Петрозаводский государственный университет
Кафедра информатики и математического обеспечения

Отчёт по курсу
«Тестирование и верификация ПО»

Тестирование мобильного приложения GeTS Supplement

Выполнил:
студент группы 22608
А. А. Трошков

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

Петрозаводск
2014

Оглавление

Объект исследования

Объектом тестирования является мобильное геоинформационное приложение для платформы Android (рабочее название — GeTS Supplement). Приложение позволяет взаимодействовать с интерфейсом веб-сервиса (API) и предоставляет возможность просмотра, добавления и удаления маркера (отметки) на карту. В качестве объекта тестирования выбрана часть приложения, составляющая набор базовых функций: получение данных с удаленного сервиса, сохранение, разбор, отображение, удаление данных, а также отправка создаваемых данных на удаленный сервер.

Открытый исходный код приложения доступен в репозитории. Язык программирования — Java (Android SDK).

Для тестирования используется следующее окружение: GeTS Supplement versionCode 2, Android Studio, Android SDK (API v21), Java SE 1.7, Ubuntu 14.04, LG Nexus 5, USB-кабель.

Задачи тестирования:

- Выявление ошибок для их дальнейшего исправления
- Проверка работы приложения в непредвиденных ситуациях
- Проверка степени готовности приложения.

Критерии остановки тестирования:

- Результаты тестирования удовлетворяют критериям качества продукта
- Новые ошибки и дефекты не обнаружились

Критериями возобновления тестирования:

- Обнаружение ошибок в работе приложения
- Выполнение действий по устранению ошибок в функционале компонентов
- Появление изменений в функционале компонентов

Архитектура и описание модулей

Приложение состоит из нескольких модулей (подсистем), взаимодействующих между собой (рис. 1).

`MapActivity` — главный модуль, осуществляющий управление остальными модулями приложения. С точки зрения пользовательского интерфейса, `MapActivity` представляет главный экран приложения, на котором расположены:

- карта с отмеченными на ней точками
- кнопка перехода к текущему местоположению пользователя
- кнопка запуска модуля `AddNewPointActivity`

Модуль `MapActivity` также осуществляет определение местоположения пользователя, используя доступные методы геолокации.

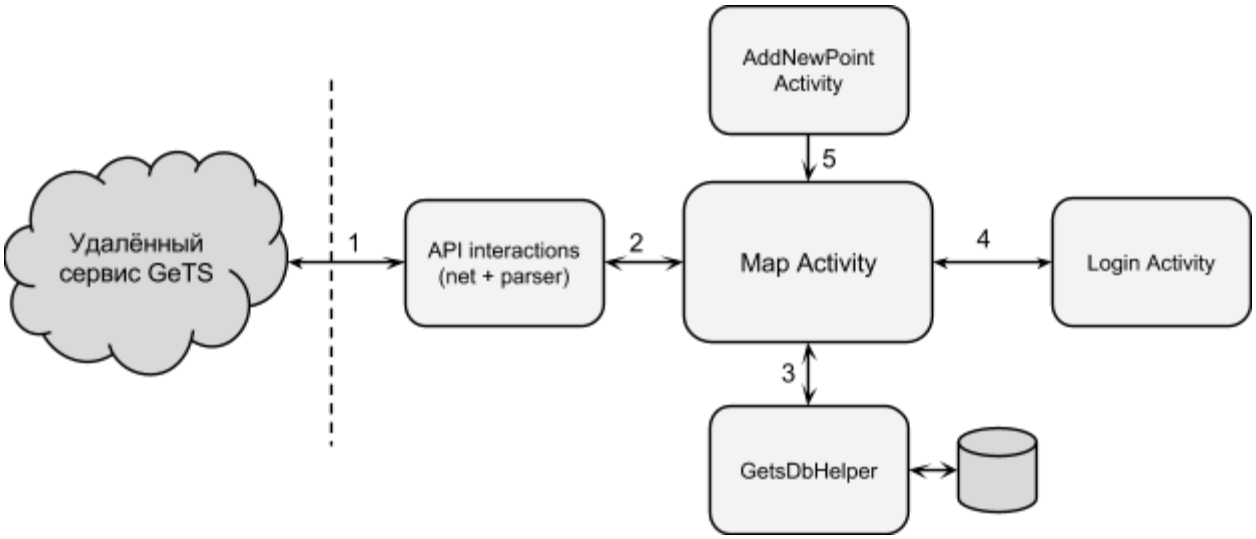

Рис. 1. Высокоуровневая архитектура приложения

API Interactions — набор классов, осуществляющих взаимодействие с удаленным сервером в формате, описанном в документации API сервера.

