Отчет по дисциплине

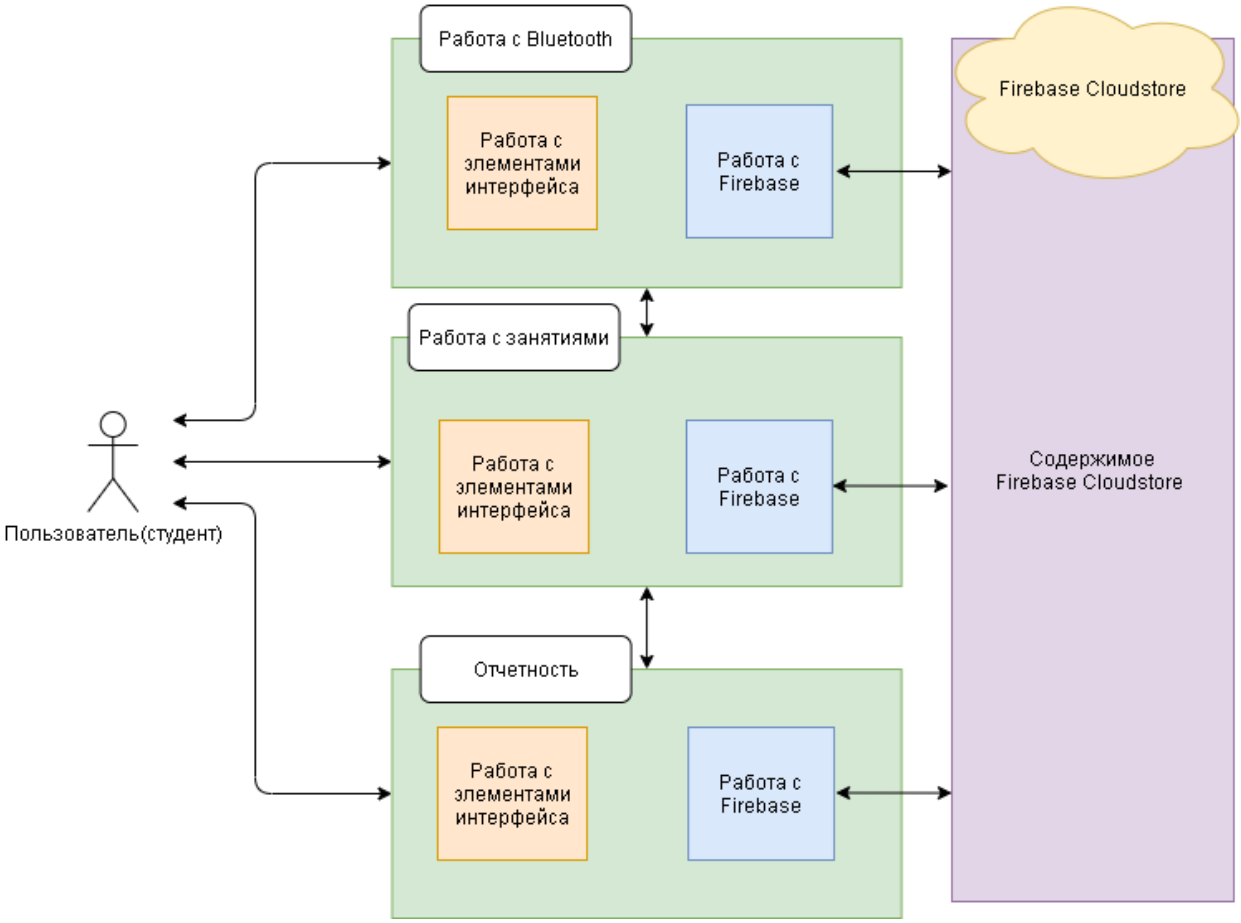
«Методы тестирования программного обеспечения»

Кузнецовой Ксении

Гр. 22407

1) Объект тестирования.

Мобильное приложение на операционной системе Android, разрабатываемое для учета посещаемости студентов групп ПетрГУ и предназначенное для старост групп университета. Все данные хранятся в облачной базе Cloud Firestore. Будет тестироваться только «Работа с занятиями».



Пользовательские требования для «Работы с занятиями» (Базовые возможности приложения для контроля посещаемости)

(a) Авторизация в системе и выход из системы.
(b) Регистрация пользователя
(c) Добавление занятий в расписание своей группы. Подзадачи:
    1. Выбор даты. Изначально стоит время и дата открытия формы добавления занятия. Выбирается сначала день, потом время.
    2. Выбор предмета.
    3. Выбор преподавателя.
    4. Отметка студентов на занятии.

(c) Редактирование и удаление созданных занятий

2. Перечень функций тестируемого объекта:

1. Авторизация пользователя по логину и паролю для доступа к дальнейшей работе с системой.
2. Выход из системы.
3. Регистрация нового пользователя в системе.
4. Установка даты проведения создаваемого занятия
5. Установка даты отображаемых занятий
6. Установка даты отображаемых занятий по умолчанию
7. Установка даты и времени создаваемых занятий по умолчанию
8. Установка времени проведения создаваемого занятия
9. Формирование списка преподавателей путем запроса к Cloud Firestore

10. Формирование списка предметов путем запроса к Cloud Firestore с фильтром по группе
11. Формирование списка студентов путем запроса к Cloud Firestore с фильтром по группе
12. Сохранение выбора предмета
13. Сохранения выбора преподавателя
14. Сохранение списка посещаемости группы
15. Сохранение посещаемости занятия в Cloud Firestore
16. Сохранение параметров занятия в Cloud Firestore
17. Внесение изменений в Cloud Firestore при редактировании занятия.
18. Обновление информации в приложении при изменении данных в Firebase.
19. Обновление расписания при изменении даты отображения занятий путем запроса к Cloud Firestore
20. Удаление созданных занятий путем запроса к Cloud Firestore.(в т.ч. удаление данных о посещаемости этого занятия)
21. Подтверждение всех потенциально опасных действий пользователя

2. Стратегия тестирования.

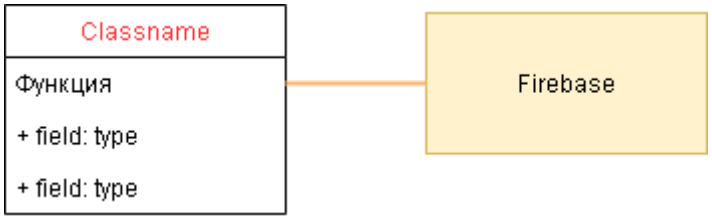
2.1. Описание структуры объекта тестирования и связей внутри объекта тестирования (архитектура).

Красным цветом отмечены не реализованные функции и классы, которые не будут тестироваться.

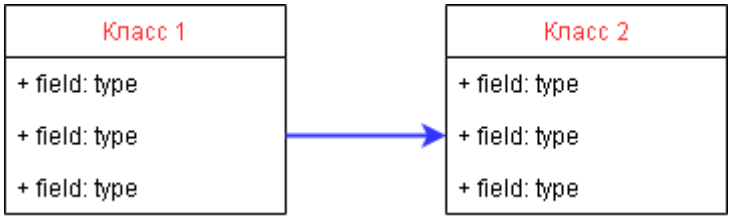
Оранжевым цветом отмечены реализованные классы и функции, но которые не будут тестироваться в связи с отсутствием необходимости их тестирования.

Обозначения:

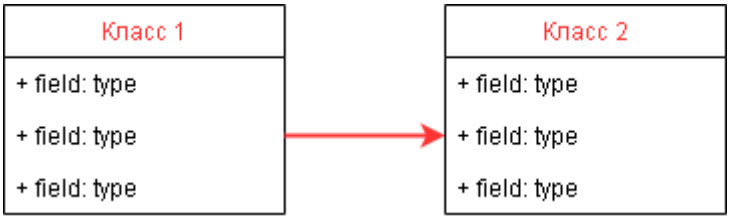
1. Функция создает запрос и/или принимает результат запроса от Firebase

