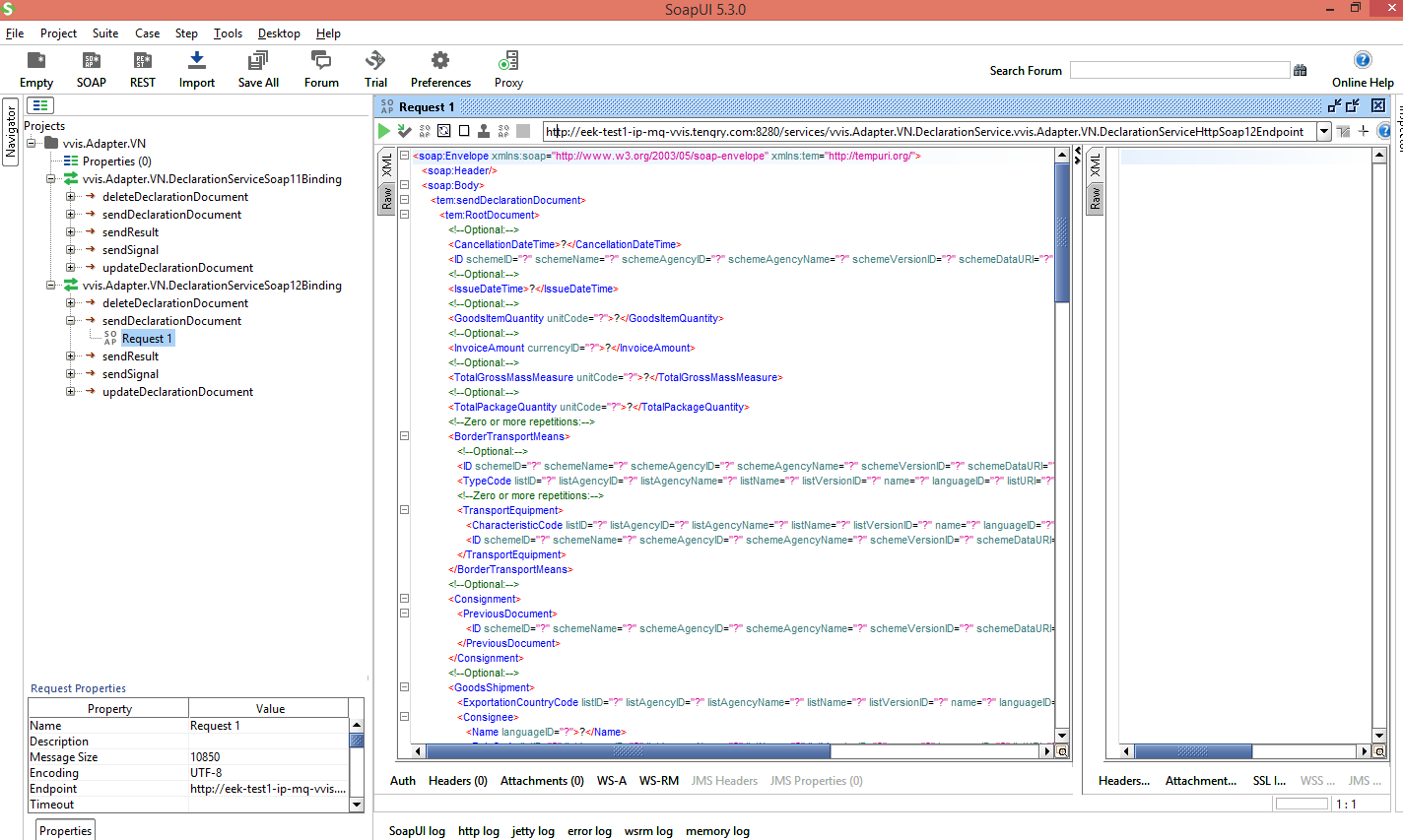


После создания project видим созданный сервис с версией 1.1 и 1.2



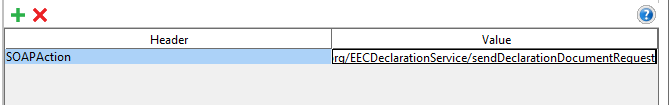
В конкретно данном случае разработчик указал, что необходимо использовать версию soap 1.2

