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 »

Прототип мобильного приложения для социальной сети facebook.com

Выполнил:
Студент 6 курса группы 22 608
Н.А Романюк

_____
подпись

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

_____
подпись

Петрозаводск
2015

Оглавление

1. Описание приложения

Объектом тестирования является прототип мобильного приложения для социальной сети facebook.com, обмен данными с социальной сетью осуществляется через BaaS – parse.com.

facebook.com — это социальная сеть для быстрой и удобной коммуникации между людьми по всему миру. Возможность обмениваться сообщениями и делиться фотографиями, следить за новостями Ваших друзей и заводить новые знакомства, смотреть видеозаписи и слушать музыку, вступать в сообщества и играть в игры.

Backendless – платформа бэкенд как сервис (Backend as a Service), которая предоставляет готовую облачную серверную инфраструктуру для всех типов приложений. API платформы доступны через нативные СДК для следующих клиентских окружений: JavaScript, Android, iOS, Windows Phone, Flex/AIR. Все АПИ также доступны через REST.

1.1. Основной функционал прототипа

- Авторизация пользователя в facebook.com
- Возможность просмотра : facebook.com/me/feed
- Возможность частичного просмотра : facebook.com/me
- Возможность добавления : комментариев, «лайков» к записям на странице facebook.com/me/feed