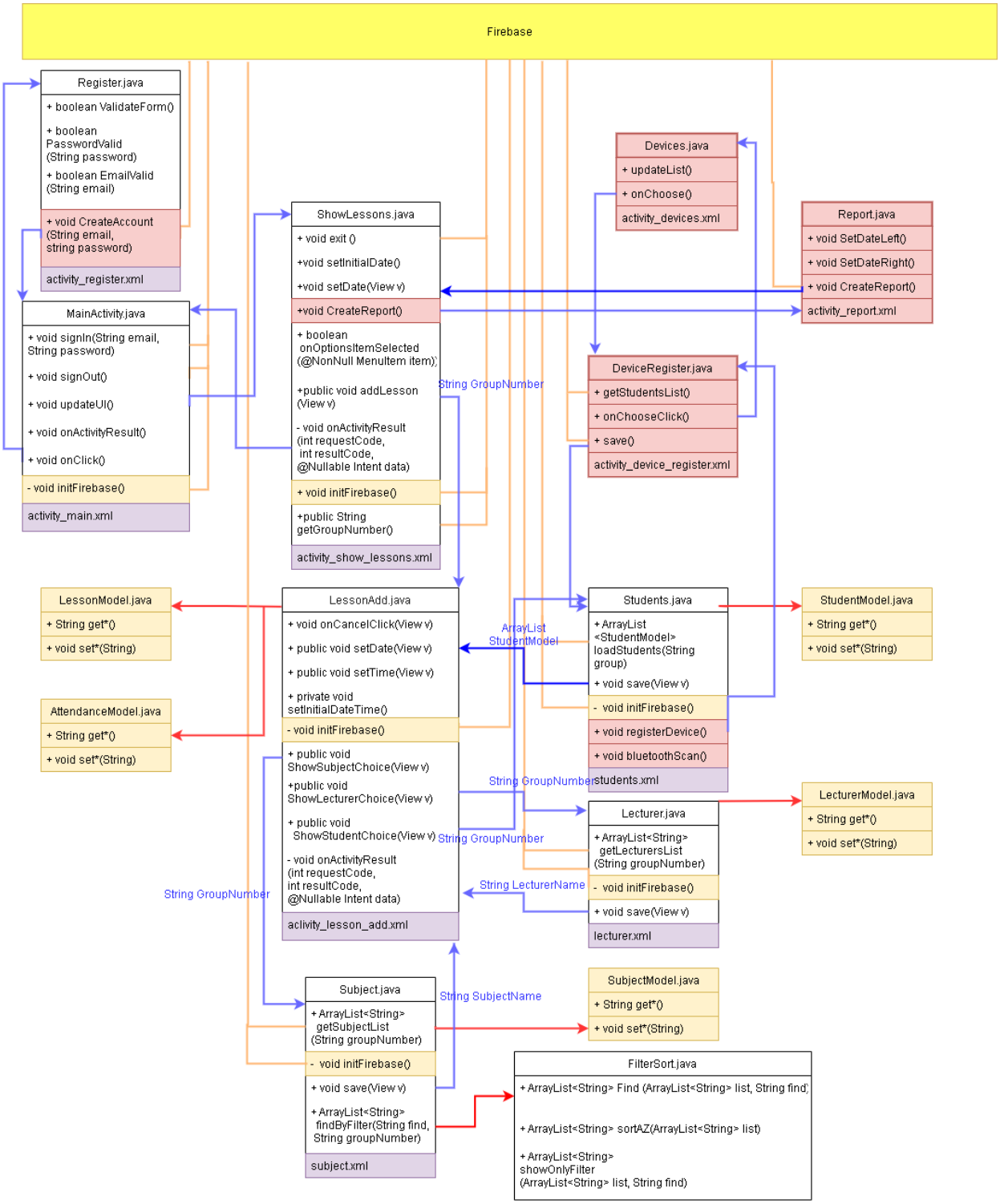


2. Функция Класса 1 производит переход к экрану Класса 2(создает intent). Если при создании intent передаются данные помимо кода запроса, эти данные указаны на стрелке



3. Класс 1 в своем теле использует экземпляр(-ы) Класса 2





Описание функций и классов:

Каждый класс активности содержит функцию OnCreate(), которая вызывается при создании экземпляра класса. В ней инициализируются элементы интерфейса, записываются в переменные значения, переданные из предыдущего класса (если таковые имеются), инициализируется Firebase. (Вызывается функция initFirebase если таковая определена)

1) MainActivity.java
   Класс окна авторизации. Содержит в себе функции
   void signIn(String email, String password) - функция входа по логину и паролю. Входит в аккаунт, созданный в облачной базе, т.е обновляет пользователя с null на текущего, если авторизация прошла успешно и выводит сообщение об ошибке и оьнуляет пользователя если безуспешно.
   void signOut() обнуляет текущего пользователя. Понадобится далее, если пользователь на

главном экране нажмет "Выйти"
void updateUI() проверяет текущего пользователя, если он не null, перенаправляет на главную страницу приложения
onActivityResult() получает данные и коды возвратов от других страниц.
onClick - обработчик кнопок. Если нажата кнопка зарегистрироваться, то перенаправляет на страничку регистрации, если нажата кнопка входа, берет данные с формы и вызывает функцию signIn с этими аргументами
void initFIrebase() –здесь и в других классах – функция, которая инициализирует текущий экземпляр приложения в Firebase и получает экземпляр базы для дальнейшей работы с ней в данном классе.(не тестируется, т.к. не содержит функциональности, которую следует проверять)

1) AttendanceModel.java, LecturerModel.java, LessonModel.java, SubjectModel.java, StudentModel.java – классы-модели
String get* () получает атрибут модели
void set*(String *) устанавливает атрибут модели.
2) Lecturer.java
Класс окна выбора преподавателя.
void save() создаёт данные для передачи в класс LessonAdd (передаёт данные о выбранном преподавателе и закрывает текущее окно)
ArrayList<String>getLecturersList()- делает запрос к базе с преподавателями, составляет список из полученного запроса.
3) Subject.java
Класс окна выбора преподавателя.
void save() создаёт данные для передачи в класс LessonAdd (передаёт данные о выбранном предмете и закрывает текущее окно)
ArrayList<String> getSubjectList(String groupNumber)- делает запрос к базе с предметами, составляет список из полученного запроса.
public ArrayList<String> findByFilter(String find, String groupNumber) – принимает на вход введенную в строку поиска строку и возвращает отфильтрованный и отсортированный список предметов.

4) LessonAdd.java
void ShowLecturerChoice(view V) - показывает окно выбора преподавателя с ожиданием результата(у каждого такого запроса есть requestCode) и передачей номера группы.
void ShowSubjectChoice(view V) - показывает окно выбора предмета с ожиданием результата(у каждого такого запроса есть requestCode)
void ShowStudents () показывает окно отметки посещаемости
setInitialDateTime() - устанавливают текущие дату и время в поле ввода
void onActivityResult() - проверяет коды возврата и работает с полученными данными (устанавливает в поля ввода)
void CreateLesson() сохраняет занятие в firebase, присваивает ему уникальный идентификатор, возвращает в главное меню.
5) Register.java
boolean PasswordValid(String password) – проверяет новый пароль на соответствие требованиям (требования указаны в п.2.2)
boolean EmailValid(String email) – проверяет корректность введенного логина
boolean validateForm() - проверяет правильность/ заполненность формы
void CreateAccount(String email, String password) создаёт в firebase аккаунт и возвращает на главный экран.

8) ShowLessons.java- Главный экран приложения.

void setInitialDate() – в поле даты текущую дату.

addLesson() вызывается при нажатии на кнопку добавить занятия, переход на форму добавления занятия(передается номер группы)

void exit() функция выхода из учётной записи

void CreateReport() переходит к форме создания отчета

String getGroupNumber() – путем запроса к firebase получает номер группы текущего пользователя.

void onActivityResult (…) – обрабатывает код и данные возврата

10) Students.java

Окно отметки посещаемости.
ArrayList <StudentModel>loadStudents(String group) - загрузка из firebase в список студентов
void save(View v) - сохранение списка посещаемости, переход обратно на форму добавления занятия с передачей списка.
void registerDevice() – переход к окну регистрации устройства.
void bluetoothScan() – сканирование и отметка студентов, чьи устройства обнаружены поблизости

11)Report.java – окно создания отчета.

SetDateLeft()- установить дату начала отчета
SetDateRight() – дата окончания отчета
CreateReport() - создаёт отчёт

12) DeviceRegister.java – форма привязки устройства к студенту группы

+ getStudentsList() – получение списка студентов группы
+ onChooseClick() – переход к окну выбора устройства.
+ save() – привязка mac адреса устройства к студенту группы и переход обратно к окну фиксирования посещаемости.

13) Devices.java – окно списка устройств, которые были обнаружены Bluetooth

updateList() – обновление списка устройств

onChoose() – сохранение выбора устройства , переход на форму назад

14) FilterSort.java

public ArrayList<String> Find (ArrayList<String> list, String find) – вызывается при изменении текста в строке поиска предметов

public ArrayList<String> sortAZ(ArrayList<String> list) – сортирует список по алфавиту

public ArrayList<String> showOnlyFilter(ArrayList<String> list, String find) – фильтрует список по введенному значению

2.2. Описание стратегии блочного тестирования

Модульные тесты пишутся и запускаются в среде разработки Android Studio. Для написания модульных тестов используется фреймворк JUnit. Также некоторые блочные тесты написаны как инструментальные, использующие запуск программы на устройстве. Для таких тестов используется фреймворк Espresso. Результаты выполнения (успех/неудача), а также данные об ошибках выводятся в консоль приложения. Метод проведения: автоматизированное тестирование.