При выборе метода – request мы видим уже теле soap сообщения, которое было автоматически создано. Здесь мы можем редактировать тело сообщения

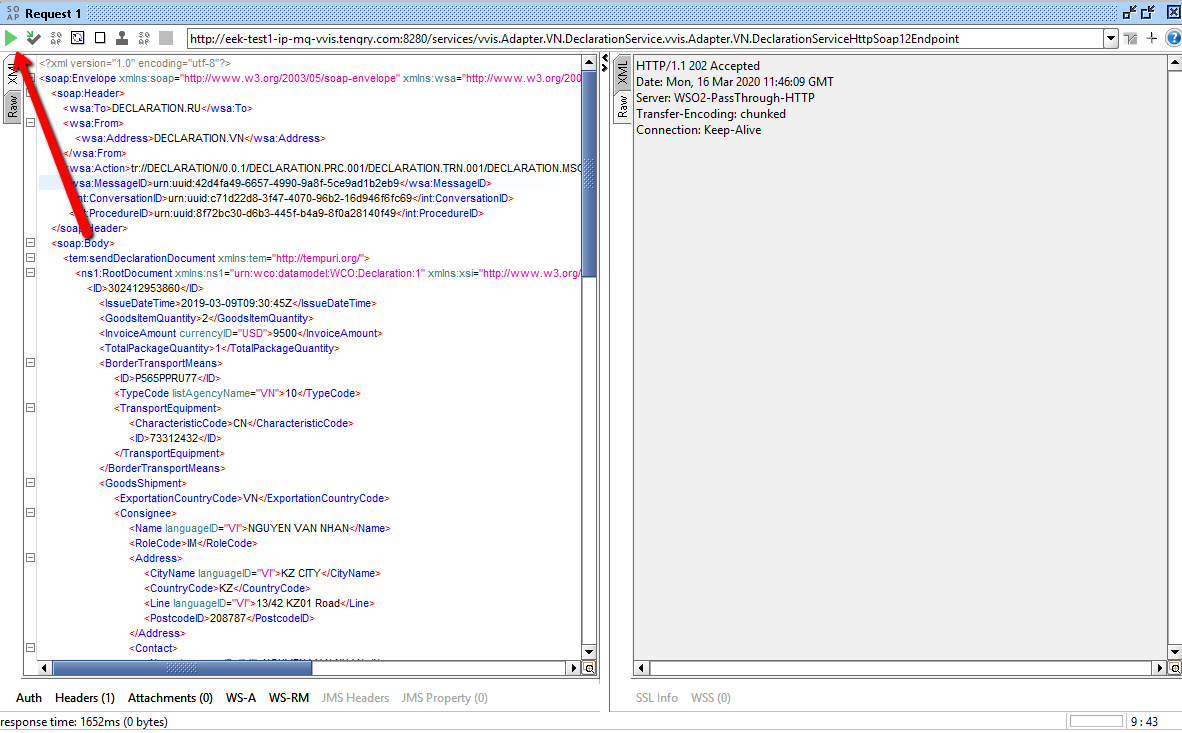


Используем подготовленную xml

При этом заполним header:

