**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Владимирский государственный университет имени**

**Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

**Институт инновационных технологий**

**Факультет информационных технологий**

**Кафедра Информационных систем и программной инженерии**

Лабораторная работа № 3

Тестирование REST API

**Подготовили:**

Конушин А. В.

Мигачёва А. А.

**Владимир 2018**

# **Цель:**

Научиться проектировать тестовые сценарии для тестирования REST API. Познакомиться с инструментом тестирования REST API - postman.

# **Теоретическая часть:**

## **API**

API расшифровывается как «интерфейс программирования приложений». Он позволяет осуществлять связь и обмениваться данными между двумя отдельными модулями программы. Система программного обеспечения, реализующая API, содержит функции/подпрограммы, которые могут быть выполнены с помощью другого программного обеспечения.

Тестирование API концентрируется на слое бизнес-логики архитектуры программного обеспечения. Вместо стандартных видов ввода пользовательских данных (заполнение форм) для передачи данных используется программное обеспечение. Для API тестирование потребуется само тестируемое приложение и приложение для работы с API.

Тестирование API отличается от других видов отсутствием пользовательского интерфейса, следовательно, мы должны настроить с необходимым набором параметров среду, тестирующую API, а затем проанализировать результаты теста.

Архитектура взаимодействия представлена на рисунке ниже.

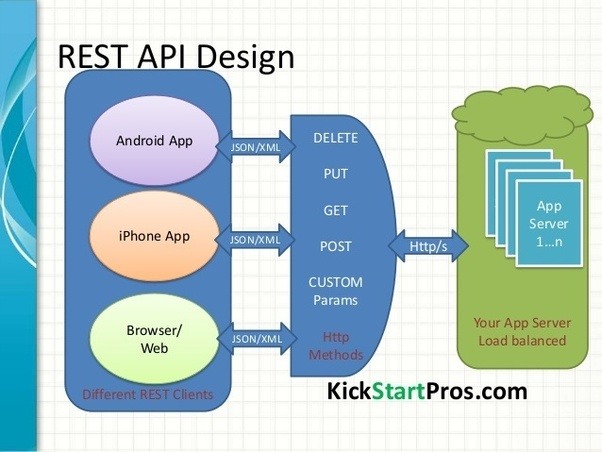


Рисунок 1 - Архитектура взаимодействия приложения[[1]](#footnote-1)

## **REST**

[REST](http://habrahabr.ru/post/38730/) (Representational State Transfer — «передача состояния представления») — это стиль архитектуры программного обеспечения для построения распределенных масштабируемых веб-сервисов. Основная идея в том, чтобы придавать запросам определённый смысл. Таким образом, данные HTTP-запросы будут иметь различную смысловую нагрузку:

* GET /object/list
* POST /object/list
* PUT /object/list

Для общей информации приведен список существующих видов запросов: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT, TRACE. Некоторые могут и вовсе не поддерживаться ни клиентским, ни серверным приложением.

HTTP-коды ответов помогают в определении смысла самих ответов. Существует примерно 38 кодов ответа, ниже приведен их список.

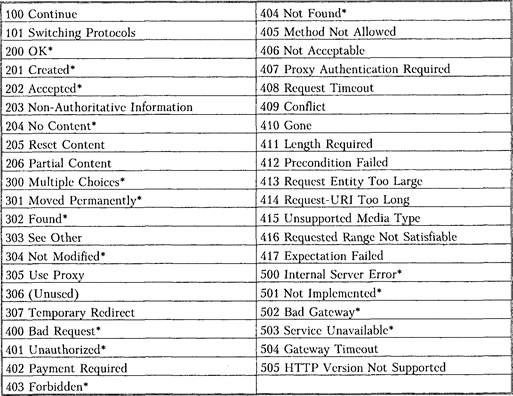


Рисунок 2 - HTTP-коды ответов[[2]](#footnote-2)

Требования к архитектуре:

Существует шесть обязательных ограничений для построения распределённых REST-приложений:

1. Модель клиент-сервер
2. Отсутствие состояния
3. Кэширование
4. Единообразие интерфейса
5. Слои
6. Код по требованию

## **JSON**

JSON (JavaScript Object Notation) — представляет собой [текстовый формат](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82) [обмена данными](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D0%B5%D0%BD_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8), основанный на [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript). Формат считается независимым от языка и может использоваться практически с любым [языком программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

Пример JSON-сообщения:

{

"firstName": "Иван",

"lastName": "Иванов",

"address": {

"streetAddress": "Московское ш., 101, кв.101",

"city": "Ленинград",

"postalCode": "101101"

},

"phoneNumbers": [

"812 123-1234",

"916 123-4567"

]

}

Возвращаемые объекты, как правило, описываются в спецификации API, в блоке DEFINITIONS.