Модульные тесты будут описаны для следующих функций:

Название ф-ии	Описание	Входные данные	Выходные данные
void signIn(String email, String password)	Функция входа в учетную запись Firebase и в приложение.	Строка логин, Строка пароль	- Изменяет текущего пользователя firebase.
void signOut()	Функция выхода из приложения	-	- Изменяет текущего пользователя firebase на null
ArrayList<String>getLecturersList()	Функция получения списка преподавателей	-	Список преподавателей
ArrayList<String> getSubjectList(String groupNumber)	Функция получения списка предметов группы	Номер группы	Список дисциплин группы
void CreateLesson()	Формирует и отправляет запрос для создания занятия и записи посещаемости студентов в Firebase по данным, заполненным в полях и полученном из окна отметки посещаемости массиву посещаемости.	Преподаватель Время Дата Предмет МассивПосещаемости	- Создаются записи в Firebase
boolean PasswordValid(String password)	Проверяет на соответствие пароля нового пользователя заданным требованиям: пароль должен содержать только латинские буквы и цифры. В пароле д.б. хотя бы одна заглавная, одна строчная буква и хотя бы одна цифра, его длина должна быть не менее 8 знаков, но не более 20	Строка – желаемый пароль	True – пароль валидный False – пароль не валидный
boolean EmailValid(String email)	Проверяет валидность введенной эл. Почты. Валидная строка эл. Почты : name@domain, где name – набор латинских букв, цифр, точки, тире, подчеркивания domain – домен почтового сервиса, состоит из слов из латинских букв, цифр, точки, тире, разделенных как минимум одной точкой.	Строка – email будущего пользователя	True – email валидный False – email не валидный
boolean validateForm()	Проверяет заполненность и правильность заполнения полей	Данные из полей email, password и password_2	True – форма заполнена верно False – форма заполнена не

	формы регистрации, сравнивает значения в поле пароля и в поле повторения пароля. Форма заполнена верно, если в нее введены валидные логин и пароль и поля пароля и подтверждения пароля совпадают.		верно
String getGroupNumber()	Получает номер группы текущего пользователя	- Использует экземпляр FirebaseAuth для получания и дальнейшего использования idтеккущего пользователяя	Идентификатор группы текущего пользователя
ArrayList <StudentModel>loadStudents(String group)	Запрашивает и возвращает список студентов группы	Идентификатор(номер) группы	Список Моделей студентов для каждого студента группы
public ArrayList<String> sortAZ(ArrayList<String> list)	Сортирует список строк от А до Я	Список для сортировки	Отсортированный список
public ArrayList<String> showOnlyFilter(ArrayList<String> list, String find)	Фильтрует список. Оставляет только те элементы, в которых имеются вхождения шаблона	Список для фильтрации Шаблон - строка	Список значений с сохранение порядка после применения фильтра

Выбраны функции, не являющиеся обработчиками нажатий и функциями работы с элементами интерфейса, а также не включают в себя взаимодействие двух и более модулей.

2.2. Описание стратегии интеграционного тестирования

Интеграционные тесты пишутся и запускаются в среде разработки Android Studio. Интеграционные тесты полностью являются инструментальными (работают непосредственно с экземпляром программы) .Для таких тестов используется фреймворк Espresso. Результаты выполнения (успех/неудача), а также данные об ошибках выводятся в консоль приложения. В тестах, включающих в себя функции работы с intent (осуществляющих переходы между окнами) проверяются только те переходы между окнами, которые передают какую-либо информацию.

При проверке взаимодействия модулей Subject.java и FilterSort.java происходит следующая последовательность вызовов функций:

Начало теста->Вызов функции public ArrayList<String> findByFilter (Входные параметры – тестовая строка) (Subject.java) -> вызов функции ArrayList<String> getSubjectList(String groupNumber)->вызов функции public ArrayList<String> Find () (FilterSort.java) -> вызов фукции public ArrayList<String> sortAZ(ArrayList<String> list) -> вызов функции public ArrayList<String> showOnlyFilter(ArrayList<String> list, String find) -> конец теста

При проверке взаимодействия модулей LessonAdd.java и Subject.java происходит следующая последовательность вызовов функций:

Начало теста-> Вызов функции AddLesson () (ShowLessons.java)-> Вызов функции OnCreate()(LessonAdd.java) ->Вызов функции onCreate()(Register.java)-> конец теста

При проверке взаимодействия модулей ShowLessons.java и LessonAdd.java происходит следующая последовательность вызовов функций:

> Начало теста->Вызов функции void ShowSubjectChoice(view V) (LessonAdd.java) -> Вызов функции OnCreate() (Subject.java) -> конец теста

Аналогичная последовательность вызова функция и при взаимодействии модулей LessonAdd.java и Subject.java, LessonAdd.java и Students.java.

При проверке взаимодействия модулей Subject.java и LessonAdd.java происходит следующая последовательность вызовов функций:

> Начало теста-> Вызов функции void save() (Subject.java)-> Вызов функции void onActivityResult() (LessonAdd.java) -> конец теста

Аналогично при взаимодействии модулей Lecturer.java и LessonAdd.java, LessonAdd.java и ShowLessons.java

При проверке взаимодействия модулей ShowLessons.java и MainActivity.java происходит следующая последовательность вызовов функций:

> Начало теста-> Вызов функции void exit() (ShowLessons.java)-> Вызов функции void onActivityResult(MainActivity.java) -> вызов функции SignOut() (MainActivity.java) -> вызов функции updateUI() (MainActivity.java) -> конец теста

2.3. Описание стратегии аттестационного тестирования

Аттестационное тестирование будет проводиться методом «живого человека». В роли такого человека выступает сам автор тестирования.

Также возможно использование сценарного тестирования. Для написания тестов используется фреймворк Espresso. В разделе «Реализация» размещены несколько примеров использования тестовых сценариев, написанных при помощи фреймворка Espresso.

Тест считается пройденным, если ожидаемый результат совпадает с фактическим результатом. В противном случае тест считается не пройденным.

2.4. Стратегия специального тестирования

Последний этап тестирования — нагрузочное тестирование. Проверяет работу системы при большом объеме одновременно совершающихся запросов к Cloud Firestore.

2.5 Критерий приостановки тестов

Тестирование должно быть приостановлено, если при прохождении теста было потрачено более 1 минуты (для инструментальных тестов) и более 5 секунд (для юнит тестов). Такой тест считается не удавшимся.

2.6 Критерий возобновления тестов.

Тестирование возобновляется после исправления ошибок, выявленных при предыдущем тестировании.

3. Детальный план тестов

3.1. Перечень модульных тестов.

	Проверяемая функция/модуль	Тип теста	Описание теста	Входные данные	Ожидаемый результат
Б - 1	Register.java public boolean emailValid(String email)	Позитивный	Тестируется функция проверки валидности email – адреса. На вход дается валидный адрес эл.почты	Строка kkuzneco@cs.karelia.ru	Функция возвращает true
Б - 2	Register.java public boolean emailValid(String email)	Позитивный	Тестируется функция проверки валидности email – адреса. На вход дается валидный адрес эл.почты. Отличается от Б-1 форматом домена	Строка tester@yandex.ru	Функция возвращает true
Б - 3	Register.java public boolean emailValid(String email)	Позитивный	Тестируется функция проверки валидности email – адреса. На вход дается не валидный адрес эл.почты.	Строка tester@cs	Функция возвращает false
Б - 4	Register.java public boolean emailValid(String email)	Позитивный	Тестируется функция проверки валидности email – адреса. На вход дается не валидный адрес эл.почты. Отличается от Б-3 отсутствием знака @	Строка test	Функция возвращает false
Б - 5	MainActivity.java void signIn(String email, String password)	Позитивный	Тестируется функция аутентификации в приложении по существующим логину/паролю, соответствующим специально созданному тестовому пользователю	Тестовая пара Логин: tester@test.ru Пароль: 123456 (такой пользователь существует в системе)	id текущего пользователя базы становится равно id тестового пользователя (Nd40A9yA9jdHZ6Zsd)
Б - 6	MainActivity.java void signIn(String email, String password)	Негативный	Тестируется функция аутентификации в приложении по не существующей паре логин/пароль	Тестовая пара Логин: tester@test.ru Пароль: 123 (такой пользователь существует в системе)	id текущего пользователя базы не меняется и равно null
Б - 7	MainActivity.java void signIn(String email, String password)	Краевой	Проверка функции аутентификации в приложении без ввода логина и пароля	На вход функции дается пустая пара логин/пароль	id текущего пользователя базы не меняется и равно null
Б - 8	MainActivity.java void signOut()	Позитивный	Проверка функции выхода из учетной записи.	Текущий пользователь системы – тестовый(id не null)	id текущего пользователя базы становится равно null
Б - 9	Register.java boolean PasswordValid(S	Позитивный	Тестируется функция проверки валидности пароля. Вводится	На вход функции строка «123»	false