`GetsDbHelper` — модуль, обеспечивающий взаимодействие модулей приложения (в частности, модуля `MapActivity`) с внутренней базой данных. Класс является потомком класса `DBHelper` и предоставляет высокоуровневые методы добавления и извлечения данных из внутренней базы данных.

`LoginActivity` — модуль, осуществляющий пользовательскую авторизацию с целью получения права доступа к удаленному серверу.

`AddNewPointActivity` — модуль, позволяющий пользователю добавлять новую точку на карту. Модуль содержит:

- карту с отмеченным текущим положением пользователя
- кнопки изменения состояния карты: переход к текущему местоположению, изменение масштаба карты
- поля ввода названия точки и рейтинга
- кнопку подтверждения создания новой точки

Данный модуль занимается исключительно сбором пользовательских данных о создаваемой точке, а не взаимодействием с внутренней базой данных или удаленным сервером. После заполнения всех полей на данном экране и нажатия на кнопку создания новой кнопки, данные пользователя передаются в вызывающий модуль, т.е. в модуль `MapActivity`, откуда и происходит отображение данных на карте, сохранение в базе данных и отправка данных на удаленный сервер.

Модули `MapActivity`, `LoginActivity` и `AddNewPointActivity` содержат не только логику приложения, но и имеют представление в виде графического интерфейса пользователя.

На рис. 1 стрелками обозначены направления передачи данных между модулями. Цифры над стрелками показывают последовательность интеграционного тестирования модулей.

Описание внутренних структур данных

В некоторых методах используются структуры данных, созданные специально для приложения. Они лучшим образом подходят для представления данных, полученных и отправляемых на удаленный сервер.

- `PointsResponse` — представление ответа от удаленного сервера
  - `int code` — код ответа сервера. Ограничение: `code ≥ 0`
  - `String message` — сопровождающее ответ сервера сообщение. Ограничение: нет.

    - `ArrayList<Point> points` — массив точек `Point`. Ограничение: нет (может быть `null` или пустым).
- `Point` — представление одного объекта типа точка, полученного с удаленного сервера (географическая точка с дополнительным описанием)
    - `String access` — права доступа к точке. Недокументированное поле. Ограничения: нет (поле может быть пустым).
    - `int id` — уникальный идентификатор, соответствующий идентификатору объекта, хранящегося в базе данных. Ограничения: `id` ≥ 0
    - `double latitude` — географическая широта. Ограничения: [-90; 90]
    - `double longitude` — аналогично полю `latitude`
    - `String name` — название точки. Ограничение: непустое поле.
    - `float rating` — рейтинг точки. Ограничение: [0.0; 5.0] шаг 0,1.
    - `String time` — время в строчном представлении. Ограничение: нет.
    - `String url` — ссылка на веб-адрес. Ограничение: нет.
    - `String uuid` — уникальный идентификатор объекта на стороне сервера. Ограничение: нет (может быть `null`).
- `enum DatabaseType` — представление типа базы данных: внутренняя для хранения полученных от сервера точек или внутренняя для хранения точек, созданных пользователем, но не отправленных на сервер.
- `Category` — представление одного объекта типа категория, полученного с удаленного сервера (категории точек)
    - `String description` — текстовое описание категории. Ограничение: нет.
    - `int id` — аналогично полю `id` объектов `Point`.
    - `String name` — имя категории. Ограничение: не `null`.

    - `String urlIcon` — ссылка на веб-адрес с изображением. Ограничение: нет.
    - Обратите внимание: у объекта типа `Point` нет поля, обозначающего принадлежность к определенной категории. На данный момент удаленный сервер не включает такую информацию в свой ответ, что, по сути, делает работу всего, что связано с объектами типа `Category`, бесполезной.

Классы `Marker`, `LatLng` и `Location` предоставлены библиотекой MapBox Sdk или являются системными:
`com.mapboxsdk.overlay.Marker`,
`com.mapboxsdk.geometry.LatLng`,
`android.location.Location` соответственно.

