Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по дисциплине «Верификация ПО»

Мобильное приложение программного комплекса оптимизации состояния склада лесопильного и/или деревообрабатывающего предприятия

Выполнил:
студент 22608 Корнышева М. А.

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

Петрозаводск
2018

1. Описание объекта тестирования

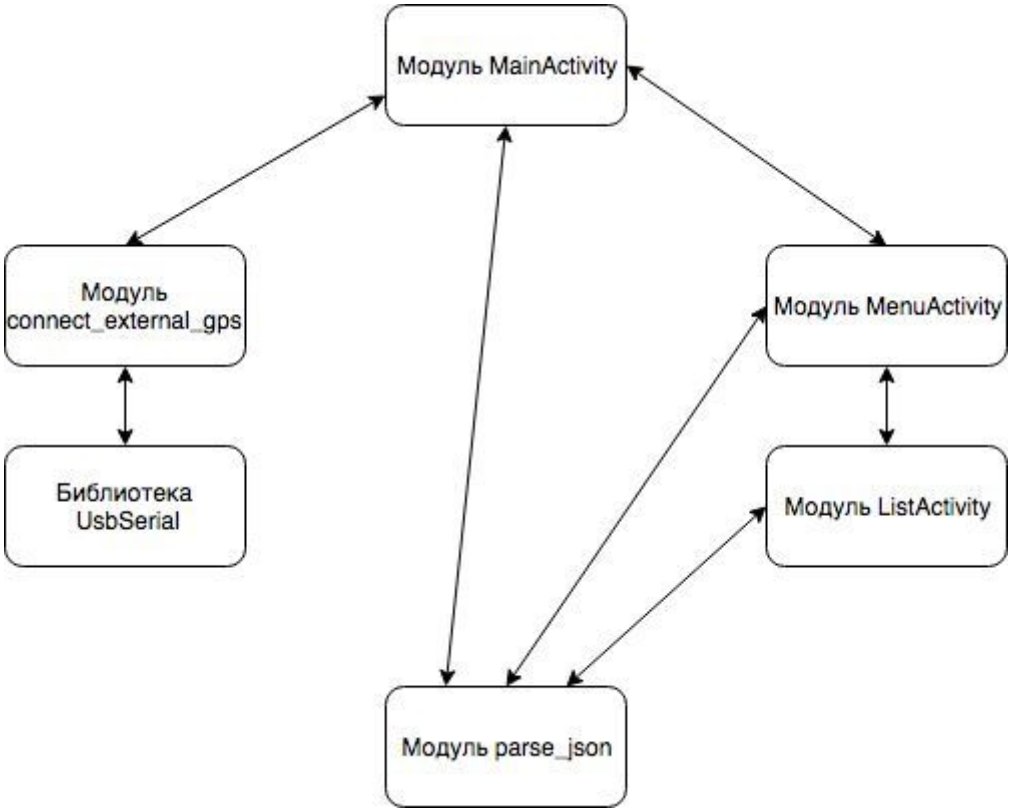
Объектом тестирования является мобильное приложение для ОС Андроид, являющееся частью программного комплекса управления складом лесопильного и/или деревообрабатывающего предприятия. Приложение реализовано на языке Java с использованием сторонней библиотеки UsbSerial для работы с внешним gps-передатчиком, подключаемым по usb. Приложение спроектировано и реализовано для функционирования с Android Api 22 и выше.

Основные функциональности объекта тестирования, предназначенные для использования пользователем:

1. установление соединения с gps-антенной при ее подключении в usb-разъем, периодический опрос gps-антенны для получения данных с текущими координатами и парсинг данных от gps-антенны для получения текущих координат
2. инициированный пользователем вход в приложение, выбор объектов работы на предприятии, получение заданий на перевозку пакетов пиломатериалов на выбранных объектах работы
3. периодический опрос сервера на предмет доступности сервера, мониторинг состояния интернет-соединения
4. формирование журнала операций, не отправленных на сервер, с последующим обновлением(очищением) журнала после успешной отправки его на сервер для обработки (т.н. аварийный режим)

2. Стратегия тестирования

2.1. Структура приложения



Модуль MainActivity является точкой входа в приложение и служит для авторизации пользователя в приложении, а также для информирования пользователя о состоянии внешней gps-антенны. Авторизация происходит путем взаимодействия пользователя с графическим интерфейсом приложения и введением некоторых значений в поля ввода. Графический интерфейс содержит следующие элементы:

- выпадающий список с логинами
- поле ввода пароля
- кнопка “Вход”
- индикатор состояния gps-антенны

- нередактируемое поле ввода даты
- нередактируемое поле ввода времени.

Чтобы продолжить работу в приложении, пользователь должен ввести свои логин и пароль в соответствующие поля (поле ввода логина и поле ввода пароля) и нажать кнопку “Вход”.

Индикатор состояния gps-антенны имеет два состояния: антенна работает и антенна не работает. О состоянии “работает” сигнализирует зеленый цвет индикатора, о состоянии “не работает” - красный. Индикатор меняет цвет с красного на зеленый при подключении работоспособной антенны и с зеленого на красный при отключении антенны или при ее неработоспособности.

Модуль содержит следующие функции:

- boolean login(String login, String pwd)
- void changeStateGps(boolean state)
- void showDate()
- void checkGps()
- void showUsers(String s)

Функция login осуществляет авторизацию пользователя и вызывается при нажатии на кнопку “Вход”. В случае успешной авторизации происходит переход на другой экран приложения. В случае неуспешной перехода не происходит, показывается уведомление “Неверный логин или пароль”. На время тестирования авторизация в приложении возможна под двумя учетными записями:

- пользователь Alex с логином “alex” и паролем “123”
- пользователь Testuser с логином “testuser” и паролем “12345”.