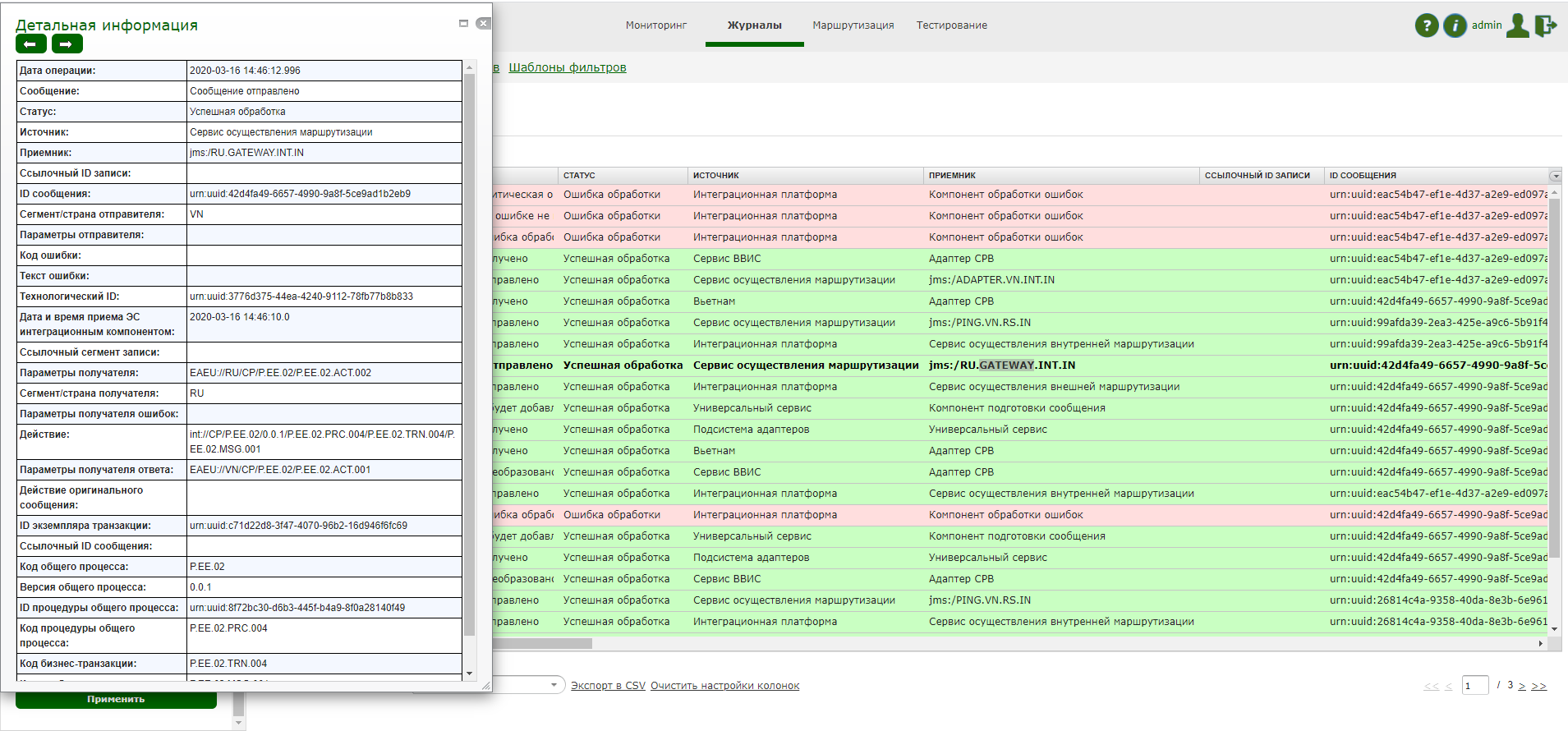


Вызовем метод



Этот метод производит трансформацию сообщения и перенаправляет в менеджер очередей.

С помощью GUI шлюза можно увидеть, что сообщение было отправлено



С помощью SOAPUI можно создать тестовые наборы, реализовывать автоматические тесты. Это рассматривать сейчас мы не будем. Для начала достаточно научиться тестировать сервисы вручную.