Описание кодов ответа сервера

В теле ответа сервера содержатся текстовые данные в формате XML. В теге `status` содержатся коды и сообщения, описывающие характер ответа сервера:

- `code` — код ответа:
    - 0 — успешное завершение запроса
    - 1 — ошибка в результате проведения запроса
    - 2 — перенаправление
- `message` — текстовое описание кода ответа.

Накладываемые ограничения

- `radius > 0` — верхняя граница не задана, но логично утверждать, что при большом (99000 км) значении радиуса сервер будет возвращать больше точек, чем при небольшом (9 км) значении.
- `latitude`, `longitude` — географическая широта и долгота соответственно. Значения ограничиваются аналогично одноименным полям в структуре `Point`, [-99.0; 99.0].
- “длинная строка” — строка, содержащая более 10 тысяч символов
- “пустая строка” — строка, не содержащая символов, “”

План тестирования

Данный документ описывает набор блочных, интеграционных и аттестационных тестов.

Тестированию подлежат методы следующих вышеописанных классов: `MapActivity`, API Interactions, `GetsDbHelper`, `AddNewPoint Activity`. Тестирование не будет проводиться для методов класса `LoginActivity`, поскольку эти методы практически не содержат логики и используют стандартные механизмы системы для реализации описанного функционала. Тестирование не будет проводиться для системных методов классов `MapActivity` и `AddNewPointActivity` (методы `onResume(), onStart(), onPause()`), поскольку при реализации этих методов в приложении они не содержат дополнительной логики.

Далее подробно описаны классы и методы, подвергаемые блочному тестированию:

1. `API Interactions`
    a. `PointsGet` — получение данных с удаленного сервера по заданным параметрам (координаты и радиус (в км), в котором необходимо получить точки)
        i. `PointsResponse PointsGet(String token, double latitude, double longitude, int radius)`
    b. `PointsAdd` — отправка создаваемой точки на удаленный сервер
        i. `PointsResponse PointsAdd(String token, int categoryId, String title, float rating, double latitude, double longitude, long unixTime)`
2. `MapActivity`
    a. `void addMarker(Point point)` — добавление маркера, создаваемого из объекта класса Point, на карту
    b. `void addMarkerLowLevel(Marker marker)` — добавление маркера на карту

c. `void deleteMarker(Marker marker)` — удаление маркера с карты
d. `boolean isAuthorized()` — проверка статуса авторизации
e. `boolean isInternetConnectionAvailable()` — проверка доступности сетевого подключения
f. `void onActivityResult(int requestCode, int resultCode, Intent data)` — метод обработки возврата из другого экземпляра класса Activity
g. `void onLocationChanged(Location location)` — метод обновления данных о местоположении пользователя

3. `GetsDbHelper`
   a. `void addPoint(Point point)` — метод добавления нового объекта класса Point в базу данных
   b. `void addPoints(ArrayList<Point> points)` — метод добавления массива объектов класса Point в базу данных
   c. `ArrayList<Point> getPoints()` — метод получения всех сохраненных в базе данных точек
   d. `void addCategory(int id, String name, String description, String url)` — метод добавления нового оюъекта класса Category в базу данных
   e. `ArrayList<Category> getCategories()` — метод получения всех сохраненных в базе данных категорий
   f. `String getCategoryName(int categoryId)` — получение имени категории по её идентификатору

4. `AddNewPointActivity`
   a. `void addMarker(LatLnt position)` — добавление маркера на карту по координатам, сохраненным в объекте типа LatLnt
   b. `boolean isStringEmpty(String stringToCheck)` — проверка строки на пустоту.

Интеграционные тесты проверяют взаимодействие двух классов, `MapActivity` и `AddNewPointActivity`. Механизм взаимодействия экземпляров объектов этих двух классов показан на изображении. `MapActivity` запускает `AddNewPointActivity` с намерением получить результат. Пользователь заполняет поля ввода в `AddNewPointActivity` и нажимает на кнопку “ОК”. Введенная пользователем информация возвращается в `MapActivity` через метод `onActivityResult()`, принимающий на вход код результата (успех или нет), код запроса (аналогичный тому, который указывает `MapActivity` при обозначении намерения получить результат), а также данные, вложенные в экземпляр класса `Intent`.