Функция changeStateGps осуществляет смену цвета индикатора состояния gps-антенны. При получении в качестве параметра значения true меняется на зеленый, в противном случае на красный.

Функция showDate производит отображение текущей даты в элемент поле ввода даты. Внутри функции используется функция parse_date из модуля parse_json.
Функция checkGps вызывается раз в N секунд, где N - число, задаваемое в настройках приложения. Задача функции - инициировать считывание нового потока данных с gps-антенны и в завимости от результатов инициировать изменение цвета индикатора состояния gps-антенны.
Функция showUsers обновляет список логинов в выпадающем списке. Список пользователей передается в функцию в качестве параметра в виде json’а вида:
“[{name:value,login:value,id:value}
{name:value,login:value,id:value}
{name:value,login:value,id:value}
{}
...]”,
преобразованного в строку. Данная строка передается в модуль parse_json функцию parse_user. В результате json преобразуется в список объектов класса User и передается обратно в showUsers, где полученный список показывается в элементе выпадающий список.

Модуль connect_external_gps предназначен для обработки событий подключения/отключения внешних usb-устройств. Обработка происходит при подключении всех usb-устройств, в процессе обработки “отсеиваются” устройства, не являющиеся gps-антеннами. Далее с подключенной gps-антенной инициализируется соединение и считывается поток данных nmea формата до того момента, пока не произойдет закрытие соединения антенной или отсоединение антенны.
Модуль содержит следующие локальные переменные:

- boolean isGpsAvailable
- String lat
- String long
- UsbDevice device
- UsbConnection connection

Модуль содержит следующие функции:

- UsbDevice openDevice(Intent i)
- UsbConnection openConnection()
- String readData()
- void parseData(String data)
- void closeConnection()

Функция **openDevice** при подключенной gps-антенне производит создание экземпляра класса UsbDevice, который в дальнейшем используется для открытия соединения и считывания потока данных.

Функция **openConnection** создает и открывает подключение (экземпляр класса UsbConnection) для ранее созданного экземпляра класса UsbDevice, которое в дальнейшем используется для считывания потока данных с gps-антенны

Функция **readData** при открытом соединении с gps-антенной считывает поток данных определенного формата:

$GPRMC,183729,A,3907.356,N,12102.482,W,000.0,360.0,08301,015.5,E*6F

$GPRMB,A,,,,,,,,,,,V*71

$GPGGA,183730,3907.356,N,12102.482,W,1,05,1.6,646.4,M,-24.1,M,,*75

$GPGSA,A,3,02,,,07,,09,24,26,,,,,1.6,1.6,1.0*3D

$GPGSV,2,1,08,02,43,088,38,04,42,145,00,05,11,291,00,07,60,043,35*71

$GPGSV,2,2,08,08,02,145,00,09,46,303,47,24,16,178,32,26,18,231,43*77

$PGRME,22.0,M,52.9,M,51.0,M*14

$GPGLL,3907.360,N,12102.481,W,183730,A*33

$PGRMZ,2062,f,3*2D

$PGRMM,WGS 84*06

$GPBOD,,T,,M,,*47

$GPRTE,1,1,c,0*07

$GPRMC,183731,A,397.482,N,12102.436,W,000.0,360.0,080301,015.5,E*67

$GPRMB,A,,,,,,,,,,,V*71

Т.е. поток данных состоих из набора строк определенного формата. Каждая строка начинается с символа “$”, за которым следует строковый признак того, какие данные находятся в строке. После признака следуют непосредственно сами данные, в примере выше приведены произвольные строковые/символьные данные для наглядности с сохранением формата. Задача функции извлечь из потока строку вида “GPGLL,3907.360,N,12102.481,W,183730,A*33”, которая в дальнейшем используется для извлечения координат.

Функция **parseData** в качестве параметра получает строку вида “GPGLL,1111.11,a,yyyyy.yy,a,hhmmss.ss,A*hh” и извлекает из нее gps-координаты широту и долготу.

Т.е. условно входная строка содержит 7 подстрок, разделенных запятыми. Рассмотрим смысл 2-7 построк:

2. Географическая широта местоположения.

3. Север/Юг (N/S).

4. Географическая долгота местоположения.

5. Запад/Восток (E/W).

6. Гринвичское время на момент определения местоположения.

7. Статус A = данные верны, V = данные не верны

8. Контрольная сумма строки.

Функция **closeConnection** закрывает открытое соединение с usb-устройством, если таковое имеется.

Модуль parse_json предназначен для обработки данных, приходящих с сервера приложения в виде Json-объектов и Json-массивов. При получении Json-объектов/массивов сервера они преобразуются в строковые данные, и задача методов модуля преобразовать полученные строки в Json-объекты/массивы.

Модуль содержит следующие функции:

- ArrayList<User> parse_user(String s)
- String parse_date(String s)

Функция **parse_date** производит перевод и отображение текущей даты из формата "dd MM yyyy EEE" в формат "день, dd месяц yyyy", где входная строка содержит 4 подстроки, разделенных пробелами. Рассмотрим смысл построк:

dd - текущее число, в двузначном формате. числа 1...9 обозначаются как 0X

MM - текущий месяц в виде числа, числа 1...9 обозначаются как 0X, в диапазоне [1, 12]

yyyy - текущий год, четырехзначное число

EEE - текущий день недели в локали US, первые 3 буквы названия

день - текущий день недели в локали RU, полностью

dd - текущее число, в двузначном формате. числа 1...9 обозначаются как 0X

месяц - текущий месяц в родительном падеже, полностью

yyyy - текущий год, четырехзначное число.

