Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Петрозаводский государственный университет
Математический факультет
Кафедра информатики и математического обеспечения

Отчет по дисциплине «Верификация ПО»

Тестирование геоинформационного сервиса «GeTS»

Выполнил:
студент 6 курса группы 22609 Н. В. Давыдовский

_____
подпись

Преподаватель:
к.ф.-м.н., доцент К. А. Кулаков
Итоговая оценка: _____

_____
подпись

Петрозаводск

2014

Содержание

1 Объект тестирования

1.1 Описание системы

Система представляет из себя веб-сервис промежуточного уровня для платформы geo2tag, где geo2tag будет использоваться в качестве хранилища геоданных. Разрабатываемый сервис планируется использовать в приложениях, ориентированных на е-туризм.

Основные цели проекта:

1. реализация возможности разбиения точек на тематические категории (кафе, кинотеатры, магазины и т.д.);
2. реализация возможности построения туристических маршрутов и их хранения в системе;
3. поддержка получения геоданных, находящихся внутри некоторой области (полигоне).

1.2 Основные функции

Ниже представлен список основных функций системы.

- Получение точек по координатам и радиусу и/или категории.
- Поддержка туристических маршрутов.
- Поддержка категорий точек.
- Взаимодействие с клиентами посредством API.
- Поддержка публичных запросов (без авторизации).
- Авторизация с использованием OAuth2.
- Разделение аккаунтов на администраторские и обычные.
- Механизмы поддержки туроператоров.
- Использование языка разметки KML в качестве формата ввода/вывода геоданных.

1.3 Архитектура системы

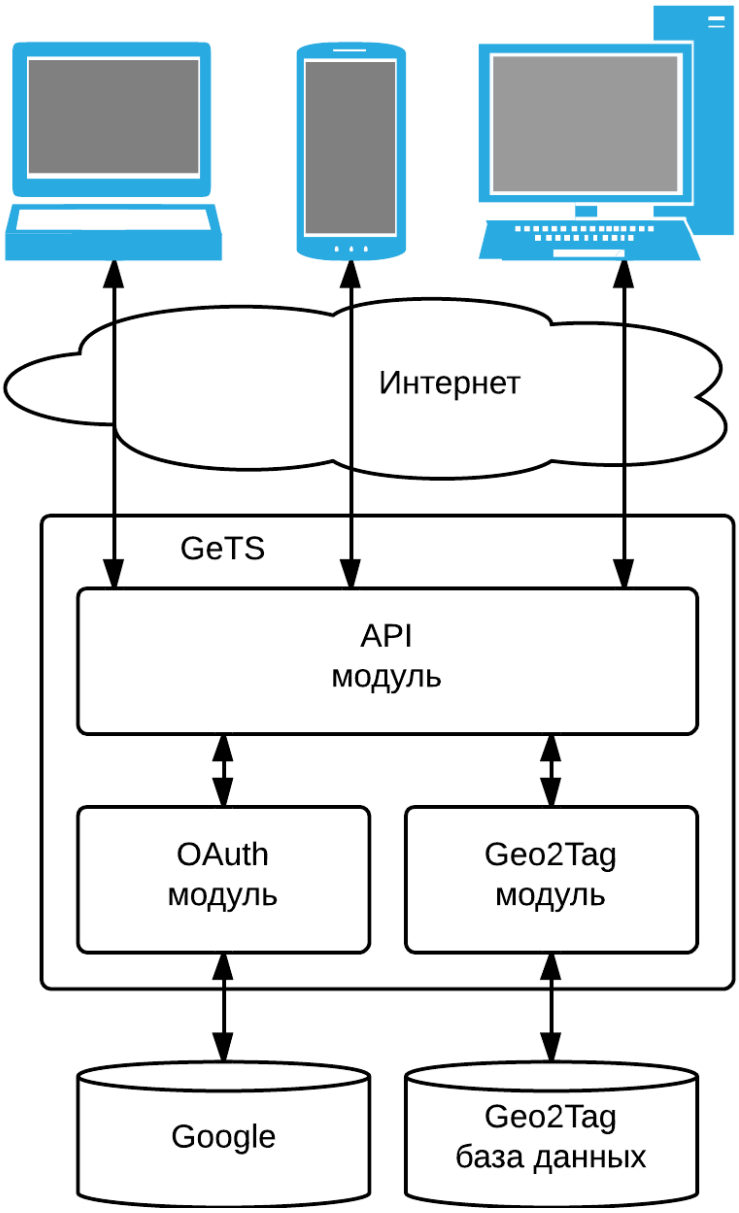


Рис. 1: Высокоуровневая архитектура

- Geo2Tag - платформа для хранения геоданных
- Geo2Tag e-Tourism Service (GeTS) - разрабатываемый сервис
- Google - предоставляет платформу для авторизации пользователей через OAuth 2.0
- Интернет - среда передачи данных