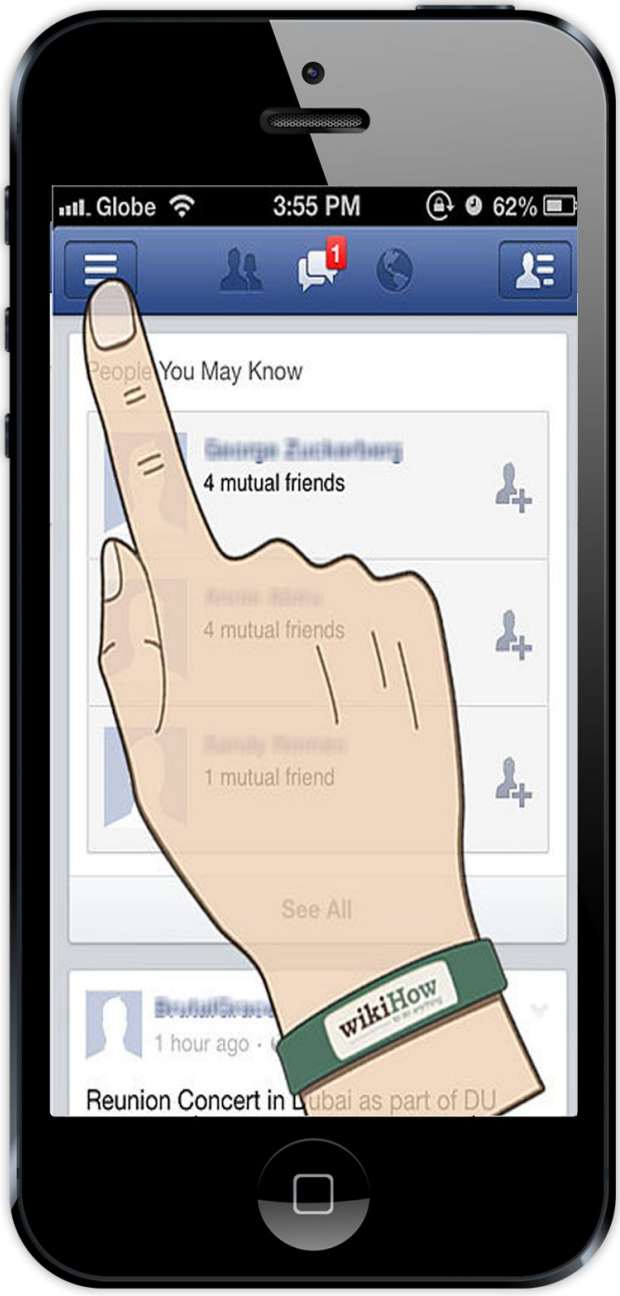


Рис 1. Прототип пользовательского интерфейса

2. Объект и стратегия тестирования

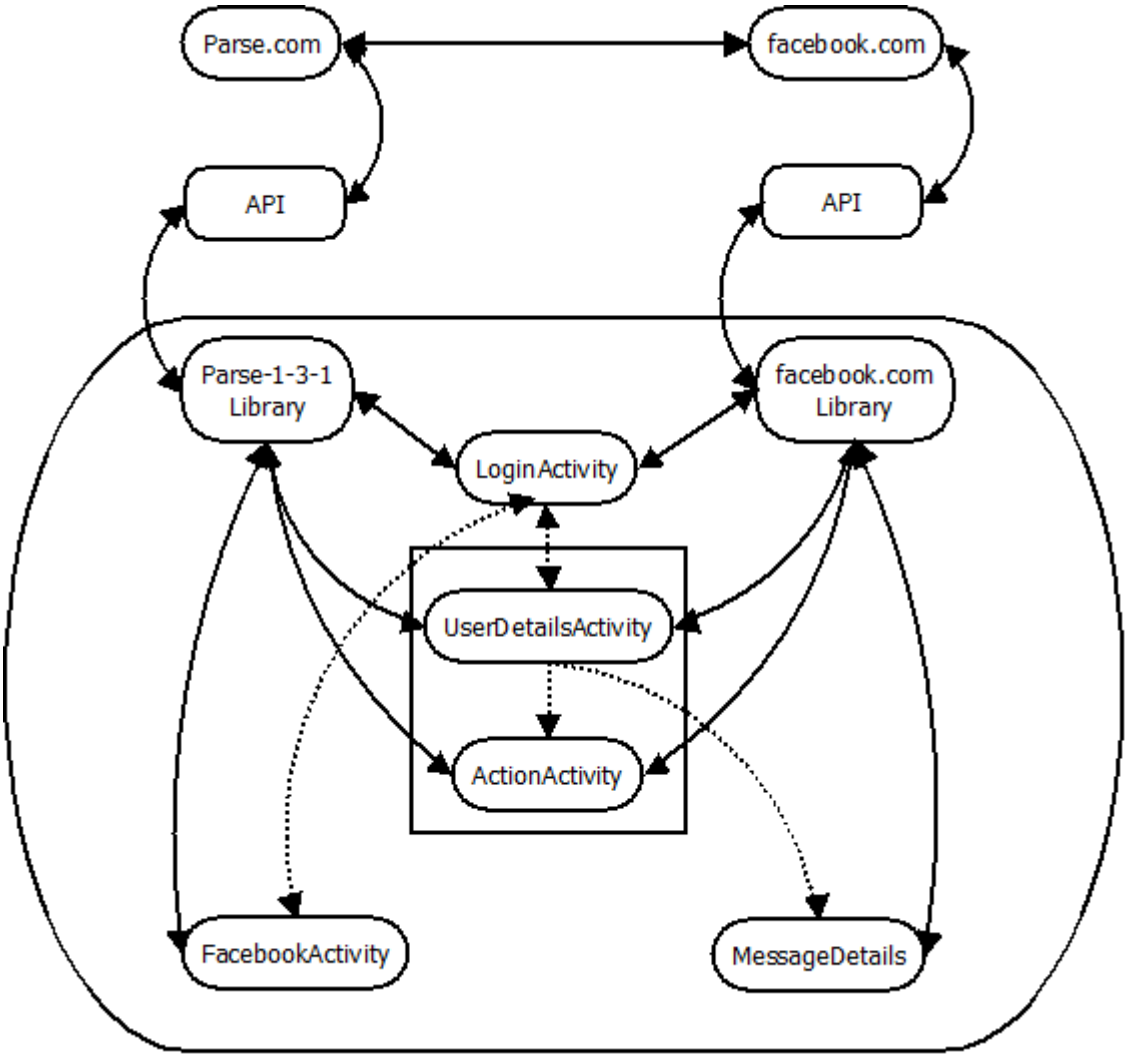


Рис.2 Архитектура прототипа приложения

2.1. Описание архитектуры

Библиотеки:

parse-1.3.1 – Официальная библиотека облачного сервиса parse.com, по средствам которой приложение обменивается информацией с облачным сервисом.

facebook.com - Официальная библиотека социальной сети facebook.com, содержащая специальные классы для обмена данными с сервисами facebook.com, а так же включающая в себя множества специальных классов адаптеров, форм авторизации итд.

Классы:

FaceebookActivity – класс содержащий один метод для инициализации необходимых параметров для работы с облачным сервисом parse.com

LoginActivity – класс содержащий методы служащие для авторизации пользователя

UserDetailsActivity – класс включающий в себя методы для деавторизация пользователя, частичного получения информации о пользователе (facebook.com/me), просмотр ленты новостей пользователя (facebook.com/me/feed/).