Функция **parse_user** преобразует список пользователей из экземпляра класса JsonArray в список экземпляров класса User. Список пользователей передается в функцию в качестве параметра в виде json’а вида:

“[{name:value,login:value,id:value}

{name:value,login:value,id:value}

{name:value,login:value,id:value}

{}

{}

...]”,

преобразованного в строку.

2.2. Стратегия блочного тестирования

Для проведения блочного тестирования необходимо определить все возможные входные данные и соответствующие им ожидаемые результаты. При этом будут тестироваться следующие функции:

- login
- changeStateGps
- showDate
- checkGps
- showUsers
- openDevice
- openConnection
- readData
- parseData
- closeConnection
- parse_user
- parse_date

Тесты будут проводиться с помощью строенного функционала среды AndroidStudio, а также тестировщиком вручную, когда автоматизировать тестирование невозможно или нецелесообразно.

Автоматические тесты включают в себя определение действий, производимых до теста, определение входных данных, тестирование функций с помощью утверждений (проверка выходных значений или выброс исключений).
Тесты, выполняемые вручную, включают действия или проверки, которые невозможно выполнить автоматически. При этом, каждый для каждого такого теста определяется алгоритм действий и набор проверок.

2.3. Стратегия интеграционного тестирования

Для проведения интеграционного тестирования необходимо определить все возможные входные данные, соответствующие им ожидаемые результаты.

Интеграция со сторонней библиотекой UsbSerial отдельно тестироваться не будет, т. к. покрывается блочными тестами.

Интеграционные тесты разделены на группы, каждая из которых тестирует свой порядок интеграции модулей системы:

1. группа тестов взаимодействия модулей MainActivity и connect_external_gps, а именно функции checkGps и функции parseData. При данном взаимодействии функция connect_external_gps.parseData вызывается в процессе работы MainActivity.checkGps.
2. группа тестов взаимодействия модулей MainActivity и parse_json, а именно функции MainActivity.showDate и parse_json.parse_date и MainActivity.showUsers и parse_json.parse_user.При данном взаимодействии parse_json.parse_date вызывается в процессе работы

MainActivity.showDate, parse_json.parse_user вызывается в процессе работы MainActivity.showUsers.

2.4. Стратегия аттестационного тестирования

Аттестационное тестирование проводится для всей системы, что подразумевает выполнение действий в пользовательском интерфейсе. Для проведения тестирования в данном случае необходимо устройство под управлением ОС Андроид с API 22 или выше. При проведении аттестационного тестирования будет проверяться набор функциональностей системы, доступный пользователю:

1. авторизация пользователя в приложении и дальнейший переход в основное меню приложения
2. подключение внешней gps-антенны для получения координат устройства и периодический опрос gps-антенны на предмет координат, отправка координат устройства на сервер при его доступности, либо накопление пакета координат до момента доступности сервера;
3. возможность одновременного подключения нескольких usb-датчиков к устройству, не мешая при этом извлечению координат;
4. получение координат как периодическим опросом антенны, так и по “инициативе” антенны.

2.5. Критерии прохождения тестов

Тест считается пройдённым, если конечный результат соответствует ожидаемому результату. Ожидаемый результат - это результат

соответствующий ожидаемой работе функции (в соответствии с ее спецификацией). Ожидаемый результат должен присутствовать в описании каждого теста в плане тестирования.

2.6. Критерии приостановления тестирования

Тестирование должно быть приостановлено, если количество непройденных тестов превысит 10% от их общего количества. После этого тестировщик должен проанализировать результат и сообщить об ошибке, либо признать новый результат корректным (ожидаемым). После этого может быть принято решение об исправлении либо программы, либо результата.

2.7. Критерии возобновления работы

Необходимо заново начать тестирование при получении уведомления, что найденные при тестировании ошибки исправлены. Повторное тестирование необходимо начинать с самого начала.

3. Детальный план тестов

3.1. Блочные тесты

3.1.1. Модуль MainActivity

3.1.1.1. Функция login

Авторизация возможна для двух учетных записей:

- логин “alex” и паролем “123”
- логин “testuser” и паролем “12345”.

Все остальные учетные записи по умолчанию считаются некорректными и не могут авторизоваться в приложении.

1. **Тип**: общий

   Описание: авторизация пользователя с заведомо корректными данными

   Входные данные: login = “alex”, pwd = “123”

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение true

   Метод: белого ящика

   Выполняется: автоматически
2. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = “alex”, pwd = “1234”

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение false

   Метод: белого ящика

   Выполняется: автоматически
3. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = “”, pwd = “”

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение false

   Метод: белого ящика

   Выполняется: автоматически
4. **Тип**: общий

Описание: авторизация пользователя с заведомо некорректными данными

Входные данные: login = null, pwd = null

Выходные данные: boolean

Ожидаемый результат: возвращено значение false

Метод: белого ящика

Выполняется: автоматически

5. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = “”, pwd = null

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение false

   Метод: белого ящика

   Выполняется: автоматически

6. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = null, pwd = “”

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение false

   Метод: белого ящика

   Выполняется: автоматически

7. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = “alex”, pwd = “”

   Выходные данные: boolean

Ожидаемый результат: возвращено значение false

Метод: белого ящика

Выполняется: автоматически

8. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = “alex”, pwd = null

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение false

   Метод: белого ящика

   Выполняется: автоматически

9. **Тип**: общий

   Описание: авторизация пользователя с заведомо некорректными данными

   Входные данные: login = “”, pwd = “123”