1.4 Подсистемы

1.4.1 Схема взаимодействия подсистем

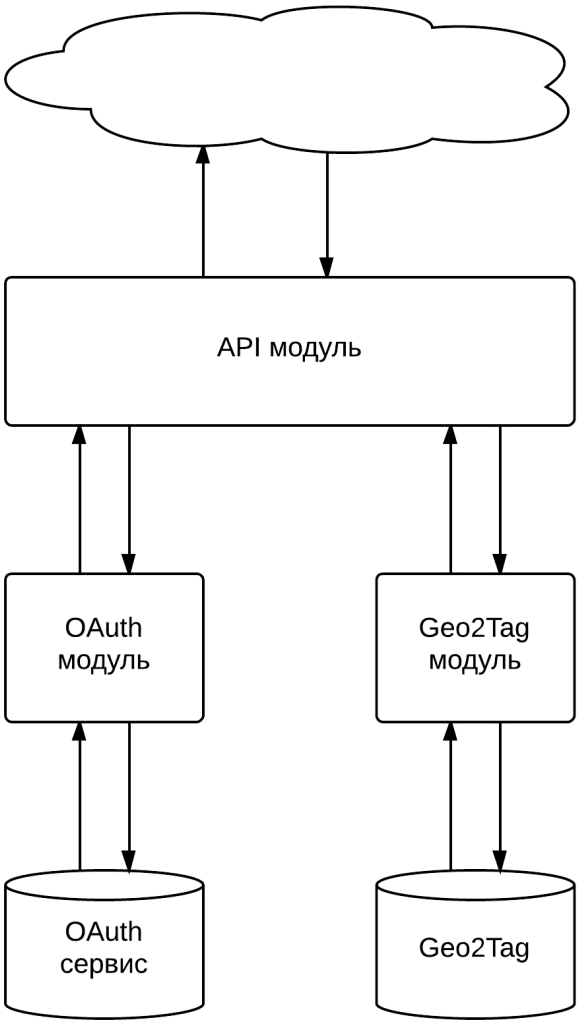


Рис. 2: Проект подсистем

1.4.2 Подсистемы

Далее рассматриваются функции, который выполняют отдельные модули (подсистемы).

- **API модуль** - выполняет функции обработки запросов и составления ответов для клиентов Состоит из набора PHP сценариев, каждый сценарий соответствует определенному API запросу. Будет подвержен тестированию, так как является основным модулем, обеспечивающим все функциональные требования системы.

- **OAuth модуль** - выполняет функцию авторизации, с использованием сторонних сервисов, по технологии Open Auth 2.0. Будет подвержен тестированию, так как обеспечивает авторизацию пользователей в системе.

- **Geo2Tag модуль** - выполняет функцию обращения к платформе Geo2Tag. Не будет подвержен тестированию, так как обеспечивает только дополнительную функциональность.

1.4.3 Взаимодействие подсистем

Далее рассматривается характер взаимодействия между отдельными модулями.

- **Взаимодействие API и OAuth модулей** - происходит обмен авторизационными данными. Клиент через API метод передает данные для авторизации на сервисе, OAuth модуль, получив эти данные, проходит авторизацию на сервисе и в ответ, через API модуль, отправляет клиенту auth token. Передача происходит через внутренние структуры данных.

- **Взаимодействие OAuth модуля и OAuth сервиса** - OAuth модуль перенаправляет, полученные от API модуля данные пользователя, на сервис. В ответ сервис отсылает данные о статусе авторизации. Передача происходит по протоколу HTTP.

- **Взаимодействие API и Geo2Tag модулей** - происходит обмен геоданными. Клент делает запрос на выборку геоданных, через API метод. API модуль разбирает запрос и передает данные в виде внутренней структуры модулю Geo2Tag. Модуль Geo2Tag делает запрос к Geo2Tag и передает ответ к API модулю. Передача происходит через внутренние структуры данных.

- **Взаимодействие Geo2Tag модуля и платформы Geo2Tag** - Geo2Tag модуль формирует запрос к платформе Geo2Tag и далее перенаправляет ответ API модулю. Передача происходит по протоколу HTTP.

1.5 Список API методов

Ниже приведен список API методов для взаимодействия с системой. Все указанные методы реализованы в модуле API.

- **addPoint** - добавление точки в систему.

- **loadPoints** - загрузка точек.

- **updatePoint** - редактирование точки.

- **deletePoint** - удаление точки.

- **loadTracks** - загрузка маршрутов.
- **createTrack** - создание/редактирование маршрута.
- **removeTrack** - удаление маршрута.
- **userLogin** - авторизация.