ActionActivity – класс содержит метод добавления комментариев к записи, добавление «лайка» к определенной записи, получение подробной информации о записи.

MessageDetails – класс-адаптер для хранения информации о записи.

2.2 Стратегия тестирования

Тестированию подлежат методы следующих классов: ActionActivity, UserDetailsActivity, LoginActivity.Тестирование не будет проводиться для системных методов ActionActivity, UserDetailsActivity, LoginActivity (методы onResume(), onStart(), onPause()). Тестирование не будет проводиться для класса MessageDetails так как представляет собой простой класс адаптер, FaceebookActivity не подлежит тестированию так как в нем содержится вызов одного метода из сторонней библиотеки, а так же тестированию не подлежат продукты сторонних разработчиков, библиотеки parse-1.3.1 и facebook.com.

Для проведения интеграционного тестирования применяется стратегия восходящего тестирования.

1. ActionActivity

- postRequest() – метод получения информации о записи
- likeRequest() – метод добавления/удаления «лайка» у записи
- commentRequest() – метод добавления комментария к записи

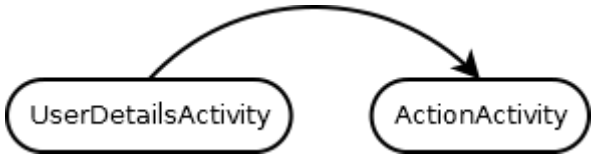
2. LoginActivity

- onLoginButtonClicked() – метод вызова механизмов авторизации пользователя

3. UserDetailsActivity

- makeMeRequest() - метод получения информации о пользователе
- feedRequest() - метод получения новостной ленты пользователя
- updateViewsWithProfileInfo() – метод отображения информации о пользователе в пользовательском интерфейсе

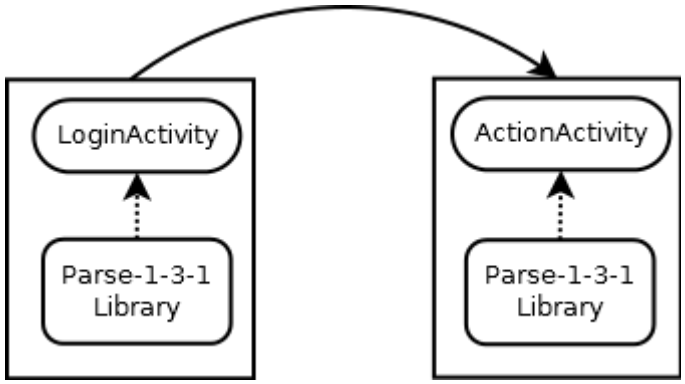
Для проведения интеграционного тестирования применяется стратегия восходящего тестирования.



1.UserDetailsActivity –> ActionActivity

1) UserDetailsActivity -> вызываем метод showActionActivity(String s), который в свою очередь инициализирует стандартный класс Intent.

2) ActionActivity -> инициализируем стандартный класс Intent, получаем переданные данные из другой формы.



2.LoginActivity + библиотека parse.com -> ActionActivity+ библиотека parse.com

1) LoginActivity + библиотека parse.com -> вызываем метод onLoginButtonClicked().

2) LoginActivity.onLoginButtonClicked() + библиотека parse.com -> вызывается метод logIn(Activity activity, LogInCallback callback) из библиотеки parse.com.

3) ActionActivity+ библиотека parse.com -> вызываем метод commentRequest (String Sessionkey,String fId, String comment), в качестве параметра Sessionkey, вызывается метод getSession() из библиотеки библиотеки parse.com.

4) ActionActivity+ библиотека parse.com -> вызываем метод likeRequest (String Sessionkey,Sting fId), в качестве параметра Sessionkey, вызывается метод getSession() из библиотеки библиотеки parse.com.

5) ActionActivity+ библиотека parse.com -> вызываем метод postRequest (String Sessionkey)в качестве параметра Sessionkey, вызывается метод getSession() из библиотеки библиотеки parse.com.

Стресс-тесты проверяют то, как поведет себя приложения в случае обработки большого объема данных. Стресс-тесты проводятся для модуля получения данных о новостной ленте пользователя (facebook.com/me/feed/), для проверки отсутствия “замораживаний” интерфейса пользователя. Эффект “замораживания” не более 2х секунд будем считать приемлемым.