   Выходные данные: boolean

   Ожидаемый результат: возвращено значение false

   Метод: белого ящика

   Выполняется: автоматически

10. **Тип**: общий

    Описание: авторизация пользователя с заведомо некорректными данными

    Входные данные: login = null, pwd = “123”

    Выходные данные: boolean

    Ожидаемый результат: возвращено значение false

    Метод: белого ящика

    Выполняется: автоматически

3.1.1.2. Функция changeStateGps()

1. **Тип**: общий

   Описание: смена состояния gps-антенны

   Входные данные: state = true

   Выходные данные: -

   Ожидаемый результат: индикатор состояния gps-антенны зеленый

   Метод: белого ящика

   Выполняется: автоматически
2. **Тип**: общий

   Описание: смена состояния gps-антенны

   Входные данные: state = false

   Выходные данные: -

   Ожидаемый результат: индикатор состояния gps-антенны красный

   Метод: белого ящика

   Выполняется: автоматически

3.1.1.3. Функция showDate

Внутри функции вызывается функция parse_date модуля parse_json. В зависимости результата, возвращенного от нее, либо отображается преобразованное значение, либо ничего. О корректных и некорректных входных данных функции parse_date указано в 3.1.3.2.

1. **Тип**: общий

   Описание: отображение текущей даты; в parse_date передается корректное значение - текущая дата

   Входные данные: -

   Выходные данные: -

   Ожидаемый результат: нередактируемое поле ввода даты содержит текущую дату в формате "день, dd месяц yyyy"

Метод: белого ящика

Выполняется: автоматически

2. **Тип**: общий

   Описание: отказ отображения даты; в parse_date передается некорректное значение

   Входные данные: -

   Выходные данные: -

   Ожидаемый результат: нередактируемое поле ввода даты является пустым

   Метод: белого ящика

   Выполняется: автоматически

3.1.1.4. Функция checkGps

1. **Тип**: общий

   Описание: смена состояния индикатора при подключении датчика

   Условия: подключить gps-датчик

   Входные данные: -

   Выходные данные: -

   Ожидаемый результат: индикатор состояния gps-антенны зеленый

   Метод: белого ящика

   Выполняется: вручную

2. **Тип**: общий

   Описание: смена состояния индикатора при отключении датчика

   Условия: отключить gps-датчик

   Входные данные: -

   Выходные данные: -

   Ожидаемый результат: индикатор состояния gps-антенны красный

   Метод: белого ящика

Выполняется: вручную

3. **Тип**: негативный

   Описание: отказ смена состояния индикатора при подключении не-gps-датчика

   Условия: подключить любое usb-устройство, не gps-датчик

   Входные данные: -

   Выходные данные: -

   Ожидаемый результат: индикатор состояния gps-антенны красный

   Метод: белого ящика

   Выполняется: вручную

3.1.1.5. Функция showUsers

Некоррекными входными данными считаются:

- входная строка неприводима к объекту класса JsonArray
- объект(ы) класса JsonObject содержит больше/меньше 3х полей
- объект(ы) класса JsonObject не содержи(а)т хотя бы одного из полей name,login,id
- поле id содержит не целочисленное значение

1. **Тип**: общий

   Описание: показ заведомо корректного списка пользователей

   Входные данные: строка

   “[{name:Alex1,login:alex,id:1}

   {name:Alex2,login:alex,id:2}

   {name:Alex3,login:alex,id:3}]”

   Выходные данные: -

   Ожидаемый результат: выпадающий список логинов содержит 3 значения

   Метод: белого ящика, выполняется автоматически

Выполняется: автоматически

2. **Тип**: общий

   Описание: отказ показа заведомо некорректного списка пользователей

   Входные данные: строка

   “uydruty”

   Выходные данные: -

   Ожидаемый результат: выпадающий список логинов не содержит значений

   Метод: белого ящика, выполняется автоматически

   Выполняется: автоматически

3. **Тип**: общий

   Описание: отказ показа заведомо некорректного списка пользователей

   Входные данные: строка

   “[{name:Alex1,id:1}

   {login:alex,id:2}

   {name:Alex3,login:alex}]”

   Выходные данные: -

   Ожидаемый результат: выпадающий список логинов не содержит значений

   Метод: белого ящика

   Выполняется: автоматически

4. **Тип**: общий

   Описание: отказ показа заведомо некорректного списка пользователей

   Входные данные: строка

   “[{name:Alex1,id:1}

{login:alex,id:2}

{name:Alex3,login:alex, id:3}]”

Выходные данные: -

Ожидаемый результат: выпадающий список логинов не содержит значений

Метод: белого ящика

Выполняется: автоматически

5. **Тип**: общий

   Описание: отказ показа заведомо некорректного списка пользователей

   Входные данные: строка

   “[{name:Alex1,login:alex,id:раз}

   {name:Alex2,login:alex,id:два}

   {name:Alex3,login:alex,id:три}]”

   Выходные данные: -

   Ожидаемый результат: выпадающий список логинов не содержит значений

   Метод: белого ящика

   Выполняется: автоматически

3.1.2. Модуль connect_external_gps

3.1.2.1. Функция openDevice

1. **Тип**: общий

   Описание: создание экземпляра класса UsbDevice

   Условия: подключить gps-датчик

   Входные данные: объект класса Intent

   Выходные данные: объект connection класса UsbDevice

Ожидаемый результат: device != null, в консоль ide выведено сообщение “Устройство присоединено”