Стресс-тесты проверяют то, как поведет себя приложения в случае обработки большого объема данных. Стресс-тесты проводятся для модуля получения данных с удаленного сервера (`PointsGet`), модуля добавления и извлечения данных из базы данных (`GetsDbHelper`), а также при добавлении точек на карту для проверки отсутствия “замораживаний” интерфейса пользователя (модуль `MapActivity`).

При стресс-тестировании возникает вопрос: что считать приемлемым количеством данных, которые обрабатывает приложение, а что считать большим. В контексте тестирования приложения GeTS Supplement, приемлемым считается следующее число обрабатываемых данных:

Информация об одной точке в среднем занимает 1Кб.

Планируемое число получаемых точек в одном запросе: 10 тысяч

Итого ок. 10 Мб данных для получения и синтаксического разбора. Приложение должно обрабатывать и бóльшие по величине файлы (объемы данных, выходящие за размеры приемлемых).

После загрузки и разбора данных о точках происходит их сохранение в базе данных с последующим отображением на карте. Эти два процесса также тестируются в разделе стресс-тестирования.

Детальный план тестов: блочные тесты

I. Взаимодействие с API

Метод `PointsResponse PointsGet(String token, double latitude, double longitude, int radius)`.

Проверка получения точек с сервера по заданным критериям.

№	Исходные данные	Ожидаемый результат
1	token — существующий, latitude и longitude, radius в пределах допустимых значений (latitude = -21.0, longitude = 21.0, radius = 5)	PointsResponse.code = 0 PointsResponse.points = массив точек типа Point
2	token = null	PointsResponse.code = 1 PointsResponse.points = null
3	Неверный token (“abc123”)	PointsResponse.code = 1 PointsResponse.points = null
4	Пустая строка token (“”)	PointsResponse.code = 1 PointsResponse.points = null
5	token с недопустимыми символами (@_%?&;\ /!$ \n \r \t)	PointsResponse.code = 1 PointsResponse.points = null
6	Большая длина строки token (token не существует)	PointsResponse.code = 1 PointsResponse.points = null
7, 8, 9, 10, 11, 12	Значения параметров выходят за рамки (latitude = {-999, 200}, longitude = {-999, 200}, radius = {-999, 0})	PointsResponse.code = 1 PointsResponse.points = null
13	Большое значение radius (2147483647)	PointsResponse.code = 1 PointsResponse.points = null
14	Нет сети	Пустой PointsResponse, PointsResponse.points = null

Метод `PointsResponse PointsAdd(String token, int categoryId, String title, float rating, double latitude, double longitude, long unixTime)`

Проверка отправки запроса на сервер для создания новой точки.

№	Исходные данные	Ожидаемый результат
1	token существует, categoryId = 14 (существующая категория), title=”Point name”, rating = 3.5, latitude = -21.0, longitude = 21.0, unixTime = 1419245683	PointsResponse.code = 0
2, 3	token = null title = null	PointsResponse.code = 1
4	Неверный token (“abc123”)	PointsResponse.code = 1
5,6	Пустые строки token и title (“”)	PointsResponse.code = 1
7	token с недопустимыми символами (@_%?&;\ /!$ \n \r \t)	PointsResponse.code = 1
8	title с недопустимыми символами	PointsResponse.code = 0
9	Большая длина строки token (token не существует)	PointsResponse.code = 1
10	Большая длина строки title	PointsResponse.code = 0
11, 12, 13, 14, 15, 16	Значения параметров выходят за рамки (latitude = {-999, 200}, longitude = {-999, 200}, rating = {-99.9, 99.0})	PointsResponse.code = 1
17	Неверное значение unixTime (-1000)	PointsResponse.code = 1
18, 19	categoryId со значением несуществующей категории (99999, -9999)	PointsResponse.code = 1
20	Нет сети	Пустой PointsResponse

II. MapActivity

Метод `void addMarkerLowLevel(Marker marker)`

Проверка добавления нового объекта типа `Marker` на карту.

№	Исходные данные	Ожидаемый результат
1	marker — существующий маркер с координатами latitude = -21.0, longitude = 21.0	Новый маркер отображается на карте в координатах -21.0; 21.0
2	marker = null	Ничего не происходит
3, 4	marker со значениями поля title, равными пустой строке (“”) и (null)	Маркер без имени отображается на карте
5	Добавление маркера при несуществующей карте	Ничего не происходит

Метод `void addMarker(Point point)`

Проверка добавления нового объекта типа `Marker` на карту.