	tring password)		пароль недостаточной длины		
Б - 10	boolean PasswordValid(String password)	Позитивный	Тестируется функция проверки валидности пароля. Вводится пароль не содержащий ни одной заглавной буквы	На вход функции строка «123456fbb»	false
Б -11	boolean PasswordValid(String password)	Позитивный	Тестируется функция проверки валидности пароля. Вводится пароль, содержащий недопустимые символы	На вход функции строка «Qe1234#»	false
Б -12	boolean PasswordValid(String password)	Позитивный	Тестируется функция проверки валидности пароля. Вводится допустимый пароль	На вход функции строка «Qef12345dd»	true
Б -13	ShowLessons.java public String getGroupNumber()	Позитивный	Проверка функции получения номера группы пользователя.	Производится вход в систему через данные тестового пользователя.	Функция возвращает номер группы тестового пользователя
Б - 14	ShowLessons.java public String getGroupNumber()	Негативный	Проверка функции получения номера группы пользователя при условии, что не был осуществлен вход в систему	Вызов функции без предварительного входа в систему	Функция возвращает null
Б - 15	Subject.java getSubjectList(String groupNumber)	Позитивный	Проверка получения списка предметов по номеру группы. На вход дается существующая в системе группа с существующими занятиями	Номер(идентификатор) группы : 22000	Возвращенный список не пуст и содержит как минимум название тестового предмета, созданного в базе
Б-16	Subject.java getSubjectList(String groupNumber)	Негативный	Проверка получения списка предметов по номеру группы для несуществующего номера группы.	Идентификатор группы «00000»	Возвращенный список равен null
Б – 17	Register.java public boolean emailValid(String email)	Негативный	Тестируется функция проверки валидности email – адреса. На вход дается пустая строка.	Пустая строка	Функция возвращает false
Б - 18	Lecturer.java ArrayList<String> getLecturersList()	Позитивный	Тестируется функция проверки валидности email – адреса. На вход дается пустой адрес эл.почты.	Пустая строка	Возвращенный список равен null
Б - 19	boolean PasswordValid(String password)	Позитивный	Тестируется функция проверки валидности пароля. Вводится пароль, содержащий количество символов больше, чем допустимо.	На вход функции строка «d6Fggdjl27i29uytTTTTYndjdsnsllndgjdn27992920»	false
Б – 20	Subject.java getLectirertList(String	Позитивный	Проверка получения списка преподавателей.	-	Возвращается список всех добавленных в систему преподавателей.

	groupNumber)				
Б-21	LessonAdd.java void CreateLesson()	Позитивный	Проверка функции создания занятия и посещаемости для $n студентов группы, у каждого из которых идентификатор равен $stud_id, а статус посещения - $status_$n.	Поля формы добавления занятия заполняются данными: Предмет: $testSubject Преподаватель: $TestLecturer Дата: $testDate Время:$testTime	В коллекцию «lessons» в Firebase добавляется новый документ с идентификатором $id с полями: subject: $testSubject lecturer: $TestLecturer date: $testDate time:$testTime В коллекцию «attendance» добавляется $n документов с полями: lesson_id: $id status: $status_$n student_id: $stud_id
Б-22	LessonAdd.java void CreateLesson()	Негативный	Проверка функции создания занятия и посещаемости для $n студентов группы, у каждого из которых идентификатор равен $stud_id, а статус посещения - $status_$n. При не заполненном поле «Предмет».	Поля формы добавления занятия заполняются данными: Преподаватель: $TestLecturer Дата: $testDate Время:$testTime Предмет: (не заполнено)	Занятие не создано. Выведено сообщение о том, что не все поля заполнены.
Б-23	Register.java boolean validateForm()	Позитивный	Тестируется функция проверки правильности заполнения формы регистрации. На вход даются правильно заполненные поля.	Поля формы добавления занятия заполняются данными: Логин: Tester100@test.ru Пароль: sTg78KKf Подтверждение пароля: sTg78KKf	true
Б-24	Register.java boolean validateForm()	Позитивный	Тестируется функция проверки правильности заполнения формы регистрации. На вход даются разные строки в полях «Пароль» и «Подтверждение пароля».	Поля формы добавления занятия заполняются данными: Логин: Tester100@test.ru Пароль: sTg78KKf Подтверждение пароля: sTg78KKd	false
Б-25	ArrayList <StudentModel>loadStudents(String group)	Позитивный	Проверяется функция получения и заполнения списка студентов существующей тестовой группы.	Строка – идентификатор существующей группы (22407)	Полный список моделей студента, заполненный данными обо всех студентах группы.
Б-26	ArrayList <StudentModel>loadStudents(String group)	Негативный	Проверяется функция получения и заполнения списка студентов несуществующей тестовой группы.	Строка – идентификатор существующей группы (00000)	Полученный список равен null
Б-27	public ArrayList<String> sortAZ(ArrayList	Позитивный	Проверяется функция сортировки списка строк в порядке возрастания	Список строк ArrayList<String>, состоящий из русских слов	Упорядоченный по алфавиту список значений

	<String> list)				
Б-28	public ArrayList<String> sortAZ(ArrayList <String> list)	Позитивный	Проверяется функция сортировки списка строк в порядке возрастания	Список строк ArrayList<String>, состоящий из английских слов	Упорядоченный по алфавиту список значений
Б-29	public ArrayList<String> sortAZ(ArrayList <String> list)	Позитивный	Проверяется функция сортировки списка строк в порядке возрастания	Список строк ArrayList<String>, состоящий из русских и английских слов	Упорядоченный по алфавиту список значений: сначала английские слова по алфавиту, затем - русские
Б-30	public ArrayList<String> showOnlyFilter(ArrayList<String> list, String find)	Позитивный	Проверяется фильтрация строк по вхождению шаблона.	Список Алгебра, Анализ требований, Тестирование ПО, Философия, История Шаблон: «А»	Список: Алгебра, Анализ требований.

3.2. Перечень интеграционных тестов

	Взаимодействующие модули, функция(-и)	Описание теста	Входные данные	Ожидаемый результат
И - 1	Students.java и LessonAdd.java + public void ShowStudentChoice(View v)	Проверяется передача данных при переходе из формы добавления занятия к отметке студентов.	Номер группы, который передается при создании intent устанавливается равным группе тестового пользователя	После создания экземпляра класса Students.java его поле GroupNumber становится равным переданному значению
И - 2	Students.java и LessonAdd.java + public void ShowStudentChoice(View v)	Проверяется передача данных при переходе из формы добавления занятия к отметке студентов.	Номер группы, который передается при создании intent устанавливается равным несуществующему в базе номеру.	После создания экземпляра класса Students.java его поле GroupNumber =null
И - 3	Students.java и LessonAdd.java + public void ShowStudentChoice(View v) ArrayList <AttendanceModel>	Проверяется передача данных при переходе из окна отметки посещаемости обратно в форму добавления занятия.	Для каждого студента устанавливается тестовое значение статуса посещаемости.(заполняется массив данных посещаемости)	Массив передается корректно, из него можно извлечь все данные о посещаемости, которые должны соответствовать тестовым.
И - 4	Students.java и LessonAdd.java + public void ShowStudentChoice(View v)	Проверяется передача массива посещаемости при переходе из	С главного экрана происходит переход к добавлению занятия.	Все статусы массива посещаемости равны false.