Метод: белого ящика

Выполняется: вручную

4. **Тип**: общий

   Описание: создание экземпляра класса UsbDevice

   Условия: подключить gps-датчик, объект device != null класса UsbDevice

   Входные данные: -

   Выходные данные: объект connection класса UsbDevice

   Ожидаемый результат: device != null, в консоль ide выведено сообщение “Устройство присоединено”

   Метод: белого ящика

   Выполняется: вручную

5. **Тип**: негативный

   Описание: отказ создания экземпляра класса UsbDevice при подключении не-gps-датчика

   Условия: подключить любое usb-устройство, не gps-датчик, device = null

   Входные данные: -

   Выходные данные: объект connection класса UsbDevice

   Ожидаемый результат: device = null, в консоль ide выведено сообщение “Устройство не присоединено”

   Метод: белого ящика

   Выполняется: вручную

6. **Тип**: негативный

   Описание: отказ создания экземпляра класса UsbDevice при подключении не-gps-датчика

Условия: подключить любое usb-устройство, не gps-датчик, device != null

Входные данные: -

Выходные данные: объект connection класса UsbDevice

Ожидаемый результат: device = null, в консоль ide выведено сообщение “Устройство не присоединено”

Метод: белого ящика

Выполняется: вручную

3.1.2.2. Функция openConnection

7. **Тип**: общий

   Описание: открытие соединения с gps-датчиком

   Условия: подключить gps-датчик

   Входные данные: -

   Выходные данные: объект connection класса UsbConnection

   Ожидаемый результат: connection != null, в консоль ide выведено сообщение “Соединение открыто”

   Метод: белого ящика, выполняется вручную

   Выполняется:

8. **Тип**: общий

   Описание: переоткрытие соединения с gps-датчиком

   Условия: подключить gps-датчик, объект connection != null класса UsbConnection

   Входные данные: -

   Выходные данные: объект connection класса UsbConnection

   Ожидаемый результат: connection != null, в консоль ide выведено сообщение “Соединение открыто”

   Метод: белого ящика, выполняется вручную

Выполняется:

9. **Тип**: негативный

   Описание: отказ открытия соединения при подключении не-gps-датчика

   Условия: подключить любое usb-устройство, не gps-датчик, device = null

   Входные данные: -

   Выходные данные: объект connection класса UsbConnection

   Ожидаемый результат: connection = null, в консоль ide выведено сообщение “Соединение не открыто”

   Метод: белого ящика, выполняется вручную

   Выполняется:

10. **Тип**: негативный

    Описание: отказ открытия соединения при подключении не-gps-датчика при открытом соединении

    Условия: подключить любое usb-устройство, не gps-датчик, device = null, connection != null

    Входные данные: -

    Выходные данные: объект connection класса UsbConnection

    Ожидаемый результат: connection = null, в консоль ide выведено сообщение “Соединение не открыто”

    Метод: белого ящика, выполняется вручную

    Выполняется:

3.1.2.3. Функция parseData

Некорретной входной строкой считается:

- отсутствует широта/долгота/и то и другое, статус A [1]
- присутствует широта и долгота, статус V [2]

- строка содержит не 7 подстрок [3]
- подстрока широты/долготы/и та и другая не является преобразуемой в число [4]

1. **Тип**: общий

   Описание: извлечение координат из корректной строки

   Входные данные: строка

   GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A,A*73

   Выходные данные: -

   Ожидаемый результат: lat != null, long != null

   Метод: белого ящика

   Выполняется: автоматически

2. **Тип**: общий

   Описание: извлечение координат из корректной строки

   Входные данные: строка

   GPGLL,,N,11402.3279144,W,205412.00,A,V*73

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

   Метод: белого ящика

   Выполняется: автоматически

3. **Тип**: общий

   Описание: извлечение координат из корректной строки

   Входные данные: строка

   GPGLL,5107.0013414,N,,W,205412.00,A,V*73

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

   Метод: белого ящика

   Выполняется: автоматически

4. **Тип**: негативный

Описание: извлечение координат из некорректной строки [1]

Входные данные: строка GPGLL,,N,,W,205412.00,A,A*73

Выходные данные: -

Ожидаемый результат: lat = null, long = null

Метод: белого ящика

Выполняется: автоматически

5. **Тип**: негативный

   Описание: извлечение координат из некорректной строки [3]

   Входные данные: строка

   GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A,A*73,AAA

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

   Метод: белого ящика

   Выполняется: автоматически

6. **Тип**: негативный

   Описание: извлечение координат из некорректной строки

   Входные данные: строка

   GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A [3]

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

   Метод: белого ящика

   Выполняется: автоматически

7. **Тип**: негативный

   Описание: извлечение координат из некорректной строки [4]

   Входные данные: строка

   GPGLL,AAA,N,AAAA,W,205412.00,A,A*73

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

Метод: белого ящика

Выполняется: автоматически

8. **Тип**: негативный

   Описание: извлечение координат из некорректной строки [3]

   Входные данные: строка AAAAA

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

   Метод: белого ящика

   Выполняется: автоматически

9. **Тип**: негативный

   Описание: извлечение координат из некорректной строки [2]

   Входные данные: строка

   GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A,V*73

   Выходные данные: -

   Ожидаемый результат: lat = null, long = null

   Метод: белого ящика

   Выполняется: автоматически

3.1.3. Модуль parse_json

3.1.3.1. Функция parse_user

Некорректными входными данными считаются:

- входная строка неприводима к объекту класса JsonArray [1]
- объект(ы) класса JsonObject содержит больше/меньше 3х полей [2]
- объект(ы) класса JsonObject не содержи(а)т хотя бы одного из полей name,login,id [3]
- поле id содержит не целочисленное значение [4]