Аттестационное тестирование, проверяет наличие основного заявленного функционала прототипа, а так же проверят работоспособность приложения в целом.

Для тестирования используется следующее окружение: Eclipse Juno, Android SDK (API v21), Java SE 1.7., Windows 7, Смартфон с ОС Android 4.0 и выше(с подключенным безлимитный 3G интернетом, доступом к facebook.com и parse.com), USB-кабель.

Язык программирования — Java (Android SDK).

Накладываемые ограничения

● “длинная строка” — строка, содержащая более 10 тысяч символов
● “пустая строка” — строка, не содержащая символов, “”
● fId – строка содержащая латинские символы и числа, длинной не более 100 символов.
●Sessionkey- строка содержащая латинские символы и числа, длинной не более 100 символов.

●currentUser - строка содержащая латинские символы и числа, длинной не более 100 символов.
●comment - строка содержащая символы и числа, длинной не более 1000 символов.

Задачи тестирования:
● Выявление ошибок для их дальнейшего исправления
● Проверка работы приложения в непредвиденных ситуациях

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

Результаты тестирования должны быть представлены в январе 2015 года.

3. Подробный план тестирования

3.1. Модульное тестирование

Класс ActionActivity

Метод commentRequest(String Sessionkey,String fId, String comment)

№	Исходные данные	Ожидаемый результат	Тип теста
1.1	Sessionkey=null	Обработка ошибки	Негативный
1.2	Нет сети	Обработка ошибки	Негативный
1.3	fId = null	Обработка ошибки	Негативный
1.4	fId- в пределах допустимых значений (существующая запись), comment - в пределах допустимых значений	Добавление комментария к записи	Общий

1.5	fId- в пределах допустимых значений (существующая запись, comment=null	Обработка ошибки	Негативный
1.6	fId- в пределах допустимых значений (существующая запись), comment = «длинной строке»	Обработка ошибки	Негативный
1.7	fId- в пределах допустимых значений (существующая запись), comment = “ ”	Добавление комментария к записи	Негативный

Метод likeRequest (String Sessionkey,Sting fId)

№	Исходные данные	Ожидаемый результат	Тип теста
2.1	Sessionkey=null	Обработка ошибки	Негативный
2.2	Нет сети	Обработка ошибки	Негативный
2.3	sId - в пределах допустимых значений (существующая запись)	Добавление “Лайка” к записи	Общий
2.4	sId - в пределах допустимых значений (существующая запись). Вторичная попытка добавления “Лайка”.	Удаление “Лайка” у записи	Общий

Метод postRequest (String Sessionkey)

№	Исходные данные	Ожидаемый результат	Тип теста
3.1	Sessionkey=null	Обработка ошибки	Негативный
3.2	Нет сети	Обработка ошибки	Негативный
3.3	sId - в пределах допустимых значений (существующая запись)	Успешно получен массив с записями	Общий

3.4	Массив записей с синтаксической ошибкой	Обработка ошибки	Негативный
3.5	Один или несколько значение массива = null	null заменяется на « »	Общий

Класс LoginActivity

Метод onLoginButtonClicked()

№	Исходные данные	Ожидаемый результат	Тип теста
4.1	Попытка авторизации несуществующего пользователя	Обработка события, регистрация нового пользователя	Общий
4.2	Нет сети	Обработка ошибки	Негативный
4.3	Авторизация существующего пользователя	Получение индентификатора пользователя текущей сессии.	Общий

Класс UserDetailsActivity

Метод makeMeRequest()

№	Исходные данные	Ожидаемый результат	Тип теста
5.1	Синтаксически правильный массив	Присвоение переменным интерфейса полученных значений	Общий
5.2	Нет сети	Обработка ошибки	Негативный
5.3	Пустой массив данных	Обработка ошибки	Негативный
5.4	Массив данных с синтаксической ошибкой	Обработка ошибки	Негативный

Метод feedRequest()

№	Исходные данные	Ожидаемый результат	Тип теста
6.1	Синтаксически правильный	Присвоение	Общий

	массив	переменным интерфейса полученных значений	
6.2	Нет сети	Обработка ошибки	Негативный
6.3	Массив содержащий “длинные строки”	Синтаксический разбор массива	Общий
6.4	Массив записей с синтаксической ошибкой	Обработка ошибки	Негативный
6.5	Один или несколько значение массива = null	null заменяется на « »	Общий