Важно отметить, что JSON это лишь один из возможных форматов, в качестве примера можно привести такой формат как SOAP.

## **Инструменты‎ тестирования REST API**

Обычные GET запросы можно посылать при помощи браузера. Но существует множество специальных инструментов, которые предназначены для разработки и тестирования REST API. Они предоставляют возможность не только отправлять различные типы запросов, но и сохранять запросы, показывать результаты в различных форматах, выступать в роли прокси-сервера. Среди таких инструментов:

* [Postman](https://www.getpostman.com/). Расширение для Google Chrome, которое в бесплатной версии позволяет посылать запросы, записывать их, показывать историю. Удобно и понятно.
* [jMeter](http://jmeter.apache.org/). Инструмент, получивший известность прежде всего благодаря нагрузочному тестированию, которое можно проводить с его помощью.
* [Fiddler](http://www.telerik.com/download/fiddler). Позволяет просматривать посылаемые HTTP запросы. И много чего еще.
* [SoapUI](http://sourceforge.net/projects/soapui/). Мощный продукт для разработки и тестирования веб приложений. Сам я его не использовал, но слышал много отзывов - и хороших, и плохих.

## **Тестирование REST API**

Для того чтобы освоить тестирование REST API достаточно выполнить несколько шагов.

Работа со средствами разработчика в браузере. После совершения обычных действий на сайте будет отображаться множество запросов, которые будут отвечать за каждое из совершенных действий. Далее необходимо исключить запросы, не относящиеся к логике. Для этого можно использовать различные текстовые фильтры. После детального анализа можно сделать заключения о том, что подразумевают в себе эти запросы. Можно попробовать самостоятельно воспроизвести такие запросы и проверить поведение.

Ниже приведен примерный перечень тех тестов, которые необходимо выполнить при тестировании API:

* Обзорное тестирование – набор вызовов для проверки общей работоспособности системы;
* Проверка граничных значений
* Проверка типов полей запроса/ответа
* Минимальный запрос
* Максимальный запрос
* Минимальный ответ
* Максимальный ответ
* Корректный запрос
* Корректный ответ
* Некорректный запрос
* Некорректный ответ
  + Успешный код ошибки
  + Неуспешный код ошибки (чем регламентируется?)
  + Не возвращается обязательное поле
  + Возвращается лишнее поле

Порядок выполнения лабораторной работы:

При помощи Postman необходимо протестировать REST API мобильного клиента портала befutsal.ru. Для этого необходимо настроить доступ и выполнить несколько запросов на основе соответствующей спецификации.

1. Зайти [http://editor.swagger.io/#](http://editor.swagger.io/)
2. Загрузить/скопировать содержимое <http://befutsal.ru/m/api>