		формы добавления занятия к отметке студентов(не редактирование).		
И - 5	Students.java и LessonAdd.java + public void ShowStudentChoice(View v)	Проверяется передача данных при переходе из формы добавления занятия к отметке студентов при редактировании занятия.	Создается тестовое занятие, устанавливаются тестовые статусы посещаемости для студентов. С главного экрана происходит переход к редактированию тестового занятия.	Статусы из полученного массива соответствуют установленным ранее тестовым статусам
И - 6	Subject.java и FilterSort.java	Эта функция вызывает множество других: getSubjectList(…), Функцию из класса FilterSort – Find(), которая в свою очередь вызывает функции сортировки и фильтрации sortAZ() и showOnlyFilter()	Создается экземпляр класса Subject.java Функции на вход дается номер тестовой группы из Firebase, для которой создаются несколько тестовых занятий, в числе которых есть предметы, начинающиеся на одну и ту же букву $letter, в произвольном порядке и строка поиска состоящая из буквы $letter.	Полученный список содержит те, и только те занятия, которые начинаются на букву $letter и расположены в алфавитном порядке
И - 7	Subject.java и FilterSort.java	Эта функция вызывает множество других: getSubjectList(…), Функцию из класса FilterSort – Find(), которая в свою очередь вызывает функции сортировки и фильтрации sortAZ() и showOnlyFilter()	Создается экземпляр класса Subject.java Функции на вход дается номер тестовой группы из Firebase, для которой создаются несколько тестовых занятий, в числе которых есть предмет с названием $subject, в произвольном порядке и строка поиска равная $subject.	Полученный список содержит только значение $subject
И - 8	Subject.java и FilterSort.java	Эта функция вызывает множество других: getSubjectList(…), Функцию из класса FilterSort – Find(), которая в свою очередь вызывает функции сортировки и фильтрации sortAZ() и showOnlyFilter()	Создается экземпляр класса Subject.java Функции на вход дается номер тестовой группы из Firebase, для которой создаются несколько тестовых занятий и пустая строка поиска.	Полученный список содержит все предметы группы, расположенные в алфавитном порядке
И - 9	Subject.java и FilterSort.java	Эта функция	Создается экземпляр	Полученный

		вызывает множество других: getSubjectList(…), Функцию из класса FilterSort – Find(), которая в свою очередь вызывает функции сортировки и фильтрации sortAZ() и showOnlyFilter()	класса Subject.java Функции на вход дается номер тестовой группы из Firebase, для которой создаются несколько тестовых занятий и строка поиска, не входящая в название ни одного из предметов	список пуст.

3.3. Перечень аттестационных тестов

Ниже описан сценарий для проверки требований.

	Описание теста (сценария)	Ожидаемый результат	Требование
А-1	Производится вход по данным тестового пользователя	Произошел переход на главный экран приложения. Отображаются все созданные на текущий день занятия Дата установлена текущая.	Авторизация пользователя по логину и паролю для доступа к дальнейшей работе с системой. Установка даты отображаемых занятий по умолчанию
А-2	Нажимается кнопка «Создать занятие»	Произошел переход на форму добавления занятия. Форма не заполнена	Установка даты и времени создаваемых занятий по умолчанию
	Нажимается кнопка выбора даты и времени	Появляется Диалоговое окно для выбора даты	
	Выбирается дата (и записывается в $testDate – для теста). Нажимается ОК	Появляется диалоговое окно для задания времени занятия	
А-3	Выбирается время (и записывается $testTime – для теста). Нажимается ОК	В поле даты/времени установлены выбранные дата и время	Установка даты проведения создаваемого занятия Установка времени проведения создаваемого занятия
А-4	Нажимается кнопка выбора преподавателя	Появляется окно со списком преподавателей. Не выбрано ни одно значение.	Формирование списка преподавателей путем запроса к Cloud Firestore
А-5	Выбирается преподаватель $testLecturer. Нажимается «сохранить»	Происходит возврат на форму добавления занятия. В поле «преподаватель» записан выбранный преподаватель.	Сохранения выбора преподавателя
А-6	Нажимается кнопка выбора предмета	Появляется окно со списком предметов группы. Не выбрано ни одно значение.	Формирование списка предметов путем запроса к Cloud Firestore с фильтром по группе

A-7	Выбирается дисциплина $testSubject. Нажимается «сохранить»	Происходит возврат на форму добавления занятия. В поле «предмет» записан выбранный предмет.	Сохранение выбора предмета
A-8	Нажимается кнопка «Отметить студентов»	Происходит переход к списку студентов для фиксирования посещаемости. В списке не отмечен ни один студент.	Формирование списка студентов путем запроса к Cloud Firestore с фильтром по группе
A-9	Отмечаются несколько тестовых студентов $СписокВыбранныхСтудентов Нажимается кнопка «Сохранить»	Происходит возврат в форму добавления занятия.	Сохранение списка посещаемости группы
A-10	Нажимается кнопка «Отмена»	Появляется диалоговое окно для подтверждения отмены.	Подтверждение всех потенциально опасных действий пользователя
	Нажимается кнопка «Отмена»	Переход обратно к форме создания занятия. Сохраняются все введенные ранее данные	
A-11	Нажимается кнопка «Сохранить»	Появляется диалоговое окно для подтверждения сохранения.	Подтверждение всех потенциально опасных действий пользователя
	Нажимается Ок	Происходит переход к главному экрану. Занятие сохраняется в Firebase	
	Нажимается кнопка выбора даты	Появляется Диалоговое окно для выбора даты	
A-12	Выбирается дата $testDate. Нажимается ОК.	В расписании есть созданное ранее занятие. Напротив него указано время $testTime.	Установка даты отображаемых занятий
A-13	Напротив занятия нажимается кнопка «Редактировать»	Происходит переход на форму добавления занятия. Значение дата и время установлены соответственно $testDate и $testTime. В поле «Предмет» установлено значение $testSubject. В поле «Преподаватель» установлено значение $testLecturer.	Сохранение параметров занятия в Cloud Firestore.
A-14	Нажимается кнопка «Отметить студентов»	Происходит переход к списку студентов для фиксирования посещаемости. В списке отмечены те и только те студенты, которые содежатся в списке $СписокВыбранныхСтудентов	Сохранение посещаемости занятия в Cloud Firestore.
	Меняются произвольные отметки посещаемости. Нажимается кнопка сохранения.	Происходит переход обратно к форме редактирования(добавления) занятия.	

	Меняется время и/или дата занятия	В поле даты и времени установлено новое значение $newTestDate и $newTestTime	
	Нажимается кнопка «Сохранить» подтверждается действие	Происходит переход к Главному экрану.	
A-15	В поле выбора даты выбирается дата $newTestDate	В списке занятий отображается созданное в начале занятие с измененными данными . Занятия со старыми данными нет. (оно перезаписано)	Внесение изменений в Cloud Firestore при редактировании занятия. Обновление информации в приложении при изменении данных в Firebase. Обновление расписания при изменении даты отображения занятий путем запроса к Cloud Firestore Установка даты для отображения занятий
A-16	Напротив занятия нажимается кнопка «Удалить»	Появляется диалоговое окно для подтверждения удаления	Подтверждение всех потенциально опасных действий пользователя
A-17	Нажимается Ок	Из расписания и из Firebase удаляются все данные о занятии, в т.ч. о посещаемости	Удаление созданных занятий путем запроса к Cloud Firestore.(в т.ч. удаление данных о посещаемости этого занятия)
A-18	В меню выбирается кнопка «Выход»	Появляется диалоговое окно для подтверждения выхода из системы	Подтверждение всех потенциально опасных действий пользователя
A-19	Нажимается Ок	Происходит переход на экран авторизации	Выход из системы.

4. Реализация тестов.

4.1. Модульные тесты.

Тестовый класс (юнит) : ExampleUnitTest.java. Тесты, не взаимодейтсвующие с элементами интерфейса

Переменные, используемые в тестах(функция выполняется перед запуском каждого теста):

```
@Before
public void CreateData(){
    sl = new ShowLessons();
    subject = new Subject();
    lm1 = new LessonModel( group_id: "", subject: "", lecturer: "", date: "", time: "");
    ma=new MainActivity();
    lm1.setid("54");
    wrong_email1 = "kkuzneco@cs";
    wrong_email2 = "tester@test";
    wrong_email3 = "test";
    email = "tester@yandex.ru";
    mail2 = "kkuzneco@cs.karelia.ru";
}
```

Тест	Реализация

Б -3	`@Test`
	`public void check_Incorrect_email1(){`
	`    assertEquals(false, ma.emailValid(wrong_email1));`
	`}`

Б -4	`@Test`
	`public void check_Incorrect_email3(){`
	`    assertEquals(false, ma.emailValid(wrong_email3));`
	`}`

Б - 2	`@Test`
	`public void check_correct_email(){`
	`    assertEquals(true, ma.emailValid(email));`
	`}`

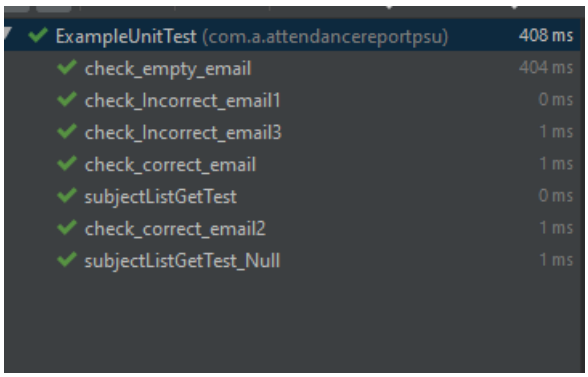
Б - 1	`@Test`
	`public void check_correct_email2(){`
	`    assertEquals(true, ma.emailValid(mail2));`
	`}`

Б - 17	`@Test`
	`public void check_empty_email(){`
	`    assertEquals(false, ma.emailValid(""));`
	`}`