Метод updateViewsWithProfileInfo(ParseUser currentUser)

№	Условия	Ожидаемый результат	Тип теста
7.1	currentUser == null	Обработка ошибки	Негативный
7.2	Один или несколько значение массива = null	null заменяется на « »	Общий
7.2	Синтаксически правильный массив	Отображение информации о пользователе	Общий

Оценка покрытия кода тестами
Оценка степени покрытия кода тестами считается по формуле
Tcov = (LOCcov / LOCtotal) × 100%
Где LOC — количество строк кода (cov — строки, покрытые тестами, total — всего строк кода).
Для совокупности рассматриваемых тестов Tcov = (475/932) × 100% ≈ 50,96%
Из общего числа строк кода исключены строки автоматически генерируемых файлов и строки кода, содержащие сценарии тестирования.

3.2. Интеграционное тестирование

1.UserDetailsActivity –> ActionActivity

1) UserDetailsActivity -> вызываем метод showActionActivity(String s), который в свою очередь инициализирует стандартный класс Intent.

2) ActionActivity -> инициализируем стандартный класс Intent, получаем переданные данные из другой формы.

№	Исходные данные	Ожидаемый результат	Тип теста
1.1	data не содержит данных	Обработка ошибки	Негативный
1.2	data содержит длинную строку	Обработка ошибки	Негативный
1.3	Нет сети	Обработка ошибки	Негативный
1.4	data – содержит все необходимые данные	Данный переданы в форму ActionActivity успешно	Общий

2.LoginActivity + библиотека parse.com -> ActionActivity+ библиотека parse.com

1) LoginActivity + библиотека parse.com -> вызываем метод onLoginButtonClicked().

2) LoginActivity.onLoginButtonClicked() + библиотека parse.com -> вызывается метод logIn(Activity activity, LogInCallback callback) из библиотеки parse.com.

3) ActionActivity+ библиотека parse.com -> вызываем метод commentRequest (String Sessionkey,String fId, String comment), в качестве параметра Sessionkey, вызывается метод getSession() из библиотеки библиотеки parse.com.

№	Исходные данные	Ожидаемый результат	Тип теста
2.1	Sessionkey =null	Обработка ошибки	Негативный
2.2	Sessionkey - в пределах допустимых значений (существующая запись)	Получен индентифкатор сессии	Общий

4) ActionActivity+ библиотека parse.com -> вызываем метод likeRequest (String Sessionkey,Sting fId), в качестве параметра Sessionkey, вызывается метод getSession() из библиотеки библиотеки parse.com.

№	Исходные данные	Ожидаемый результат	Тип теста

3.1	Sessionkey =null	Обработка ошибки	Негативный
3.2	Sessionkey - в пределах допустимых значений (существующая запись)	Получен индентифкатор сессии	Общий

5) ActionActivity+ библиотека parse.com -> вызываем метод postRequest (String Sessionkey)в качестве параметра Sessionkey, вызывается метод getSession() из библиотеки библиотеки parse.com.

№	Исходные данные	Ожидаемый результат	Тип теста
4.1	Sessionkey =null	Обработка ошибки	Негативный
4.2	Sessionkey - в пределах допустимых значений (существующая запись)	Получен индентифкатор сессии	Общий

3.3. Аттестационное тестирование

№	Исходные данные	Ожидаемый результат	Тип теста	Тестируемый функционал
1	Первый запуск приложения	Открытие окна с формой для авторизации	Общий	-
2	Осуществление входа при помощи своей учетной записи	Открытие окна с формой с информацией о пользователе	Общий	Авторизация пользователя в facebook.com
3	Выход из приложения	Закрытие приложения	Общий	-
4	Повторный запуск приложения	Открытие окна с формой с информацией о пользователе	Общий	-
5	Деавторизация пользователя	Открытие окна с формой для авторизации	Общий	-

6	Открытие окна с формой с информацией о пользователе	Отображение информации о пользователе, отображение новостной ленты пользователя	Общий	1)Возможность просмотра : facebook.com/me/feed 2)Возможность частичного просмотра : facebook.com/me
7	Открытие окна записи из индивидуальной ленты пользователя	Открыто окно с информацией о записи, а также с формой для отправки комментария к записи	Общий	-
8	Отправка комментария к записи	Комментарий к записи отправлен	Общий	Возможность добавления : комментариев, «лайков» к записям на странице facebook.com/me/feed
9	Отправка «лайка» к записи	«Лайк» к записи отправлен	Общий	Возможность добавления : комментариев, «лайков» к записям на странице facebook.com/me/feed
10	Запуск приложения с выключенным интернетом	Отображение сообщения пользователя о отсутствие соединения с интернетом	Негативный	-
11	Открытие окна с формой информацией о	Отображение информации о пользователе, ни	Негативный	-

	пользователе с отсутствующими записями новостной в ленте	одной записи не отображено		

3.4 Стресс-тестирование

Среднее размер получаемой записи 300 байт.