<https://software-testing.ru/library/testing/testing-automation/2968-api>

## REST API

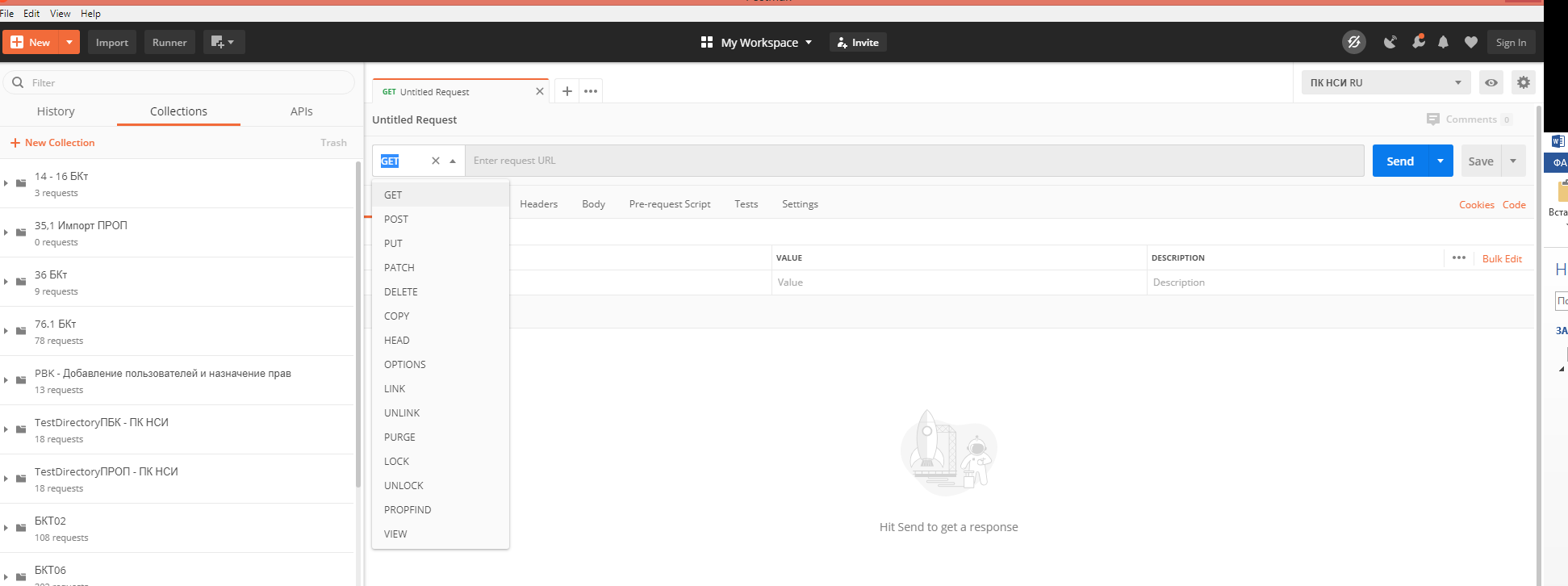
REST (от англ. *Representational State Transfer* – «передача состояния представления») обеспечивает общение между клиентом (как правило, это браузер) и сервером с помощью обычных HTTP-запросов (GET, POST, PUT, DELETE и т. д), передавая информацию от клиента в параметрах самих запросов, информацию от сервера – в теле ответа (который может быть, например, JSON-объектом или XML-документом). REST является архитектурным стилем, а не стандартом.

Ранее при рассмотрении веб-приложений мы уже рассматривали как происходит вызов сервиса.

Для тестирования rest запросов использовать можно, как и soapUI, так и Postman или другой инструмент.

Рассматривать будем именно Postman (<https://www.postman.com/>)

С ПОМОЩЬЮ Postman можно отправлять запросы с разными методами. Стандартные – CRUD