1. **Тип**: общий

Описание: создание списка экземпляров класса User из корректного json’a

Входные данные: строка

“[{name:Alex1,login:alex,id:1}

{name:Alex2,login:alex,id:2}

{name:Alex3,login:alex,id:3}]”

Выходные данные: список, содержащий 3 элемента

Ожидаемый результат: список объектов класса User содержит 3 элемента

Метод: белого ящика

Выполняется: автоматически

2. **Тип**: негативный

   Описание: отказ создания списка экземпляров класса User из некорректного json’a [1]

   Входные данные: строка “uydruty”

   Выходные данные: пустой список

   Ожидаемый результат: список объектов класса User не содержит элементов

   Метод: белого ящика

   Выполняется: автоматически

3. **Тип**: общий

   Описание: отказ создания списка экземпляров класса User из некорректного json’a [3]

   Входные данные: строка

   “[{name:Alex1,id:1}

   {login:alex,id:2}

   {name:Alex3,login:alex}]”

   Выходные данные: пустой список

Ожидаемый результат: список объектов класса User не содержит элементов

Метод: белого ящика

Выполняется: автоматически

4. **Тип**: негативный

   Описание: отказ создания списка экземпляров класса User из некорректного json’a [4]

   Входные данные: строка

   “[{name:Alex1,login:alex,id:раз}

   {name:Alex2,login:alex,id:два}

   {name:Alex3,login:alex,id:три}]”

   Выходные данные: пустой список

   Ожидаемый результат: список объектов класса User не содержит элементов

   Метод: белого ящика

   Выполняется: автоматически

5. **Тип**: негативный

   Описание: отказ создания списка экземпляров класса User из некорректного json’a [2]

   Входные данные: строка

   “[{name:Alex1,login:alex,id:1,id_:2}

   {name:Alex2,login:alex,id:2,id_:2}

   {name:Alex3,login:alex,id:3,id_:2}]”

   Выходные данные: пустой список

   Ожидаемый результат: список объектов класса User не содержит элементов

   Метод: белого ящика

   Выполняется: автоматически

3.1.3.2. Функция parse_date

Некорректными входными данными считаются:

- входная строка содержит больше/меньше четырех подстрок, разделенных пробелами [1]
- подстрока dd/MM/yyyy не приводима к целому положительному числу [2]

Тип: общий

Описание: парсинг коррекных данных

Входные данные: “*19 01 2018 Wed*”

Выходные данные: boolean

Ожидаемый результат: возвращено значение true

Метод: белого ящика

Выполняется: автоматически

Тип: негативный

Описание: парсинг некоррекных данных [1]

Входные данные: “*19 10 2018 Wed rr*”

Выходные данные: boolean

Ожидаемый результат: возвращено значение false

Метод: белого ящика

Выполняется: автоматически

Тип: общий

Описание: парсинг коррекных данных

Входные данные: “*19 20 2018 Wed*”

Выходные данные: boolean

Ожидаемый результат: возвращено значение false

Метод: белого ящика

Выполняется: автоматически

Тип: негативный

Описание: парсинг некоррекных данных [2]

Входные данные: “*19 rr 2018 Wed*”

Выходные данные: boolean

Ожидаемый результат: возвращено значение false

Метод: белого ящика

Выполняется: автоматически

3.2. Интеграционные

3.2.1. MainActivity (showDate) и parse_data (parse_date)

1. **Тип**: общий

   Описание: проверка корректности отображения даты и времени после обработки объекта Datetime модулем parse_data

   Алгоритм: запустить MainActivity, вызывается showDate, в parse_date

   Ожидаемый результат: отображается экран логина; в поле текущей даты отображается текущая дата в виде “dd, xx yyy zzzz”; в поле времени отображается текущее время в формате “hh:mm”

3.2.1. MainActivity (showUsers) и parse_json (parse_user)

1. **Тип**: общий

   Описание: проверка корректности отображения выпадающего списка пользователей после обработки модулем parse_data

   Алгоритм: запустить MainActivity, вызывается showUsers, в parse_user передается строка вида: “[{name:Alex1,login:alex1,id:1}{name:Alex2,login:alex2,id:2}{name:Alex3,login:alex3,id:3}]”

Ожидаемый результат: отображается экран логина; в выпадающем списке пользователей отображаются 3 пользователя: Alex1, Alex2, Alex3

3.2.3. MainActivity (checkGps) и connect_external_gps (parseData)

1. **Тип**: общий

   Описание: проверка корректности смены цвета индикатора после обработки строки с координатами модулем connect_external_gps

   Алгоритм: запустить MainActivity, подключить gps-антенну по usb, вызвать changeStateGps(false); вызывается checkGps, в parseData передается строка вида:

   “GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A,A*73”

   Ожидаемый результат: индикатор состояния gps-антенны меняет цвет с красного на зеленый

2. **Тип**: общий

   Описание: проверка корректности смены цвета индикатора после обработки строки с координатами без широты и долготы модулем connect_external_gps

   Алгоритм: запустить MainActivity, подключить gps-антенну по usb, вызвать changeStateGps(false); вызывается checkGps, в parseData передается строка вида:

   “GPGLL,,N,,W,205412.00,A,A*73”

   Ожидаемый результат: индикатор состояния gps-антенны не меняет цвет с красного на зеленый

3. **Тип**: общий

   Описание: проверка корректности смены цвета индикатора после обработки null строки модулем connect_external_gps

Алгоритм: запустить MainActivity, подключить gps-антенну по usb, вызвать changeStateGps(false); вызывается checkGps, в parseData передается строка null

Ожидаемый результат: индикатор состояния gps-антенны не меняет цвет с красного на зеленый