№	Исходные данные	Ожидаемый результат
1	Структура point с заполненными полями, не выходящими за пределы допустимых значений (point.name = “name”, point.latitude = -21.0, point.longitude = 21.0)	Успешное добавление новой точки на карту
2, 3, 4, 5, 6, 7, 8	Структура с одним из параметров, выходящим за границы ограничений. latitude = {-200, 200}, longitude = {-200, 200}, rating = {-5.0, 6.0}	Ничего не происходит
9	null в поле name структуры	Успешное добавление новой точки без имени на карте
10	null в качестве значения point	Ничего не происходит
11	Добавление уже существующей точки	Успешное добавление новой точки на карте (две одинаковые точки на карте)
12	Большие по длине строки в структуре	Успешное добавление новой точки на карте
13	Добавление маркера при несуществующей карте	Ничего не происходит

Метод `void deleteMarker(Marker marker)`

Проверка адекватного удаления отображаемого на карте маркера.

№	Исходные данные	Ожидаемый результат
1	marker — существующий и добавленный маркер с координатами latitude = -21.0, longitude = 21.0	Успешное удаление маркера (маркер перестает отображаться на карте)
2	marker = null	Ничего не происходит
3	Удаление маркера, который не размещен на карте	Ничего не происходит
4	Удаление маркера при несуществующей карте	Ничего не происходит

Метод `boolean isAuthorized()`

Проверка статуса авторизации пользователя (авторизован или нет).

№	Исходные данные	Ожидаемый результат
1	Пользователь авторизован (token сохранен и валиден*)	true
2	Пользователь не авторизован (token не сохранен)	false
3	token существует, но не валиден*	false

* — под валидностью token понимается возможность его использования при осуществлении запросов к серверу. Например, если token существует, но устарел, то он считается невалидным.

Метод `boolean isInternetConntectionAvailable()`

Проверка доступности подключения к сети Интернет.

№	Исходные данные	Ожидаемый результат
1	Имеется подключение к сети	true
2	Нет подключения к сети	false

3	Подключение к сети есть (например, беспроводное соединение установлено), но запросы невозможны	false

Метод `void onLocationChanged(Location location)`

Проверка метода, срабатывающего при изменении местоположении пользователя.

№	Исходные данные	Ожидаемый результат
1	Структура location с полями, значения которых не выходят за рамки допустимых значений	this.location == location
2	location = null	Ничего не происходит

III. GetsDbHelper

Метод `void addPoint(Point point)`

Проверка добавления точки в базу данных.

№	Исходные данные	Ожидаемый результат
1	Структура point с заполненными полями, не выходящими за пределы допустимых значений (point.name = “name”, point.latitude = -21.0, point.longitude = 21.0)	Успешно добавленная точка в БД
2, 3, 4, 5, 6, 7	Структура с одним из параметров, выходящим за границы ограничений, latitude = {-200, 200}, longitude = {-200, 200}, raing = {-99.0, 99.0}	Ничего не происходит
8	null в поле name структуры	Добавляется точка с пустым именем.
9	null в качестве значения point	Ничего не происходит
10	Добавление уже существующей точки	Ничего не происходит (точка не добавляется)
11	Большие по длине строки в структуре	Добавление точки в БД

12	Добавление новой точки с существующим UUID	Ничего не происходит (точка не добавляется)
13	Недопустимые символы в строке	Успешное добавление точки, символы экранируются
14	SQL-инъекция в name	Добавление точки с экранированным name
15	Добавление при несуществующей БД	Ничего не происходит

Метод `void addPoints(ArrayList<Point> points)`

Проверка добавления массива точек в базу данных. Метод является надстройкой над методом `addPoint()`.

№	Исходные данные	Ожидаемый результат
1	Массив структур point с заполненными полями, не выходящими за пределы допустимых значений (point.name = “name”, point.latitude = -21.0, point.longitude = 21.0). Поля latitude и longitude каждой точки отличаются на 0.1	Успешно добавленные точки в БД
2, 3, 4, 5, 6, 7	Массив точек с объектом у которого один из параметров выходит за границы ограничений, latitude = {-200, 200}, longitude = {-200, 200}, raing = {-99.0, 99.0}	Успешно добавлены все точки, кроме указанной
8	Массив точек, в котором у каждого объекта один из параметров выходит за границы ограничений.	Ничего не происходит
9	null как параметр метода	Ничего не происходит
10	Массив из нескольких идентичных точек, у которых значения полей не выходят за пределы ограничений	Успешное добавление одной точки
11	Массив из нескольких идентичных точек, данные о которой уже	Ничего не происходит

	существуют в БД	

Метод `ArrayList<Point> getPoints()`

Проверка получения всех точек из базы данных.