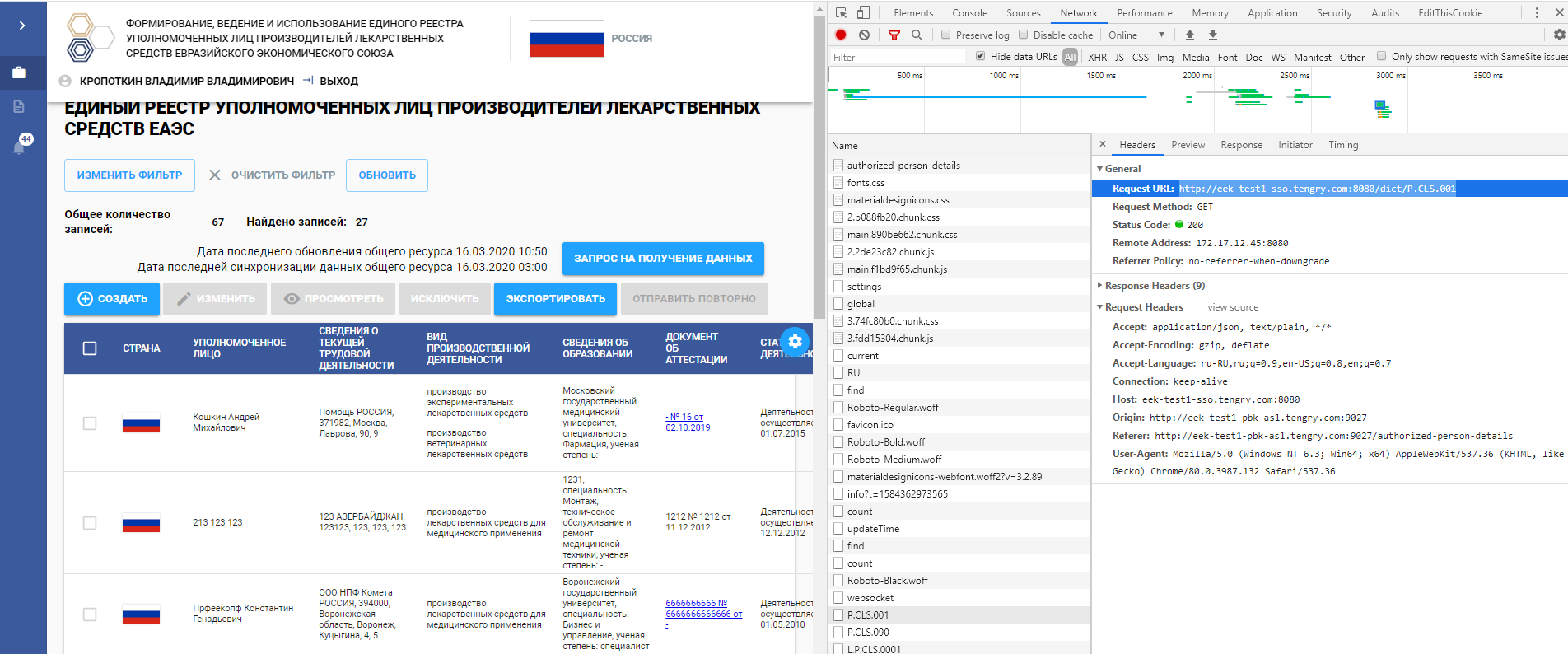


Заполняя при этом заголовки, параметры, скрипты, теле сообщение и т.д.

Рассмотрим простой вариант get-запроса.

Веб-приложение, с которым вы ранее уже были знакомы при загрузке страницы тянет справочники из другого модуля с помощью get-запроса