1.6 Список функций OAuth модуля

Далее приведен список функций входящих в OAuth модуль.

- **boolean auth_is_public_token($token)** - проверка является ли токен публичным.
- **string auth_extract_sub_token($token)** - извлечение саб-токена.
- **string auth_get_geo2tag_login(void)** - получение geo2tag логина, установленного в текущей сессии.
- **void auth_set_token($token)** - установка заданного токена в сессию.
- **string auth_get_google_token(void)** - получение приватного токена, установленного в текущей сессии.

1.7 Основные структуры данных

- **Публичный токен** - строка, содержащая токен для аутентификации на сервисе. Первые два символа строки должны быть "p:". Недоступен для клиента. Пример: **p:1/BiX9fk-W8nOtgyf6SDntmjG6RdC2qugd6DNEK6Sw-4I**.
- **Приватный токен** - строка, содержащая токен для аутентификации на сервисе. Первые два символа строки должны быть "g:". Передается клиенту. Пример: **g:1/BiX9fk-W8nOtgyf6SDntmjG6RdC2qugd6DNEK6Sw-4I**.
- **Саб-токен** - строка, содержащая часть токена без первый двух символов.
- **Запрос к системе** - представляет из себя XML файл со следующей структурой:

```
<request>
        <params>
                ...
        </params>
</request>
```

, где вместо «...» подставляются специфичные для конкретного запроса XML-теги.

- **Ответ системы** - представляет из себя XML файл со следующей структурой:

```
<response>
        <status>
                <code>...</code>
                <message>...</message>
        </status>
        <content>
                ...
        </content>
</response>
```

, где <**code**/> - содержит код статуса запроса, <**message**/> - содержит дополнительные сведения о запросе (коды и текст сообщений приведен ниже), <**content**/> - содержит данные, запрашиваемые пользователем (содержание специфично для каждого конкретного запроса).

1.7.1 Коды статуса запроса

Коды статуса запроса:

- **0** - запрос выполнен успешно;
- **1** - произошла ошибка при выполнении запроса;
- **2** - требуется перенаправление запроса.

2 План тестирования

2.1 Процедура тестирования

Основные положения процедуры проведения тестирования:

1. в рамках курса «Верификация ПО», тестирование будет проведено один раз, то есть без повторного проведения после исправления найденных ошибок;
2. процедура тестирования будет остановлена, только в случае нахождения блокирующих ошибок;
3. тестирование будет проводится автором данного отчета, без привлечения других лиц;

4. процедура тестирования будет признана выполненной успешно, если в ходе проведения не будет найдено блокирующих или критических ошибок.

2.2 Блочное тестирование

Первый вид тестирования, которому будет подвержена система, будет блочное тестирование. Данный вид тестирования будет применен к некоторым функциям из модулей API и OAuth. Список функций для тестирования.

- **Модуль API:**
    1. addPoint;
    2. loadPoints;
    3. loadTracks;
    4. loadTrack;
    5. userLogin.
- **Модуль OAuth:**
    1. auth_is_public_token;
    2. auth_extract_sub_token.
    3. auth_get_geo2tag_login;
    4. auth_set_token;
    5. auth_get_google_token.

Детальное описание функций можно найти в разделе 1.

2.3 Интеграционное тестирование

Второй этап - интеграционное тестирование. Будет проверяться интеграция модулей API и OAuth, протестированных блочно, на предыдущем этапе. Интегрироваться модули будут по принципу использования функций из OAuth модуля в функциях модуля API.

2.4 Аттестационное тестирование

Аттестация системы будет производится по следующим высокоуровневым функциям:

1. Авторизация в системе через сервис Google OAuth2.0.

2. Получение выборки точек по уровню доступа:

   - публичные точки;
   - приватные точки.

3. Получение выборки точек по фильтрам:

   - фильтр по категории;
   - фильтр по географическим координатам и радиусу;
   - фильтр по географические координатам, радиусу и категории.

4. Получение выборки маршрутов по уровню доступа:

   - публичные маршруты;
   - приватные маршруты.

5. Получение выборки маршрутов по фильтрам:

   - без фильтров;
   - фильтр по категории.

6. Получение маршрута по идентификатору.

2.5 Нагрузочное тестирование

Третий этап - нагрузочное тестирование. Данный вид тестирования будет проводится на системе, развернутой на web-сервере.

Для того, чтобы провести нагрузочные тесты нужно,определить модель нагрузки.

Модель нагрузки:

1. Ответ от системы, для любого запроса, не должен превышать 2-х минут.

Тестируемые функции:

1. addPoint;
2. loadPoints;
3. loadTracks;
4. loadTrack.