Средняя скорость высокоскоростного мобильного доступа к сети Интернет 256 Кбайт/с.

Ожидаемое количество записей у пользователя 250-400, 400 и более записей будем считать большим объемом данных.

№	Число получаемых записей	Ожидаемое время загрузки записи
1	10	< 0.01 сек
2	100	< 0.1 сек
3	250	< 0.3 сек
4	500	< 0.06 сек
5	10000	< 12 сек

Класс UserDetailsActivity feedRequest()

№	Исходные данные	Ожидаемый результат
1	Число получаемых записей: 10	<1 сек
2	Число получаемых записей: 100	<1 сек
3	Число получаемых записей: 250	<2 сек
4	Число получаемых записей: 500	<4 сек

5	Число получаемых записей: 10000	<20 сек

3.5. Пример реализации тестов:

Пример тестирования метода

Тест 1.4

```
…
commentRequest(null,1321333212, "test"));
…

        private void commentRequest(String Sessionkey,String fId, String comment) {

                if (Sessionkey!=null && fId !=null && comment !=null){
                Bundle params = new Bundle();
                params.putString("message", editComment.getText().toString());
                /* make the API call */
                new Request(
                                ParseFacebookUtils.getSession(),
                                "/"+sId+"/comments",
                   params,
                   HttpMethod.POST,
                   new Request.Callback() {
                      public void onCompleted(Response response) {

                       }
                   }
                ).executeAsync();
        }}else{printerr(10);}

        }
```

4. Результаты

4.1. Блочное тестирование

№	Класс	Всего тестов	№ Ошибки
1	ActionActivity	16	1.1,1.2,2.1,2.2,3.1,3.2
2	LoginActivity	3	4.2
3	UserDetailsActivity	12	5.2, 6.2

Тест №	**Метод**	**Описание ошибки**	**Методы устранения ошибки**
1.1	CommentRequest	Ошибка не обрабатывается	Добавить проверку наличия активной сессии
1.2	commentRequest	Вызывается необрабатываемое исключение NullPointerException	Добавить проверку на наличие подключению к интернету
2.1	LikeRequest	Ошибка не обрабатывается	Добавить проверку наличия активной сессии
2.2	LikeRequest	Вызывается необрабатываемое исключение NullPointerException	Добавить проверку на наличие подключению к интернету
3.1	postRequest	Ошибка не обрабатывается	Добавить проверку наличия активной сессии
3.2	postRequest	Вызывается необрабатываемое исключение NullPointerException	Добавить проверку на наличие подключению к интернету
4.2	onLoginButtonClicked	Вызывается необрабатываемое исключение NullPointerException	Добавить проверку на наличие подключению к интернету
5.2	makeMeRequest	Вызывается необрабатываемое исключение NullPointerException	Добавить проверку на наличие подключению к интернету
6.2	feedRequest	Вызывается необрабатываемое исключение NullPointerException	Добавить проверку на наличие подключению к интернету

.

4.2 Интеграционное тестирование:

№	Тестирование	Всего тестов	№ Ошибки
1	Интеграционное	9	1.3
2	Стресс-тестирование*	5	5
3	Аттестационное	11	0

Тест №	**Описание ошибки**	**Методы устранения ошибки**
Интеграционное		
1.3	Ничего не происходит	Добавить проверку на наличие подключению к интернету
Стресс-тестирование*		
5	Подвисание смартфона	Использовать многопоточные методы обработки данных

* — Стресс-тестирование

UserDetailsActivity feedRequest()

Данный метод в состоянии загружать большие объемы данных, но в связи с осуществлением синтаксический разбор, требует много памяти и может завершаться ошибкой (тест № 5) на устройствах с небольшим количеством оперативной памяти.

Дата проведения тестирования и обнаружения ошибок: 30.01.2015.