№	Исходные данные	Ожидаемый результат
1	Вызов функции для базы данных, в которой есть n объектов (n > 0)	Успешное получение массива с точками из БД (n штук)
2	Вызов функции, когда БД не существует	null
3	Вызов функции, когда в БД нет ни одного объекта	Пустой массив объектов типа Point

Метод `void addCategory(int id, String name, String description, String url)`

Проверка добавления категории в базу данных.

№	Исходные данные	Ожидаемый результат
1	Значения аргументов функции не выходят за ограничения значений. id = 1, name = “Name”, description = “Descr”, url = null	Успешно добавленная категория в БД
2	Параметр id выходит за границы ограничений (id = -90)	Ничего не происходит
3	null в поле name структуры	Добавляется категория с пустым именем
4	Добавление уже существующей категории (id уже существует в БД)	Ничего не происходит (категория не добавляется)
5	Большие по длине строки	Добавление точки в БД
6, 7, 8	SQL-инъекция в name, description и url	Добавление категории с экранированием
9	Добавление при несуществующей БД	Ничего не происходит

Метод `ArrayList<Category> getCategories()`

Проверка получения всех категорий из базы данных.

№	Исходные данные	Ожидаемый результат
1	Вызов функции для базы данных, в которой есть n категорий (n > 0)	Успешное получение массива с точками из БД (n штук)
2	Вызов функции, когда БД не существует	null
3	Вызов функции, когда в БД нет ни одного объекта	Пустой массив объектов типа Category

Метод `String getCategoryName(int categoryId)`

Проверка получения названия категории по её идентификатору.

№	Исходные данные	Ожидаемый результат
1	Вызов функции для базы данных, в которой есть указанный идентификатор	Успешное получение строки с названием категории, соответствующей идентификатору
2	Вызов функции, когда БД не существует	null
3	Вызов функции, когда в БД нет ни одного объекта	null
4	Идентификатор categoryId не существует в БД	null
5	Значение categoryId выходит за ограничения (categoryId = -90)	null

IV. AddNewPointActivity

Метод `void addMarker(LatLng position)`

Проверка получения названия категории по её идентификатору.

№	Исходные данные	Ожидаемый результат

1	Вызов функции с position, указывающим на существующее местоположение ([-90; 90]]	Успешное добавление маркера на карту (маркер отображается на карте)
2	null в качестве параметра функции	Ничего не происходит

Метод `boolean isStringEmpty(String string)`

Проверка строки на пустоту.

№	Исходные данные	Ожидаемый результат
1	Вызов функции с параметром, равным “string”	false
2	Вызов функции с параметром, равным “строка 線ø° å#$ %^&*(”	false
3	Вызов функции с параметром, равным null	false
4	Вызов функции с параметром, равным “ “	false
5	Вызов функции с параметром, равным пустой строке (“”)	true
6	Вызов строки с системными инструкциями для строк: \n, \r, \t	

Оценка покрытия кода тестами

Оценка степени покрытия кода тестами считается по формуле

$$T_{cov} = (LOC_{cov} / LOC_{total}) \times 100\%$$

Где LOC — количество строк кода (cov — строки, покрытые тестами, total — всего строк кода).

Для совокупности рассматриваемых тестов $T_{cov} = (810/1589) \times 100\% \approx 50{,}94\%$

Из общего числа строк кода исключены строки автоматически генерируемых файлов и строки кода, содержащие сценарии тестирования.

Интеграционные тесты

`MapActivity`: метод `void onActivityResult(int requestCode, int resultCode, Intent data)`

№	Исходные данные	Ожидаемый результат
1	Набор параметров, обозначающий переход из AddPointActivity, requestCode = 1, resultCode = RESULT_OK, data = [“latitude”:21.0, “longitude”: -21.0, “name”:”name”]	Данные о точке появились в `MapActivity` и отобразились на карте
2	Неизвестный код запроса	Ничего не происходит
3	Код результата равен RESULT_CANCELED (отмена создания точки)	Ничего не происходит
4	Код результата не равен RESULT_OK и RESULT_CANCELED	Ничего не происходит
5	data не содержит данных	Ничего не происходит
6	data содержит координаты в неправильном формате	Отображение ошибки
7	data не содержит данных о координатах	Ничего не происходит
8	Нет сети	Сохранение точки в локальную базу данных приложения

Стресс-тестирование

`PointsGet`: метод `PointsResponse PointsGet(String token, double latitude, double longitude, int radius)`.