3 Инструменты для тестирования

3.1 Блочное тестирование

Так как весь исходный код модулей написан на языке PHP, то для блочного тестирования будет использоваться инструмент PHPUnit в связке NetBeans IDE.

3.1.1 Пример реализации блочного теста

Для того, чтобы реализовать блочный тест PHPUnit с помощью NetBeans IDE нужно, в заголовочном комментарии к тестируемой функции добавить ключевое слово @asset и параметры теста.

Рассмотрим пример для функции **auth_is_public_token**.

```
<?php
/**
 *
 * @assert ('p:some_string') == true
 * @param string $gets_token
 * @return boolean
 */
public static function auth_is_public_token($gets_token) {
        return substr_compare($gets_token, 'p:', 0, 2) === 0;
}
?>
```

В данном случае функция будет тестироваться с входным параметром **p:some_string**, а ожидаемый результат выполнения функции будет **true**.

Далее, нужно запустить генерацию тестов на основе исходно кода. Для написанного выше описания теста сгенерируется следующий блочный тест PHPUnit.

```
<?php
class Auth_Module_TestingTest extends PHPUnit_Framework_TestCase {

    protected $object;

    /**
     * Sets up the fixture, for example, opens a network connection.
     * This method is called before a test is executed.
     */
    protected function setUp() {
        $this->object = new Auth_Module_Testing;
    }
```

```
    /**
     * Generated from @assert ('p:some_string') == true.
     *
     * @covers Auth_Module_Testing::auth_is_public_token
     */
    public function testAuth_is_public_token() {
        $this->assertTrue(
                Auth_Module_Testing::auth_is_public_token(
                        'p:some_string'
                    )
        );
    }
}
?>
```

3.2 Нагрузочное тестирование

Для нагрузочных тестов будет использоваться инструмент Apache JMeter.

3.2.1 Пример реализации нагрузочного теста

Для того, чтобы создать нагрузочный тест в Apache JMeter нужно, создать **HTTP Request** с необходимыми параметрами и, сопутствующими запросу, данными.

Пример для функции **addPoint**.

4 Детальный план тестов

4.1 Блочные тесты

Функция: boolean auth_is_public_token(string: $token)	
Название теста:	Auth-Unit-Test-1
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность определения является ли данный токен публичным

Входные данные:	Строка – публичный токен
Ожидаемый результат:	Функция возвращает значение true
Название теста:	Auth-Unit-Test-2
Тип теста:	Блочный, негативный
Описание:	Тест проверяет правильность определения является ли данный токен публичным
Входные данные:	Строка – приватный токен
Ожидаемый результат:	Функция возвращает значение false
Название теста:	Auth-Unit-Test-3
Тип теста:	Блочный, краевой
Описание:	Тест проверяет правильность определения является ли данный токен публичным
Входные данные:	Случайное число
Ожидаемый результат:	Функция возвращает значение false
Название теста:	Auth-Unit-Test-4
Тип теста:	Блочный, краевой
Описание:	Тест проверяет правильность определения является ли данный токен публичным
Входные данные:	Значение NULL
Ожидаемый результат:	Функция возвращает значение false
Функция: string auth_extract_sub_token($token)	
Название теста:	Auth-Unit-Test-5
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность извлечения саб-токена
Входные данные:	Строка – публичный токен
Ожидаемый результат:	Функция возвращает саб-токен

Название теста:	Auth-Unit-Test-6
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность извлечения саб-токена
Входные данные:	Строка – приватный токен
Ожидаемый результат:	Функция возвращает саб-токен
Название теста:	Auth-Unit-Test-7
Тип теста:	Блочный, краевой
Описание:	Тест проверяет правильность извлечения саб-токена
Входные данные:	Пустая строка
Ожидаемый результат:	Функция возвращает значение false
Название теста:	Auth-Unit-Test-8
Тип теста:	Блочный, краевой
Описание:	Тест проверяет правильность извлечения саб-токена
Входные данные:	Значение NULL
Ожидаемый результат:	Функция возвращает значение false
Функция: string auth_get_geo2tag_login()	
Название теста:	Auth-Unit-Test-9
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность получения логина geo2tag, установленного для текущей сессии
Входные данные:	Нет
Ожидаемый результат:	Функция возвращает логин
Функция: void auth_set_token($token)	
Название теста:	Auth-Unit-Test-10
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность установки токена в сессию