4. **Тип**: негативный

   Описание: проверка корректности смены цвета индикатора после обработки строки с координатами модулем connect_external_gps при неподключенной антенне

   Алгоритм: запустить MainActivity, вызвать changeStateGps(false); вызывается checkGps, в parseData передается строка null

   Ожидаемый результат: индикатор состояния gps-антенны не меняет цвет с красного на зеленый

5. **Тип**: негативный

   Описание: проверка корректности смены цвета индикатора после обработки строки с координатами без широты и долготы модулем connect_external_gps при неподключенной антенне

   Алгоритм: запустить MainActivity, вызвать changeStateGps(false); вызывается checkGps, в parseData передается строка вида: “GPGLL,,N,,W,205412.00,A,A*73”

   Ожидаемый результат: индикатор состояния gps-антенны не меняет цвет с красного на зеленый

6. **Тип**: негативный

   Описание: проверка корректности смены цвета индикатора после обработки null строки модулем connect_external_gps при неподключенной антенне

Алгоритм: запустить MainActivity, подключить gps-антенну по usb, вызвать changeStateGps(false); вызывается checkGps, в parseData передается строка null

Ожидаемый результат: индикатор состояния gps-антенны не меняет цвет с красного на зеленый

3.3. Аттестационные

Набор функций приложения, которые проверяются в аттестационном тестировании, определен в п. 2.4.

При аттестационном тестировании в качестве веб-сервера будет использоваться веб-сервер store.opti-soft.ru

3.3.1. Тесты для проверки функции 1 (Авторизация и переход в меню):

1. Авторизация с заведомо корректной парой логин-пароль

Тип теста: общий

Алгоритм:

1) в выпадающем списке пользователей выбрать “Testuser”

2) в поле ввода пароля ввести “12345”

3) нажать на кнопку “Вход”

Ожидаемый результат: успешная авторизация, переход в меню.

2. Авторизация с заведомо некорректной парой логин-пароль

Тип теста: негативный

Алгоритм:

1) в выпадающем списке пользователей выбрать “Testuser”

2) в поле ввода пароля ввести “123456”

3) нажать на кнопку “Вход”

Ожидаемый результат: неуспешная авторизация, уведомление “Неверный логин и/или пароль”.

Тесты для проверки функции 2 (подключение внешней gps-антенны):

3. Получение координа от gps-антенны

Тип теста: специальный

Алгоритм:

1) запустить приложение

2) подключить к устройству gps-антенну (в консоль ide выведено сообщение “Устройство присоединено”)

3) подождать 90 минут

Ожидаемый результат: индикатор состояния gps-антенны зеленого цвета

4. Накопление координат от gps-антенны

Тип теста: специальный

Алгоритм:

1) запустить приложение

2) убедиться, что файл sdcard0/logs/log_coordinates.txt не существует либо является пустым

3) отключить wifi/передачу данных по мобильной сети

4) подключить к устройству gps-антенну (в консоль ide выведено сообщение “Устройство присоединено”)

5) убедиться, что файл sdcard0/logs/log_coordinates.txt существует и не является пустым

6) включить wifi/передачу данных по мобильной сети

Ожидаемый результат: после появления wifi/передачи данных по мобильной сети файл sdcard0/logs/log_coordinates.txt будет очищен, накопленные координаты будут отправлены на сервер.

Тесты для проверки функции 3 (возможность одновременного подключения нескольких usb-датчиков к устройству, не мешая при этом извлечению координат):

5. Подключение нескольких usb-датчиков к устройству

Тип теста: общий

Алгоритм:

1) запустить приложение;

2) отключить wifi/передачу данных по мобильной сети

3) подключить к устройству gps-антенну (консоль ide выведено сообщение “Устройство присоединено”)

4) остановить к устройству usb-устройство другого типа, например, сканер штрих-кода (в консоль ide ничего не выведено)

5) убедиться, что файл sdcard0/logs/log_coordinates.txt не существует либо является пустым

Ожидаемый результат: файл sdcard0/logs/log_coordinates.txt существует и не является пустым.

Тесты для проверки функции 4 (возможность смены стратегии считывания координат):

6. получение координат как периодическим опросом антенны, так и по “инициативе” антенны

Тип теста: общий

Алгоритм:

1)в настройках приложения поменять тип считывания координат с периодического на “инициативный”

2) запустить приложение

3) убедиться, что файл sdcard0/logs/log_coordinates.txt не существует либо является пустым

4) отключить wifi/передачу данных по мобильной сети

5) подключить к устройству gps-антенну (в консоль ide выведено сообщение “Устройство присоединено”)

Ожидаемый результат: файл sdcard0/logs/log_coordinates.txt существует и не является пустым.

3.4. Пример реализации тестов

Далее приведен пример реализации нескольких автоматических блочных тестов для функции parseData содуля connect_external_gps. Код теста написан на языке Java. Функция assertEquals сравнивает результат работы функции (в данном случае запись значений в переменные класса) с ожидаемыми значениями в тесте.