<http://eek-test1-pbk-as1.tengry.com:9027/authorized-person-details>

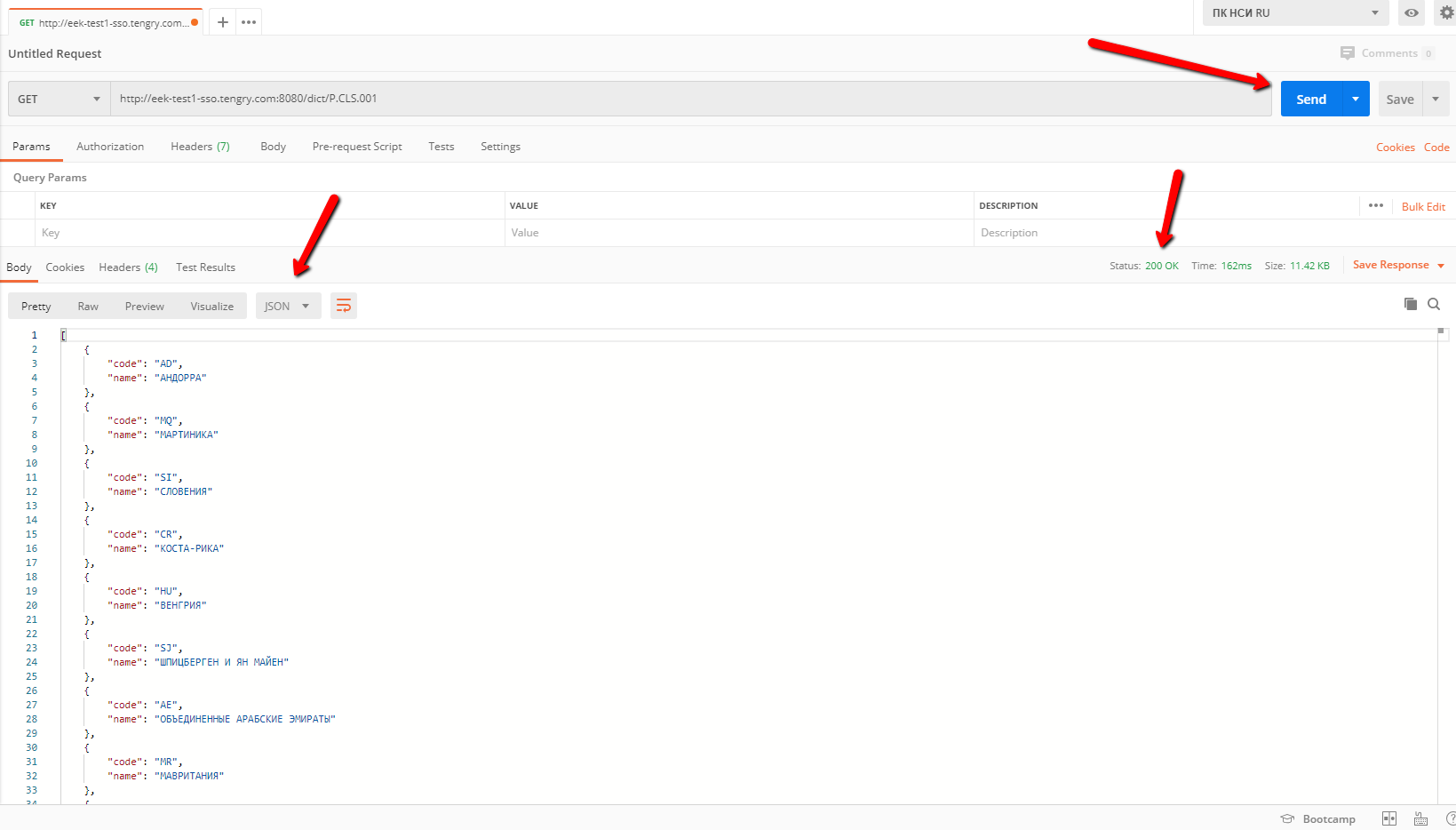


Мы уже говорили, что просто get-запрос можно выполнить и с помощью браузера

<http://eek-test1-sso.tengry.com:8080/dict/P.CLS.001>



Или с помощью инструмента postman



Результатом вызова сервиса будет статус код 200 OK. Этот статус ранее так же рассматривали, он означает Усепх. Ответ вернулся в формате Json

<https://ru.wikipedia.org/wiki/JSON>

### JSON

— текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

* Набор пар *ключ: значение*.
* Упорядоченный набор *значений*. Во многих языках это реализовано как массив, вектор, список или последовательность.