Входные данные:	Строка – публичный токен
Ожидаемый результат:	Функция ничего не возвращает
Название теста:	Auth-Unit-Test-11
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность установки токена в сессию
Входные данные:	Строка – приватный токен
Ожидаемый результат:	Функция ничего не возвращает
Функция: string auth_get_google_token()	
Название теста:	Auth-Unit-Test-12
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность получения приватного, установленного для текущей сессии
Входные данные:	Нет
Ожидаемый результат:	Функция возвращает приватный токен
Функция: loadPoints	
Название теста:	API-Unit-Test-1
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность выборки точек
Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. latitude 2. longitude 3. radius
Ожидаемый результат:	Функция возвращает выборку точек, соответствующую заданным параметрам, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор точек в KML формате

Название теста:	API-Unit-Test-2
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность выборки точек
Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. latitude 2. longitude 3. radius 4. category_id
Ожидаемый результат:	Функция возвращает выборку точек, соответствующую заданным параметрам, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор точек в KML формате
Название теста:	API-Unit-Test-3
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность выборки точек
Входные данные:	XML файл "Запрос к системе"с дополнительными тегом: 1. category_id
Ожидаемый результат:	Функция возвращает выборку точек, соответствующую заданным параметрам, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор точек в KML формате
Название теста:	API-Unit-Test-4
Тип теста:	Блочный, негативный
Описание:	Тест проверяет корректность реакции функции на неправильный набор параметров
Входные данные:	XML файл "Запрос к системе"без дополнительных тегов

Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, not valid input XML document»
Название теста:	API-Unit-Test-5
Тип теста:	Блочный, негативный
Описание:	Тест проверяет корректность реакции функции на вызов без входного файла
Входные данные:	Отсутствуют
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, no input XML file»
Функция: addPoint	
Название теста:	API-Unit-Test-6
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность добавления точки
Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. title 2. description 3. link 4. latitude 5. longitude
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 0, сообщение равно «success»
Название теста:	API-Unit-Test-7
Тип теста:	Блочный, негативный
Описание:	Тест проверяет обработку невалидного XML документа
Входные данные:	XML файл "Запрос к системе"без дополнительных тегов

Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, not valid input XML document»
Название теста:	API-Unit-Test-8
Тип теста:	Блочный, негативный
Описание:	Тест проверяет обработку отсутствия XML документа
Входные данные:	Отсутствуют
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, no input XML file»
Функция: loadTracks	
Название теста:	API-Unit-Test-9
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность получения маршрутов
Входные данные:	XML файл "Запрос к системе"без дополнительных тегов
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы» со всеми публичными маршрутами, где код ответа равен 0, сообщение равно «success»
Название теста:	API-Unit-Test-10
Тип теста:	Блочный, негативный
Описание:	Тест проверяет обработку отсутствия XML документа
Входные данные:	Отсутствуют
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, no input XML file»
Функция: loadTrack	
Название теста:	API-Unit-Test-11
Тип теста:	Блочный, общий

Описание:	Тест проверяет правильность получения маршрута
Входные данные:	XML файл "Запрос к системе"с дополнительным тегом: 1. name
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы» со всеми точками маршрута, где код ответа равен 0, сообщение равно «success»
Название теста:	API-Unit-Test-12
Тип теста:	Блочный, негативный
Описание:	Тест проверяет обработку невалидного XML документа
Входные данные:	XML файл "Запрос к системе"без дополнительных тегов
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, not valid input XML»
Название теста:	API-Unit-Test-13
Тип теста:	Блочный, негативный
Описание:	Тест проверяет обработку отсутствия XML документа
Входные данные:	Отсутствуют
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, no input XML file»
Функция: userLogin	
Название теста:	API-Unit-Test-14
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность первого этапа авторизации
Входные данные:	XML файл "Запрос к системе"без дополнительных тегов
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы» с идентификатором и ссылкой, где код ответа равен 2, сообщение равно «redirect»

Название теста:	API-Unit-Test-15
Тип теста:	Блочный, общий
Описание:	Тест проверяет правильность второго этапа авторизации
Входные данные:	XML файл "Запрос к системе"с дополнительным тегом: 1. id
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы» с токеном, где код ответа равен 0, сообщение равно «success»
Название теста:	API-Unit-Test-16
Тип теста:	Блочный, негативный
Описание:	Тест проверяет обработку отсутствия XML документа
Входные данные:	Отсутствуют
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 1, сообщение равно «Error, no input XML file»

4.2 Интеграционные тесты

Функция: loadPoints, взаимодействие с auth_is_public_token и auth_set_token	
Название теста:	Auth-API-Integrac-Test-1
Описание:	Тест проверяет правильность выборки точек
Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. auth_token 2. latitude 3. longitude 4. radius
Ожидаемый результат:	Функция возвращает выборку точек, соответствующую заданным параметрам, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор точек в KML формате