Проверка получения и синтаксического разбора большого по величине файла.

№	Исходные данные	Ожидаемый результат

1	Размер получаемого файла с данными: 5 Мб	Загрузка и синтаксический разбор файла. На выходе: массив PointsReponse.points содержит количество элементов, соответствующее числу элементов, описанному во входящем (загружаемом файле)
2	Размер получаемого файла с данными: 10 Мб	
3	Размер получаемого файла с данными: 25 Мб	
4	Размер получаемого файла с данными: 50 Мб	
5	Размер получаемого файла с данными: 100 Мб	

`GetsDbHelper`: метод `void addPoints(ArrayList<Point> points)`.

Проверка сохранения больших объемов данных в базе данных.

№	Исходные данные	Ожидаемый результат
1	Количество элементов массива points: 500	Сохранение всех объектов в базу данных
2	Количество элементов массива points: 5000	
3	Количество элементов массива points: 10 000	
4	Количество элементов массива points: 500 000	

`MapActivity`: метод `void addMarkerLowLevel(Marker marker)`

Проверка добавления большого числа маркеров на карту.

№	Исходные данные	Ожидаемый результат
1	Число добавляемых точек: 50	Отображение всех маркеров, отсутствие “замораживаний” пользовательского интерфейса при добавлении
2	Число добавляемых точек: 250	
3	Число добавляемых точек: 1000	

4	Число добавляемых точек: 10000	

Аттестационное тестирование

№	Исходные данные	Ожидаемый результат
1	Первый запуск приложения	Открытие окна входа при помощи своей учетной записи
2	Осуществление входа при помощи своей учетной записи	Открытие главного окна приложения
3	Выход из приложения	Закрытие приложения
4	Повторный запуск приложения	Открытие главного окна приложения
5	Открытие главного окна приложения	Отображение карты и местоположения пользователя
6	Загрузка и отображение точек	Точки загружаются и отображаются на карте
7	Открытие окна создания новой точки	Открыто новое окно с картой и отмеченным на ней маркером и предложением ввести название точки и рейтинг
8	Сохранение новой точки	Новая точка создается и успешно отправляется на сервер
9	Сохранение новой точки при отсутствующем соединении с сетью	Точка сохраняется во внутренней базе данных, о чем сообщается пользователю

Примеры реализации тестов

Описанные ниже тесты реализованы при помощи библиотеки JUnit с использованием Android SDK (поддержка платформозависимых средств).

Более подробно тестирование для Android описано в TestingFundamentals на портале разработчиков Android Developers.

```
// Пример тестирования метода addPoint()

// Тест 1 (заполненная структура не содержит полей, значения которых
выходят за рамки ограничений)
public void testDb1_1() {
    Point point = new Point();
    point.name = "name";
    point.latitude = -21.0;
    point.longitude = 21.0;
    point.rating = 4.5f;

    GetsDbHelper dbHelper = new GetsDbHelper(getContext(),
     DatabaseType.USER_GENERATED);
    dbHelper.clearDatabase();  // Очистка базы данных
    dbHelper.addPoint(point);  // Добавление точки

    ArrayList<Point> points = dbHelper.getPoints();
    Point pointFromDb = dbHelper.getPoints().get(0);

    // В БД должна быть одна точка
    assertEquals(points.size(), 1);

    // Данные о точке должны совпадать с добавленными
    assertEquals(pointFromDb.name, point.name);
    assertEquals(pointFromDb.latitude, point.latitude);
    assertEquals(pointFromDb.longitude, point.longitude);
    assertEquals(pointFromDb.rating, point.rating);
}



// Тест 8. null в качестве значения поля name
public void testDb1_8() {
    Point point = new Point();
    point.name = null;
    point.latitude = -21.0;
    point.longitude = 21.0;
    point.rating = 4.5f;

    GetsDbHelper dbHelper = new GetsDbHelper(getContext(),
     DatabaseType.USER_GENERATED);
    dbHelper.clearDatabase();  // Очищаем БД
    dbHelper.addPoint(point);  // Добавляем точку
```

```
    ArrayList<Point> points = dbHelper.getPoints();

    /* БД не должна содержать объектов, т.к. структура заполнена
     * некорректно
     */
    assertEquals(points.size(), 0);
}


// Тест 9. null в качестве аргумента функции
public void testDb1_9() {
    GetsDbHelper dbHelper = new GetsDbHelper(getContext(),
      DatabaseType.USER_GENERATED);
    dbHelper.clearDatabase();

    // Добавляем "точку"
    dbHelper.addPoint(null);

    ArrayList<Point> points = dbHelper.getPoints();

    /* В БД не должно быть объектов
     *
     * При тестировании было обнаружено, что метод addPoint(null)
     * аварийно завершает работу с NullPointerException
     */
    assertEquals(points.size(), 0);
}
```