Б - 15	`@Test`
	`public void subjectListGetTest(){`
	`    ArrayList<String> subject_list = new ArrayList<>();`
	`    subject_list = subject.getSubjectList("22000");`
	`    assertNotNull(subject_list);`
	`    assertTrue(subject_list.contains("testSubject"));`
	`}`

Б-16	`@Test`
	`public void subjectListGetTest_Null(){`
	`    ArrayList<String> subject_list = new ArrayList<>();`
	`    subject_list = subject.getSubjectList("00000");`
	`    assertNull(subject_list);`
	`}`

Результат выполнения класса:



Блочные тесты в ExampleInstrumentedTest.java

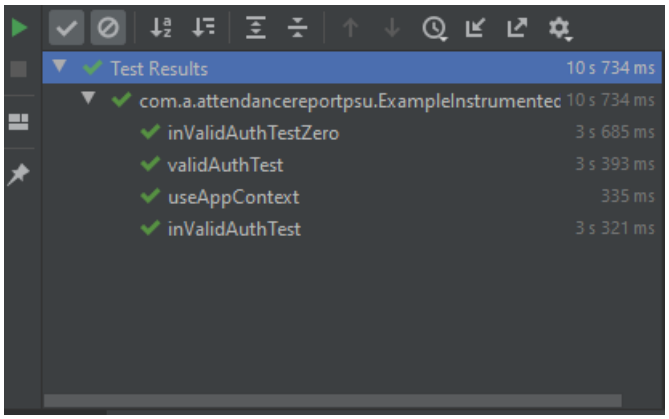
Тест	Реализация

Б - 5	`@Test`
	`public void validAuthTest() throws InterruptedException {`
	`String s = "123";`
	`activityActivityTestRule.getActivity().signIn("tester@test.ru", "123456",true);`
	`Thread.sleep(3000);`
	`Log.d("TAG", activityActivityTestRule.getActivity().getUID());`
	
	`assertEquals(activityActivityTestRule.getActivity().getUID(),"Nd40A9yA9jdHZ6Zsd9uqWAEmEMr2")`
	`;`
	`}`

Б-6	`@Test`
	`public void inValidAuthTest() throws InterruptedException{`
	`activityActivityTestRule.getActivity().signIn("tester@test.ru", "123", true);`
	`Thread.sleep(3000);`
	`assertEquals(null,activityActivityTestRule.getActivity().getUID());`
	`}`

Б - 7	`@Test`
	`public void inValidAuthTestZero() throws InterruptedException{`
	`activityActivityTestRule.getActivity().signIn("", "", true);`
	`Thread.sleep(3000);`
	`assertEquals(null,activityActivityTestRule.getActivity().getUID());`
	`}`

Результаты выполнения тестов:



4.2. Интеграционные тесты:

Т.к. проверка передачи данных при помощи intent достаточно трудно реализуема, были реализованы тесты И-6, И-7, И-8, И-9.

Тест	Реализация

И - 6	`@Test`
	`public void subjectFind() {`
	`ArrayList<String> subject_list = new ArrayList<>();`

	subject_list = subject.findByFilter("А", "22407");
	assertEquals(2, subject_list.size());
	assertTrue(subject_list.get(0).toString() == "Алгебра");
	assertTrue(subject_list.get(1).toString() == "Анализ требований");
	}

И - 7	@Test
	public void subjectFind_2() {
	ArrayList<String> subject_list = new ArrayList<>();
	subject_list = subject.findByFilter("Анализ требований", "22407");
	assertEquals(1, subject_list.size());
	assertTrue(subject_list.get(0).toString() == "Анализ требований");
	}

И - 8	@Test
	public void subjectFind_3() {
	ArrayList<String> subject_list = new ArrayList<>();
	subject_list = subject.findByFilter("", "22407");
	assertEquals(5, subject_list.size());
	ArrayList<String> arrayExample = new ArrayList<>();
	arrayExample.add("Алгебра");
	arrayExample.add("Анализ требований");
	arrayExample.add("Обеспечение информационной безопасности");
	arrayExample.add("Тестирование ПО");
	arrayExample.add("Философия");
	assertTrue(subject_list.get(0).toString() == "Алгебра");
	assertTrue(subject_list.get(1).toString() == "Анализ требований");
	assertTrue(subject_list.get(2).toString() == "Обеспечение информационной безопасности");
	assertTrue(subject_list.get(3).toString() == "Тестирование ПО");
	assertTrue(subject_list.get(4).toString() == "Философия");
	}

И - 9	@Test
	public void subjectFind_4() {
	ArrayList<String> subject_list = new ArrayList<>();
	subject_list = subject.findByFilter("История", "22407");
	assertEquals(0, subject_list.size());
	assertTrue(subject_list.isEmpty());
	}

Результаты выполнения:

4.3. Сценарий аттестационного теста реализован частично.

Шаги первого реализованного сценария(MainActivityTest.java):

1) Вводим в поле email логин тестового пользователя, в password – пароль

2) Нажимаем «Войти»

3) На главном экране ищем меню «еще», нажимаем на нее

4) Выбираем «Выход»

5) Должно появиться окошко для подтверждения действия пользователя.

6) Нажимаем «Да»

Код теста:

```
public class MainActivityTest {
/*
* роверяет обмен между intent-ми MainActivity и ShowLesons заходит по тестовой паре
логин/пароль, нажимает на "Выход"
* */

    @Rule
    public ActivityTestRule<MainActivity> mActivityTestRule = new
ActivityTestRule<>(MainActivity.class);
    @After
    public void tearDown() throws Exception {
        mActivityTestRule.getActivity().signOut();
    }
    @Test
    public void mainActivityTest() throws InterruptedException {
        ViewInteraction appCompatEditText = onView(
                allOf(withId(R.id.fieldEmail),
                        childAtPosition(
                                allOf(withId(R.id.linearLayout),
                                        childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                                2)),
                                0),
                        isDisplayed()));
        appCompatEditText.perform(click());

        ViewInteraction appCompatEditText2 = onView(
                allOf(withId(R.id.fieldEmail),
                        childAtPosition(
                                allOf(withId(R.id.linearLayout),
                                        childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                                2)),
                                0),
                        isDisplayed()));

        appCompatEditText2.perform(replaceText("tester@test.ru"));

        ViewInteraction appCompatEditText5 = onView(
                allOf(withId(R.id.fieldEmail), withText("tester@test.ru"),
                        childAtPosition(
                                allOf(withId(R.id.linearLayout),
                                        childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                                2)),
                                0),
                        isDisplayed()));
        appCompatEditText5.perform(closeSoftKeyboard());

        ViewInteraction appCompatEditText6 = onView(
                allOf(withId(R.id.fieldPassword),
                        childAtPosition(
                                allOf(withId(R.id.linearLayout),
                                        childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                                2)),
                                1),
```

```
                        isDisplayed()));
        appCompatEditText6.perform(replaceText("123456"), closeSoftKeyboard());

        ViewInteraction appCompatButton = onView(
                allOf(withId(R.id.signInButton), withText("Войти"),
                        childAtPosition(
                                allOf(withId(R.id.linearLayout),
                                        childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                                2)),
                                2),
                        isDisplayed()));
        appCompatButton.perform(click());
        Thread.sleep(500);
        ViewInteraction overflowMenuButton = onView(
                allOf(withContentDescription("Еще"),
                        childAtPosition(
                                childAtPosition(
                                        withId(R.id.action_bar),
                                        1),
                                0),
                        isDisplayed()));
        overflowMenuButton.perform(click());

        ViewInteraction appCompatTextView = onView(
                allOf(withId(R.id.title), withText("Выход"),
                        childAtPosition(
                                childAtPosition(
                                        withId(R.id.content),
                                        0),
                                0),
                        isDisplayed()));
        appCompatTextView.perform(click());

        ViewInteraction appCompatButton2 = onView(
                allOf(withId(android.R.id.button1), withText("ДА"),
                        childAtPosition(
                                childAtPosition(

withClassName(is("android.widget.ScrollView")),
                                        0),
                                3)));
        appCompatButton2.perform(scrollTo(), click());
    }

    private static Matcher<View> childAtPosition(
            final Matcher<View> parentMatcher, final int position) {

        return new TypeSafeMatcher<View>() {
            @Override
            public void describeTo(Description description) {
                description.appendText("Child at position " + position + " in parent
");
                parentMatcher.describeTo(description);
            }

            @Override
            public boolean matchesSafely(View view) {
                ViewParent parent = view.getParent();
                return parent instanceof ViewGroup && parentMatcher.matches(parent)
                        && view.equals(((ViewGroup) parent).getChildAt(position));
            }
        };
```

```
    }
}
```

Второй сценарий: (MainActivityTest2.java)

1) Вводим тестовые логин/пароль

2) Нажимаем «Войти»(Переход на главнй экран)

3) На главном экране нажимаем «Добавить занятие» (Переход на форму добавления занятия)

4) На форме добавления занятия нажимаем «Отмена» (Появляется диалоговое окно)

5) Появляется окошко подтверждения действий пользователя.