Функция: addPoint, взаимодействие с auth_is_public_token и auth_set_token	
Название теста:	Auth-API-Integrac-Test-2
Описание:	Тест проверяет добавления точки
Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. auth_token 2. title 3. description 4. link 5. latitude 6. longitude
Ожидаемый результат:	Функция возвращает XML файл «Ответ системы», где код ответа равен 0, сообщение равно «success»
Функция: loadTracks, взаимодействие с auth_is_public_token и auth_set_token	
Название теста:	Auth-API-Integrac-Test-3
Описание:	Тест проверяет правильность выборки маршрутов
Входные данные:	XML файл "Запрос к системе"с дополнительным тегом: 1. auth_token
Ожидаемый результат:	Функция возвращает выборку маршрутов, соответствующую заданным параметрам, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор маршрутов
Функция: loadTrack, взаимодействие с auth_is_public_token и auth_set_token	
Название теста:	Auth-API-Integrac-Test-4
Описание:	Тест проверяет правильность выборки точек маршрута

Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. auth_token 2. name
Ожидаемый результат:	Функция возвращает выборку точек, соответствующую заданным параметрам, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор точек в KML формате
Функция: userLogin, взаимодействие с auth_is_public_token и auth_set_token	
Название теста:	Auth-API-Integrac-Test-5
Описание:	Тест проверяет правильность авторизации
Входные данные:	XML файл "Запрос к системе"с дополнительными тегами: 1. id
Ожидаемый результат:	Функция возвращает токен, в виде XML файла «Ответ системы», где код ответа равен 0, сообщение равно «success», а тег content содержит набор токен

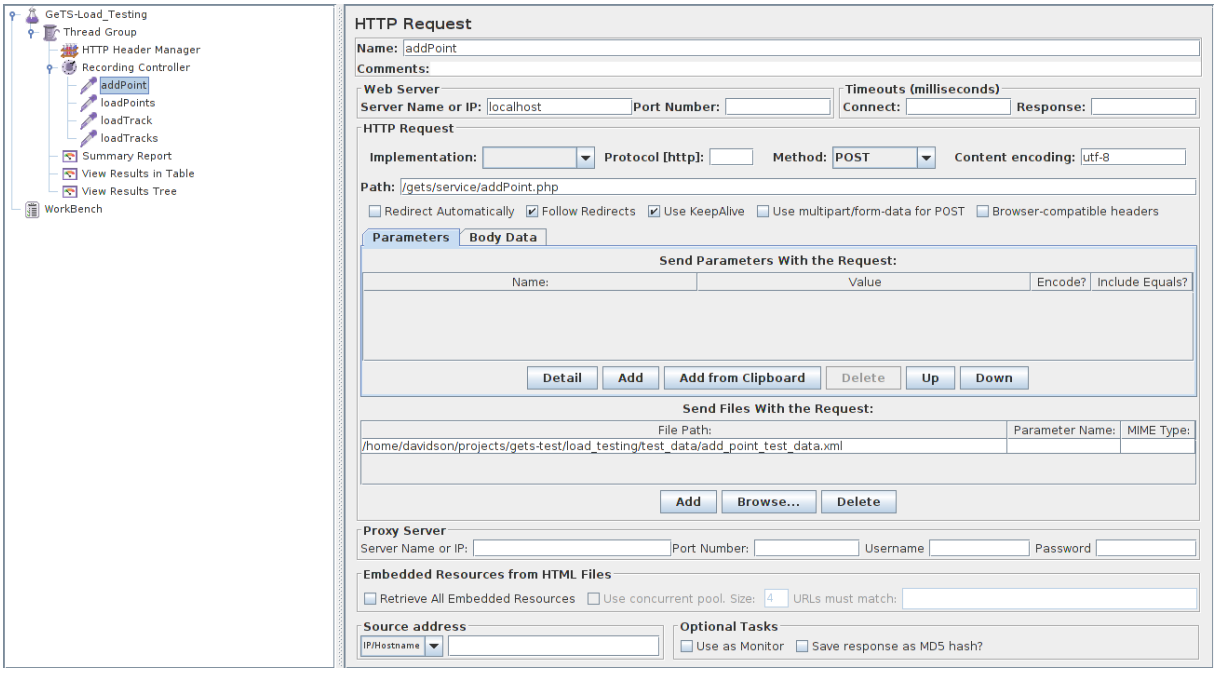
4.3 Аттестационное тестирование

Обозначения:

- «Клиент» - любое программное обеспечение, вступающее во взаимодействие с системой, по средствам методов API.
- Набор методов API представляет из себя внешний интерфейс системы.

4.3.1 Авторизация в системе через сервис Google OAuth2.0

1. «Клиент» отправляет пустой «Запрос системе» методу **userLogin**.
2. Метод генерирует идентификатор и ссылку для авторизации и отсылает их в ответ «Клиенту».
3. «Клиент», пользуясь полученной ссылкой, проходит авторизацию в Google.