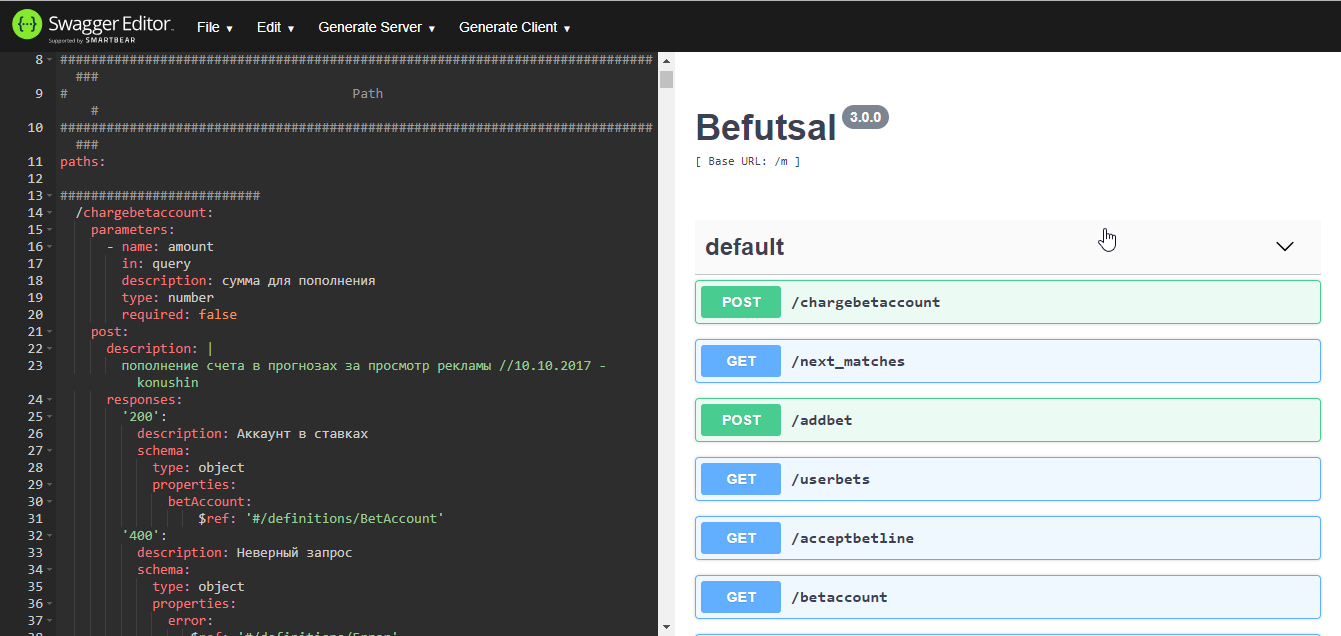


Рисунок 3 – Отображение API в среде swagger

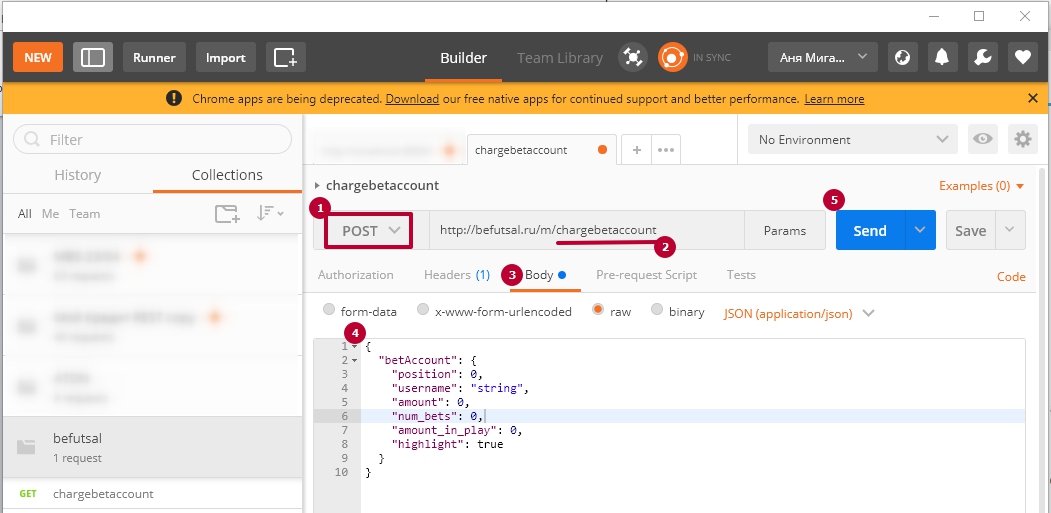
1. Открыть postman (создать проект и добавить/настроить новый запрос)
2. В блоке Params указать « <http://befutsal.ru/m/>»

Рисунок 4 - Работа в postman

1. Сохранение изменений – ctrl + S
2. По аналогии настроить другие запросы

# **Задание**

1. Изучить спецификацию API портала
2. Составить тест кейсы или чек листы по 5 сценариям

* Зарегистрироваться, вспомнить пароль, войти с новым паролем.
* Зарегистрироваться, получить список всех команд, добавить в избранное 2 команды.
* Войти, получить список избранных команд, удалить из избранного 1 команду.
* Войти, получить список всех команд, получить список предстоящих игр для любой команды.
* Войти, получить список всех команд, получить состав любой команды.
* Войти, получить список всех команд, получить турнирную таблицу любой команды.
* Войти, получить список всех команд, получить результаты игр любой команды.
* Войти, получить информацию о контактах федераций.
* Придумать свой сценарий.

1. Подготовить тестовые данные (по необходимости)
2. Провести тестирование
3. Проанализировать результат

# **Содержание отчета**

* 1. Титульный лист
  2. Цель
  3. Вариант задания
  4. Ход работы
* Чек лист
* Примеры запроса по заданию
* Примеры ответа по заданию
* Результаты тестирования
  + 1. Вывод

# **Вопросы для подготовки**

* 1. Какие этапы проведения тестирования REST API?
  2. Какой минимальный список тестов, которые необходимо провести?
  3. Надо ли тестировать GUI во время проверки REST API?
  4. Какие проблемы могут возникнуть проблемы при тестировании REST API?
  5. Пример формата передачи данных
  6. В чем отличие запроса GET от POST
  7. Последствия некорректной работы API
  8. Что означает минимальный запрос
  9. Что означает максимальный запрос
  10. Особенности тестирования API
  11. Чем регламентируется набор параметров в запросе/ответе

1. Источник: [Электронный ресурс] http://www.kickstartpros.com/ [↑](#footnote-ref-1)
2. Источник: [Электронный ресурс] http://crypto.pp.ua/2011/02/ [↑](#footnote-ref-2)