6) Нажимаем «Да» (Переход обратно на главный экран)

7) На главном экране с расписанием нажимаем на кнопку выбора даты.(появляется диалоговое окошко с календарем)

8) В появившемся календаре нажимаем «ок» (возврат к расписанию)

9) В меню «Еще» нажимаем на добавить отчет (ничего не происходит, так как обработкик нажатия пока не реализован)

10) Нажимаем добавить занятие (Переход на форму добавления занятия)

11) Нажимаем на выбор даты и времени (Появляется диалоговое окно с календарем)

12) В календаре нажимаем ок (появляется диалоговое окно с часами)

13) В окне выбора времени нажимаем ок

14) Нажимаем на выбрать предмет (переход к окну со списком предметов)

*Если на каком-либо этапе что-то не выполнилось – тест провален

Код теста:

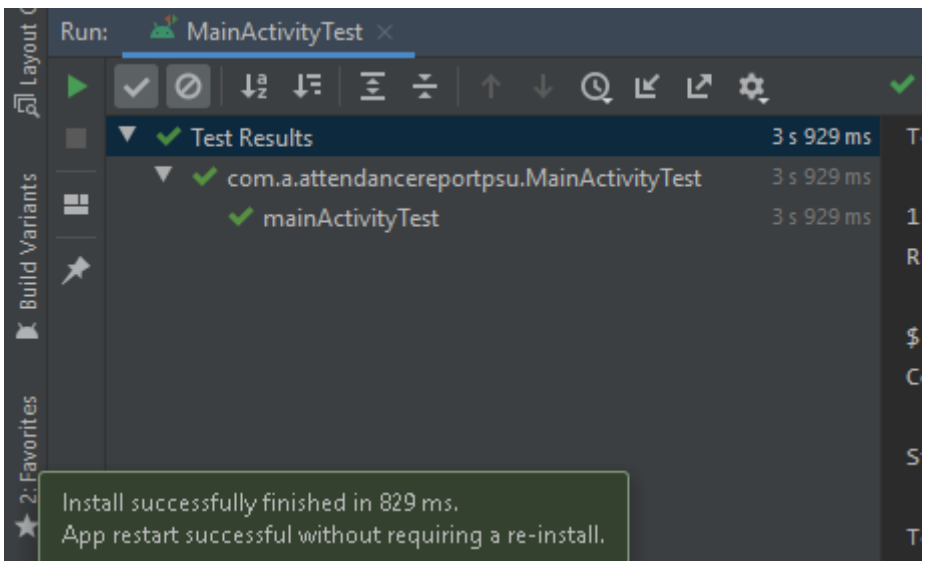
```
@Rule
public ActivityTestRule<MainActivity> mActivityTestRule = new
ActivityTestRule<>(MainActivity.class);
@After
public void tearDown() throws Exception {
    mActivityTestRule.getActivity().signOut();
}
@Test
public void mainActivityTest2() throws InterruptedException {
    Calendar dateAndTime = Calendar.getInstance();
    Date date = new Date();
    ViewInteraction appCompatEditText = onView(
            allOf(withId(R.id.fieldEmail),
                    childAtPosition(
                            allOf(withId(R.id.linearLayout),
                                    childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                            2)),
                            0),
```

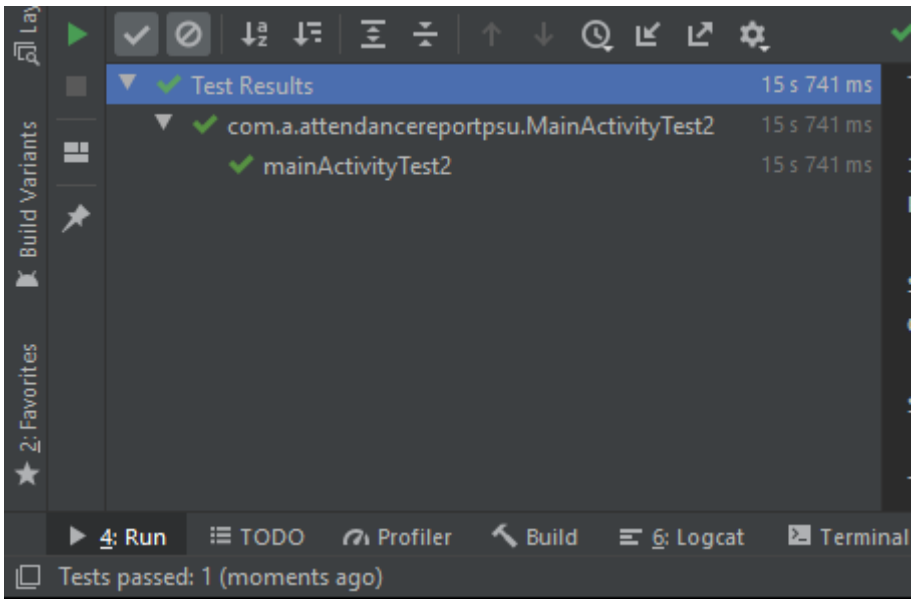
```
                    isDisplayed()));
    appCompatEditText.perform(replaceText("tester@test.ru"), closeSoftKeyboard());

    ViewInteraction appCompatEditText2 = onView(
            allOf(withId(R.id.fieldPassword),
                    childAtPosition(
                            allOf(withId(R.id.linearLayout),
                                    childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                            2)),
                            1),
                    isDisplayed()));
    appCompatEditText2.perform(replaceText("123456"), closeSoftKeyboard());

    ViewInteraction appCompatButton = onView(
            allOf(withId(R.id.signInButton), withText("войти"),
                    childAtPosition(
                            allOf(withId(R.id.linearLayout),
                                    childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                            2)),
                            2),
                    isDisplayed()));
    appCompatButton.perform(click(),closeSoftKeyboard());
    Thread.sleep(2200);
    ViewInteraction appCompatButton2 = onView(
            allOf(withId(R.id.button2), withText("добавить занятие"),
                    childAtPosition(
                            childAtPosition(

withClassName(is("androidx.constraintlayout.widget.ConstraintLayout")),
                                    0),
                            1),
                    isDisplayed()));
    appCompatButton2.perform(click());
    Thread.sleep(500);
    ViewInteraction appCompatButton3 = onView(
            allOf(withId(R.id.cancel), withText("отмена"),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.LinearLayout")),
                                    0),
                            5),
                    isDisplayed()));
    appCompatButton3.perform(click());
    Thread.sleep(500);
    ViewInteraction appCompatButton4 = onView(
            allOf(withId(android.R.id.button1), withText("ДА"),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.ScrollView")),
                                    0),
                            3)));
    appCompatButton4.perform(scrollTo(), click());
    Thread.sleep(500);
    ViewInteraction appCompatButton5 = onView(
            allOf(withId(R.id.date), withText("30 ноября 2020 г."),
                    childAtPosition(
                            childAtPosition(

withClassName(is("androidx.constraintlayout.widget.ConstraintLayout")),
                                    0),
```

```
                        0),
                isDisplayed()));
    appCompatButton5.perform(click());
    Thread.sleep(600);
    ViewInteraction appCompatButton6 = onView(
            allOf(withId(android.R.id.button1), withText("OK"),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.ScrollView")),
                                    0),
                            3)));
    appCompatButton6.perform(scrollTo(), click());
    Thread.sleep(600);
    ViewInteraction overflowMenuButton = onView(
            allOf(withContentDescription("Еще"),
                    childAtPosition(
                            childAtPosition(
                                    withId(R.id.action_bar),
                                    1),
                            0),
                    isDisplayed()));
    overflowMenuButton.perform(click());
    Thread.sleep(500);
    ViewInteraction appCompatTextView = onView(
            allOf(withId(R.id.title), withText("Создать отчет"),
                    childAtPosition(
                            childAtPosition(
                                    withId(R.id.content),
                                    0),
                            0),
                    isDisplayed()));
    appCompatTextView.perform(click());
    Thread.sleep(500);
    ViewInteraction appCompatButton7 = onView(
            allOf(withId(R.id.button2), withText("добавить занятие"),
                    childAtPosition(
                            childAtPosition(

withClassName(is("androidx.constraintlayout.widget.ConstraintLayout")),
                                    0),
                            1),
                    isDisplayed()));
    appCompatButton7.perform(click());

    Thread.sleep(500);
    ViewInteraction appCompatButton8 = onView(
            allOf(withId(R.id.dateEdit), withText( containsString("30 ноября 2020 г.,
")),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.LinearLayout")),
                                    0),
                            0),
                    isDisplayed()));
    appCompatButton8.perform(click());
    Thread.sleep(500);
    ViewInteraction appCompatButton9 = onView(
            allOf(withId(android.R.id.button1), withText("OK"),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.ScrollView")),
                                    0),
                            3)));
    appCompatButton9.perform(scrollTo(), click());
```

```
    Thread.sleep(500);
    ViewInteraction appCompatButton10 = onView(
            allOf(withId(android.R.id.button1), withText("OK"),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.ScrollView")),
                                    0),
                            3)));
    appCompatButton10.perform(scrollTo(), click());
    Thread.sleep(500);
    ViewInteraction appCompatButton11 = onView(
            allOf(withId(R.id.setSubject), withText("Выбрать предмет"),
                    childAtPosition(
                            childAtPosition(
                                    withClassName(is("android.widget.LinearLayout")),
                                    0),
                            1),
                    isDisplayed()));
    appCompatButton11.perform(click());
}

private static Matcher<View> childAtPosition(
        final Matcher<View> parentMatcher, final int position) {

    return new TypeSafeMatcher<View>() {
        @Override
        public void describeTo(Description description) {
            description.appendText("Child at position " + position + " in parent ");
            parentMatcher.describeTo(description);
        }

        @Override
        public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup && parentMatcher.matches(parent)
                    && view.equals(((ViewGroup) parent).getChildAt(position));
        }
    };
}
```