4. «Клиент» делает повторный «Запрос системе» методу **userLogin**, с полученным ранее идентификатором.
5. Метод возвращает «Клиенту» приватный токен.

4.3.2 Получение выборки точек по уровню доступа

Публичные точки:

1. «Клиент» отправляет «Запрос системе» методу **loadPoints**, без тега auth_token и любыми фильтрами.
2. Метод возвращает «Клиенту» выборку всех публичных точек, соответствующих заданным фильтрам.

Приватные точки:

1. «Клиент» отправляет «Запрос системе» методу **loadPoints**, с тегом auth_token и любыми фильтрами.
2. Метод возвращает «Клиенту» выборку всех публичных и приватных точек, соответствующих заданным фильтрам.

4.3.3 Получение выборки точек по фильтрам

Фильтр по категории:

1. «Клиент» отправляет «Запрос системе» методу **loadPoints**, с/без тега auth_token и тегом category_id.

2. Метод возвращает «Клиенту» выборку всех публичных/приватных точек, соответствующих заданной категории.

Фильтр по географическим координатам и радиусу:

1. «Клиент» отправляет «Запрос системе» методу **loadPoints**, с/без тега auth_token и тегами latitude, longitude и radius.
2. Метод возвращает «Клиенту» выборку всех публичных/приватных точек, соответствующих заданной географической области.

Фильтр по географические координатам, радиусу и категории:

1. «Клиент» отправляет «Запрос системе» методу **loadPoints**, с/без тега auth_token и тегами latitude, longitude, radius и category_id.
2. Метод возвращает «Клиенту» выборку всех публичных/приватных точек, соответствующих заданной географической области и категории.

4.3.4 Получение выборки маршрутов по уровню доступа

Публичные маршруты:

1. «Клиент» отправляет «Запрос системе» методу **loadTracks**, без тега auth_token.
2. Метод возвращает «Клиенту» выборку всех публичных маршрутов.

Приватные маршруты:

1. «Клиент» отправляет «Запрос системе» методу **loadTracks**, с тегом auth_token.
2. Метод возвращает «Клиенту» выборку всех публичных и приватных маршрутов.

4.3.5 Получение выборки маршрутов по фильтрам

Без фильтра:

1. «Клиент» отправляет «Запрос системе» методу **loadTracks**, с/без тега auth_token.
2. Метод возвращает «Клиенту» выборку всех публичных/приватных маршрутов.

Фильтр по категории:

1. «Клиент» отправляет «Запрос системе» методу **loadTracks**, с/без тега auth_token и тегом category_id.
2. Метод возвращает «Клиенту» выборку всех публичных/приватных маршрутов, соответствующих заданной категории.

4.3.6 Получение маршрута по идентификатору

1. «Клиент» отправляет «Запрос системе» методу **loadTrack**, с/без тега auth_token (в зависимости от трека) и тегом track_name.
2. Метод возвращает «Клиенту» все точки заданного маршрута.

4.4 Нагрузочные тесты

Сценарии:

1. Генерация 100 запросов к функции **loadPoint**. Содержимое запросов:

```
<request>
        <params>
                <auth_token>
                        g:1/B8ywqIdL4TT3ZUaVNqpXiiglQVF7q3B
                        47WZ6-sVEwS0
                </auth_token>
                <latitude>63.84</latitude>
                <longitude>34.45</longitude>
                <radius>63</radius>
        </params>
</request>
```

2. Генерация 100 запросов к функции **loadTracks**. Содержимое запросов:

```
<request>
        <params>
                <auth_token>
                        g:1/B8ywqIdL4TT3ZUaVNqpXiiglQVF7q3B
                        47WZ6-sVEwS0
                </auth_token>
        </params>
</request>
```

3. Генерация 100 запросов к функции **loadTrack**. Содержимое запросов:

```
<request>
        <params>
                <auth_token>
                        g:1/B8ywqIdL4TT3ZUaVNqpXiiglQVF7q3B
                        47WZ6-sVEwS0
                </auth_token>
                <name>
                        tr+admin+Track Load Test addPoint
                        method+en_US
                </name>
        </params>
</request>
```

4. Генерация 100 запросов к функции **addPoint**. Содержимое запросов:

```
<request>
        <params>
                <auth_token>
                        g:1/B8ywqIdL4TT3ZUaVNqpXiiglQVF7q3B
                        47WZ6-sVEwS0
                </auth_token>
                <channel>
                        tr+admin+Track Load Test addPoint
                        method+en_US
                </channel>
                <title>
                        Point Load Test addPoint method
                </title>
                <description>
                        You should see the HTTP requests
                        that were recorded, depending on
                        which URL Patterns you have included
                        and excluded
                </description>
```

```
                <link>
                        http://localhost/gets/service/addPoint.php
                </link>
                <latitude>63.84</latitude>
                <longitude>34.45</longitude>
                <extended_data>
                        <radius>63</radius>
                        <raiting>5</raiting>
                </extended_data>
        </params>
</request>
```