Результаты

Блочное тестирование

№	Модуль	Всего тестов	Ошибки
1	API Interactions	34	2
2	MapActivity	30	12
3	GetsDbHelper	46	21
4	AddNewPointActivity	8	1
5	Интеграционное	8	0

6	Стресс-тестирование*	13	10
7	Аттестационное	9	0

* — Стресс-тестирование

`PointsGet`: метод `PointsGet(String token, double latitude, double longitude, int radius)`.

Данный метод в состоянии загружать большие объемы данных, но, поскольку в нем не только загружаются данные, но и осуществляется синтаксический разбор, операции требуют много памяти и могут завершаться ошибкой (тесты 3, 4, 5) на устройствах с небольшим количеством оперативной памяти.

`GetsDbHelper`: метод `void addPoints(ArrayList<Point> points)`.

База данных успешно справляется с сохранением больших объемов данных (до 10 000 элементов за раз), но в контексте пользовательского интерфейса большие объемы данных необходимо передавать и сохранять в отдельном потоке для избежания “замораживания” приложения и, как следствие, его аварийного завершения.

`MapActivity`: метод `void addMarkerLowLevel(Marker marker)`

“Замораживания” интерфейса пользователя начинают проявляться на втором тесте (добавление 250-ти точек). В данном случае также применимы рекомендации о многопоточности.

Обнаруженные ошибки

Функция	№ теста	Ошибочный результат
Модуль `MapActivity`		
`addMarkerLowLevel()`	5	`Вызывается необрабатываемое исключение NullPointerException`

addMarker()	2-8	Вызывается внутреннее исключение MapBox SDK
addMarker()	10	Вызывается необрабатываемое исключение NullPointerException
addMarker()	13	Вызывается необрабатываемое исключение NullPointerException
deleteMarker()	4	Вызывается необрабатываемое исключение NullPointerException
isAuthorized()	3	true (невалидный token считается валидным)
isInternetConntection Available()	3	true (подключение к точке доступа без возможности совершения запросов ошибочно считается доступом к сети)
Модуль GetsDbHelper		
addPoint()	2-7	Все точки с неправильными параметрами добавляются в БД
addPoint()	9	Вызывается необрабатываемое исключение NullPointerException
addPoint()	10	Несколько одинаковых точек добавляются в БД
addPoint()	12	Несколько одинаковых точек добавляются в БД
addPoints()	2-7	Все точки с неправильными параметрами добавляются в БД
addPoints()	8	Вызывается необрабатываемое исключение NullPointerException
addPoints()	10	Добавляются все точки
getPoints()	1	Всегда возвращается null (*критическая ошибка*)
getPoints()	3	null вместо пустого массива
addCategory()	2	Данные добавляются в БД
getCategories()	3	null вместо пустого массива
Модуль AddNewPointActivity		

`isStringEmpty()`	3	true (null считается пустой строкой)
Стресс-тестирование		
`PointsGet()`	3-5	Ошибка разбора JSON-файла
`addPoints()`	3-4	“Замораживание”
`addMarkerLowLeve()`	2-4	“Замораживание”

Дата проведения тестирования и обнаружения ошибок: 17.12.2014.

Обнаруженные ошибки можно разделить на три класса:

1. Отсутствие проверки равенства переменной null
2. Отсутствие проверки выхода значения переменной за интервал допустимых значений
3. Другое