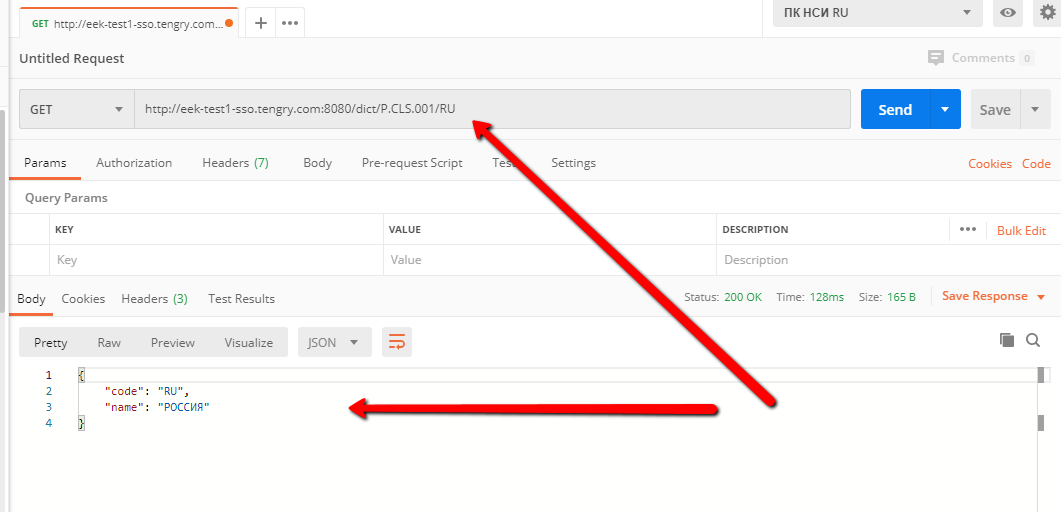
Форматировать текст в удобочитаемый вид можно в notepade плагин JSON Viewer. Или в Chrome плагин <https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=ru>

И теперь результат запроса в браузере chrome стало более удобнее читать



В данном сервисе был реализован поиск по коду страны.

http://eek-test1-sso.tengry.com:8080/dict/P.CLS.001/RU



Чтобы выполнить запрос по условию, выбираем уже метод POST. Добавляем параметр Find

<http://eek-test1-sso.tengry.com:8080/dict/P.CLS.001/find>

{

"constraints": [

{

"ctype": "simple",

"property": "name",

"op": "LIKE",

"value": "Казах"

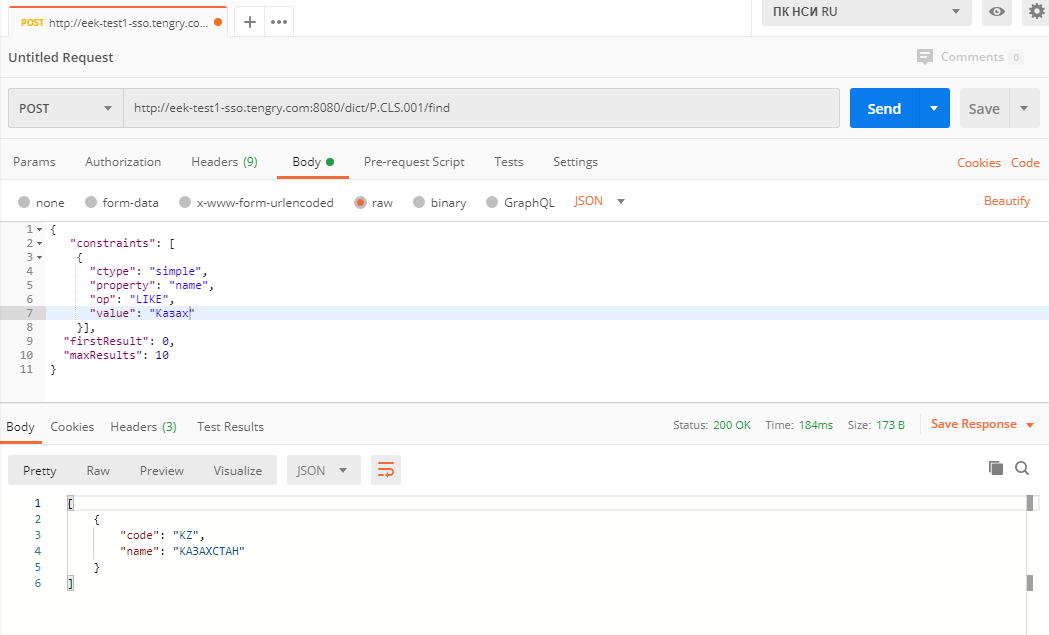
}],

"firstResult": 0,

"maxResults": 10

}

Этот запрос используется на форме заполнения карточки в поле «Страна» (например, для поиска по части названия страны Казахстан)



Таким образом мы проверили поиск значения в справочнике по части имени.