Остальные автоматические тесты реализованы аналогично.

```
package ru.opti_soft.sawmill_warehouse_android;
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class Test {
  @Test
  public void parseData_isCorrect_0() throws Exception {
     connect_external_gps.lat = null;
     connect_external_gps.long = null;
    connect_external_gps.isGpsAvailable = false;

connect_external_gps.parseData("GPGLL,5107.0013414,N,11402.3279144,W,
205412.00,A,A*73");
     assertEquals(true, connect_external_gps.lat!=null &&
connect_external_gps.long!=null && connect_external_gps.isGpsAvailable ==
true);
  }

  @Test
  public void parseData_isCorrect_1() throws Exception {
```

```
        connect_external_gps.lat = null;
        connect_external_gps.long = null;
       connect_external_gps.isGpsAvailable = false;

connect_external_gps.parseData("GPGLL,,N,11402.3279144,W,205412.00,A,A
*73");
        assertEquals(true, connect_external_gps.lat==null &&
connect_external_gps.long==null && connect_external_gps.isGpsAvailable
== false);
    }

    @Test
    public void parseData_isCorrect_2() throws Exception {
        connect_external_gps.lat = null;
        connect_external_gps.long = null;
       connect_external_gps.isGpsAvailable = false;

connect_external_gps.parseData("GPGLL,5107.0013414,N,,W,205412.00,A,A*
73");
        assertEquals(true, connect_external_gps.lat==null &&
connect_external_gps.long==null && connect_external_gps.isGpsAvailable
== false);
    }
}
```

3.5. Оценка покрытия кода

Ввиду сложности современного программного обеспечения становится невозможно добиться 100% покрытия тестами. Кроме того, некоторые

тесты невозможно автоматизировать, что сокращает долю кода, покрытого автоматическими тестами (хотя код при этом протестирован). Расчет доли покрытого тестами кода проводится по формуле Tcov = Ltc/Lcode * 100%, где Tcov – доля кода, покрытого тестами, Ltc – количество строк кода, покрытого тестами, Lcode – общее количество строк кода. При тестировании покрытия программы кроме общей доли покрытия тестами вычислялась также доля кода, покрытого тестами, который содержится в функциях, тестируемых автоматически. В таблице приведен результат вычисления тестового покрытия.

Модуль	Строк кода	Общее покрытие	Покрытие выделенных функций
MainActivity	241	43%	88%
connect_external_gps	528	78%	91%
parse_json	467	21%	93%

4. Журнал тестирования

Номера тестов	Объект	Количество тестов	Количество ошибок	Дата проведения
3.1.1.1.1, 3.1.1.1.2, 3.1.1.1.3, 3.1.1.1.4, 3.1.1.1.5, 3.1.1.1.6, 3.1.1.1.7, 3.1.1.1.8, 3.1.1.1.9,	MainActivity. login	10	0	2.12.2018

3.1.1.1.10				
3.1.1.2.1, 3.1.1.2.2	MainActivity.changeStateGps	2	0	2.12.2018
3.1.1.3.1, 3.1.1.3.2	MainActivity.showDate	2	0	2.12.2018
3.1.1.4.1, 3.1.1.4.2, 3.1.1.4.3	MainActivity.checkGps	3	0	2.12.2018
3.1.1.5.1, 3.1.1.5.2, 3.1.1.5.3, 3.1.1.5.4, 3.1.1.5.5	MainActivity.showUsers	5	0	2.12.2018
3.1.2.1.1, 3.1.2.1.2, 3.1.2.1.3, 3.1.2.1.4	connect_external_gps.openDevice	4	0	5.12.2018
3.1.2.2.1, 3.1.2.2.2, 3.1.2.2.3, 3.1.2.2.4	connect_external_gps.openConnection	4	0	5.12.2018
3.1.2.3.1, 3.1.2.3.2, 3.1.2.3.3, 3.1.2.3.4, 3.1.2.3.5, 3.1.2.3.6, 3.1.2.3.7, 3.1.2.3.8, 3.1.2.3.9,	connect_external_gps.parseData	9	3 Отчет об ошибке №1, №2, №3	5.12.2018

3.1.3.1.1, 3.1.3.1.2, 3.1.3.1.3, 3.1.3.1.4, 3.1.3.1.5,	parse_json.parse_user	5	0	18.12.2018
3.1.3.2.1, 3.1.3.2.2, 3.1.3.2.3, 3.1.3.2.4	parse_json.parse_date	4	0	18.12.2018

5. Журнал ошибок

5.1. Отчет об ошибке №1

Объект тестирования: функция parseData из модуля connect_external_gps

Условия: планшет Archos 116 neon, android api 25

Приоритет: низкий

Описание: извлечение координат из некорректной строки типа [3]

Входные данные: строка

GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A,A*73,AAA

Выходные данные: -

Ожидаемый результат: lat = null, long = null

Фактический результат: lat != null, long != null

5.2. Отчет об ошибке №2

Объект тестирования: функция parseData из модуля connect_external_gps

Условия: планшет Archos 116 neon, android api 25

Приоритет: низкий

Описание: извлечение координат из некорректной строки типа [3]

Входные данные: строка

GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A

Выходные данные: -

Ожидаемый результат: lat = null, long = null

Фактический результат: lat != null, long != null

Воспроизводимость: всегда

Дата проведения: 5.12.2018.

5.3. Отчет об ошибке №3

Объект тестирования: функция parseData из модуля connect_external_gps

Условия: планшет Archos 116 neon, android api 25

Приоритет: низкий

Описание: извлечение координат из некорректной строки типа [2]

Входные данные: строка

GPGLL,5107.0013414,N,11402.3279144,W,205412.00,A,V*73

Выходные данные: -

Ожидаемый результат: lat = null, long = null

Фактический результат: lat != null, long != null

Воспроизводимость: всегда

Дата проведения: 5.12.2018.

6. Результаты

В ходе тестирования были обнаружены 3 ошибки в функции parseData. При проведении аттестационного тестирования ошибок обнаружено не было. Найденные ошибки были устранены в рамках отладки, последовавшей за тестированием.

Таким образом, после проведения тестирования можно сказать, что приложение соответствует предъявленным к нему требованиям.