Результаты выполнения тестов:





Т.к. я не владею достаточными навыками по написанию тестов для тестирования intent-ов и передачи данных между ними, а передаваемые данные никак не отображаются на элементах интерфейса, то , например, для теста И-1 можно предложить следующее решение:

На форму добавить label или кнопку, текст которых будет меняться в зависимости от полученных данных. Т.е текст кнопки будет содержать номер переданной группы. Т.о. при запуске сценария текст кнопки(или label) будет содержать значение, которое нам нужно проверить. Если оно не соответствует тестовому (т.е тест не может найти кнопки с тестовой надписью) то тест проваливается.

Реализация теста:

```
@Test
public void integrationTest() throws InterruptedException {
    ViewInteraction appCompatEditText = onView(
            allOf(withId(R.id.fieldEmail),
                    childAtPosition(
                            allOf(withId(R.id.linearLayout),
                                    childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                            2)),
                            0),
                    isDisplayed()));
    appCompatEditText.perform(replaceText("tester@test.ru"), closeSoftKeyboard());

    ViewInteraction appCompatEditText2 = onView(
            allOf(withId(R.id.fieldPassword),
                    childAtPosition(
                            allOf(withId(R.id.linearLayout),
                                    childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                            2)),
                            1),
                    isDisplayed()));
    appCompatEditText2.perform(replaceText("123456"), closeSoftKeyboard());

    ViewInteraction appCompatButton = onView(
            allOf(withId(R.id.signInButton), withText("Войти"),
                    childAtPosition(
```

```
                        allOf(withId(R.id.linearLayout),
                                childAtPosition(

withClassName(is("androidx.coordinatorlayout.widget.CoordinatorLayout")),
                                        2)),
                        2),
                isDisplayed()));
    appCompatButton.perform(click());
    Thread.sleep(2200);
    ViewInteraction appCompatButton2 = onView(
            allOf(withId(R.id.button2), withText("добавить занятие"),
                childAtPosition(
                        childAtPosition(

withClassName(is("androidx.constraintlayout.widget.ConstraintLayout")),
                                0),
                        1),
                isDisplayed()));
    appCompatButton2.perform(click());
    Thread.sleep(2200);
    ViewInteraction appCompatButton3 = onView(
            allOf(withId(R.id.buttontest), withText("22000"),
                childAtPosition(
                        childAtPosition(
                                withClassName(is("android.widget.LinearLayout")),
                                0),
                        6),
                isDisplayed()));

}
```

Аналогично можно проверить и другие связи, в т.ч. передачу списка посещаемости. Например, сделав диалоговое окно, которое буде выводить список отмеченных студентов.

5. Журнал тестирования.

№	Дата	Тестировщик	Объект	Кол-во тестов	Из них ошибочных
Б - 3	14.12.2020	Кузнецова К.К.	public boolean emailValid(String email)	1	0
Б - 4	14.12.2020	Кузнецова К.К.	public boolean emailValid(String email)	1	0
Б - 2	14.12.2020	Кузнецова К.К.	public boolean emailValid(String email)	3	1 Отчет об ошибке № 1
Б - 1	14.12.2020	Кузнецова К.К.	public boolean emailValid(String email)	1	0
Б - 17	14.12.2020	Кузнецова К.К.	public boolean emailValid(String email)	1	0
Б - 15	14.12.2020	Кузнецова К.К.	getSubjectList(String groupNumber)	4	3 Отчет об ошибке № 2

Б - 16	14.12.2020	Кузнецова К.К.	Subject.java getSubjectList(String groupNumber)	1	0
Б - 5	14.12.2020	Кузнецова К.К.	void signIn(String email, String password)	1	0
Б - 6	14.12.2020	Кузнецова К.К.	void signIn(String email, String password)	1	0
Б - 7	14.12.2020	Кузнецова К.К.	void signIn(String email, String password)	1	0
И - 6	14.12.2020	Кузнецова К.К.	Subject.java и FilterSort.java	1	0
И - 7	14.12.2020	Кузнецова К.К.	Subject.java и FilterSort.java	1	0
И - 8	14.12.2020	Кузнецова К.К.	Subject.java и FilterSort.java	2	2 Отчет об ошибке № 3
И - 9	14.12.2020	Кузнецова К.К.	Subject.java и FilterSort.java	1	0
А - 1	14.12.2020	Кузнецова К.К.	Экран авторизации	1	0
А - 2	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А - 3	14.12.2020	Кузнецова К.К.	Форма добавления занятия	1	0
А - 4	14.12.2020	Кузнецова К.К.	Окно выбора преподавателя	1	0
А - 5	14.12.2020	Кузнецова К.К.	Окно выбора преподавателя	1	0
А - 6	14.12.2020	Кузнецова К.К.	Форма добавления занятия	1	0
А – 7	14.12.2020	Кузнецова К.К.	Окно выбора предмета	1	0
А – 8	14.12.2020	Кузнецова К.К.	Форма добавления занятия	1	0
А – 9	14.12.2020	Кузнецова К.К.	Фиксирование посещаемости студентов	1	0
А – 10	14.12.2020	Кузнецова К.К.	Фиксирование посещаемости студентов	1	0
А – 11	14.12.2020	Кузнецова К.К.	Форма добавления занятия	1	0
А – 12	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А – 13	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А - 14	14.12.2020	Кузнецова К.К.	Форма добавления(редактирования) занятия	1	0
А – 15	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А – 16	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А – 17	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А - 18	14.12.2020	Кузнецова К.К.	Главный экран	1	0
А - 19	14.12.2020	Кузнецова К.К.	Главный экран	1	0

6. Журнал найденных ошибок

Отчет об ошибке № 1

Дата составления отчета: 14.12.2020

Номер теста: Б - 2

Ожидаемый результат: true

Фактический результат: false

Статус ошибки: Исправлена

Комментарий: ошибка в коде

Отчет об ошибке № 2

Дата составления отчета: 14.12.2020

Номер теста: Б - 15

Ожидаемый результат: массив значений, содержащий значение «testSubject»

Фактический результат: null

Статус ошибки: Исправлена

Комментарий: ошибка в коде

Отчет об ошибке № 3

Дата составления отчета: 14.12.2020

Номер теста: И - 8

Ожидаемый результат: размер полученного массива равен 5

Фактический результат: размер полученного массива равен 4

Статус ошибки: Новая

Комментарий: Возможно, ошибка в тесте

7. Покрытие требований тестами. Матрица трассируемости.

Ф.т.	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
1	■																		
2																			■
3	Функционал не реализован																		
4			■																
5												■							
6	■																		
7		■																	
8			■																
9				■															
10						■													
11								■											
12							■												
13					■														
14									■										
15																			
16													■						
17														■	■				
18															■				
19															■				
20																	■		
21										■	■					■		■	

8. Вывод:

В результате тестирования было выявлено 3 ошибки, две из которых- ошибки кода. Они были исправлены. В результате заполнения матрицы трассируемости было обнаружено 3 дублирования функциональных требований. Матрица трассируемости показывает, что предложенные аттестационные тесты полностью покрывают выделенные функциональные требования. Т.к. тестируется только небольшая реализованная часть, то можно говорить только о качестве именно этой части. Но функционал приложения будет расширяться, и наличие ошибок на этом этапе говорит о том, что в дальнейшем приложение будет необходимо тестировать.