5 Результаты тестирования

5.1 Блочное тестирование

Метод	**Количество тестов**	**Количество ошибок (Идентификаторы ошибок)**
auth_is_public_token	4	1 (GeTS-OAuth-bug-1)
auth_extract_sub_token	4	0
auth_get_geo2tag_login	1	0
auth_set_token	2	0
auth_get_google_token	1	0
loadPoints	5	0
addPoint	3	0
loadTracks	2	0
loadTrack	3	0
userLogin	3	0
Всего:	28	1

Таблица 3: Показатели блочного тестирования

5.2 Интеграционное тестирование

Метод	**Количество тестов**	**Количество ошибок**
loadPoints	1	0
addPoint	1	0
loadTracks	1	0
loadTrack	1	0
userLogin	1	0
Всего:	5	0

Таблица 4: Показатели интеграционного тестирования

5.3 Аттестационное тестирование

Все аттестационные тесты были пройдены успешно.

5.4 Нагрузочное тестирование

Метод	**Кол-во запросов**	**Среднее время ответа (мс)**	**Мин. время ответа (мс)**	**Макс. время ответа (мс)**	**Среднекв. откл.**	**Ошибка (%)**
addPoint	100	1587	941	1917	269.85	0.0
loadPoints	100	1084	665	1503	247.35	0.0
loadTrack	100	1266	697	1814	325.92	0.0
loadTracks	100	1118	460	1734	357.71	0.0
Total:	400	1264	460	1917	362.77	0.0

Таблица 5: Показатели нагрузочного тестирования

5.5 Пример ошибки

В данном разделе будет представлен пример ошибки, которая была найдена в ходе тестирования.

Ниже представлена реализация функции **auth_is_public_token**.

```
<?php
public static function auth_is_public_token($gets_token) {
        return substr_compare($gets_token, 'p:', 0, 2) === 0;
}
?>
```

Если в качестве аргумента, данной функции передать значение **NULL**, то функция вернет ошибку «Error: substr_compare(): The start position cannot exceed initial string length». Чтобы избавиться от подобной ошибки можно, например, сделать проверку входного аргумента на **NULL**. Если входной аргумент равен **NULL**, то будем возвращать **false**.

В итоге получим следующую исправленную реализацию.

```
<?php
public static function auth_is_public_token($gets_token) {
        return $gets_token
                ? substr_compare($gets_token, 'p:', 0, 2) === 0
                : false;
```

```
}
?>
```

5.5.1 Отчеты об ошибке

Идентификатор	GeTS-OAuth-bug-1
Проект	GeTS
Номер версии	1.0.0
Компонент приложения	Модуль OAuth, функция auth_is_public_token
Короткое описание	Несоответствие реального и ожидаемого результатов
Серьезность	S5 (Тривиальная)
Приоритет дефекта	P3 (Низкий)
Автор	Давыдовский Никита
Назначен на	Ивашов Кирилл
Окружение	• ОС: Ubuntu 14.04 64bit • PHP: 5.5.3
Шаги воспроизведения	1. Вызов функции с входным параметром NULL
Фактический результат	Ошибка PHP: «Error: substr_compare(): The start position cannot exceed initial string length»
Ожидаемый результат	Значение: false

Таблица 6: Отчет об ошибке

Градация серьезности ошибки:

1. S1 Блокирующая (Blocker)
2. S2 Критическая (Critical)
3. S3 Значительная (Major)
4. S4 Незначительная (Minor)
5. S5 Тривиальная (Trivial)

Приоритет ошибки:

1. P1 Высокий (High)
2. P2 Средний (Medium)
3. P3 Низкий (Low)

5.6 Покрытие кода

Для вычисления покрытия кода тестами, будет использоваться критерий - **покрытие функций**. Вычисление покрытия производится по формуле: $T_{cov} = (\frac{F_{cov}}{F_{total}}) \times 100\%$, где

- T_{cov} - общее значение покрытия модуля/проекта;
- F_{cov} - количество покрытых функций модуля/проекта;
- F_{total} - всего функций в модуле/проекте.

Модуль	**Покрытие функций** $(\frac{F_{cov}}{F_{total}})$	**Процент покрытия** (T_{cov})
API	4/8	50 %
OAuth	4/15	26.6 %
Geo2Tag Scripts	0/16	0 %
Total	**8/39**	**20.51 %**

Таблица 7: Покрытие кода