Минобрнауки России

Федеральное государственное автономное образовательное учреждение высшего образования   
«Национальный исследовательский университет   
«Московский институт электронной техники»»

Институт системной и программной инженерии и информационных технологий

Бакалаврская работа   
по направлению 09.03.04 «Программная инженерия»

«Разработка мобильного приложения для просмотра расписания МИЭТ»

(шифр МП ПР)

Студент Артамонова А. Ю.

Научный руководитель,   
Доц. Института СПИНТех, к.т.н. Фёдоров А. Р.

Москва 2023

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 3](#_Toc136681805)

[ВВЕДЕНИЕ 4](#_Toc136681806)

[1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 6](#_Toc136681807)

[1.1. Анализ существующих решений 6](#_Toc136681808)

[1.2. Концептуальное описание предметной области МП ПР 10](#_Toc136681809)

[1.3. Схема данных приложения 14](#_Toc136681810)

[1.4. Алгоритм работы приложения 14](#_Toc136681811)

[Выводы по исследовательскому разделу 17](#_Toc136681812)

[2. КОНТРУКТОРСКИЙ РАЗДЕЛ 18](#_Toc136681813)

[2.1. Выбор языка программирования 18](#_Toc136681814)

[2.2. Выбор среды разработки 22](#_Toc136681815)

[2.3. Особенности разработки Android приложений 28](#_Toc136681816)

[2.4. Разработка мобильного приложения МП ПР 36](#_Toc136681817)

[Выводы по конструкторскому разделу 41](#_Toc136681818)

[3. ИСПЫТАТЕЛЬНЫЙ РАЗДЕЛ 42](#_Toc136681819)

[3.1. Отладка 42](#_Toc136681820)

[3.2. Тестирование 44](#_Toc136681821)

[3.3. Выбор инструмента тестирования 49](#_Toc136681822)

[3.4. Проведение тестирования 61](#_Toc136681823)

[Выводы по испытательному разделу 70](#_Toc136681824)

[ЗАКЛЮЧЕНИЕ 71](#_Toc136681825)

[Список литературы 72](#_Toc136681826)

ПРИЛОЖЕНИЕ 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ.

ПРИЛОЖЕНИЕ 2. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.

ПРИЛОЖЕНИЕ 3. ТЕКСТ ПРОГРАММЫ.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

МП ПР – мобильное приложение для просмотра расписания

ПО – программное обеспечение

ОС – операционная система

БД – база данных

UML – язык графического описания для объектного моделирования

ИЛМ – инфологическая модель

ДЛМ – даталогическая модель

IDE – интегрированная среда разработки

SDK – Software Development Kit (комплект для разработки программного обеспечения)

NDK – Native Development Kit (комплект для нативной разработки)

dpi – количество точек на дюйм

ВВЕДЕНИЕ

Студенты просматривают свое расписание по несколько десятков раз за день, чтобы узнать сколько у них сегодня пар, в каком кабинете, что предстоит сделать на неделю. Заходить в браузер, искать расписание на сайте вуза, вводить свою группу является слишком длинным и долгим путём для просмотра своего расписания. А если интернет долго грузит страницу или отключится совсем? Даже если фотография с расписанием сохранена на телефоне, искать ее среди тысячи других проблематично.

Также у студентов возникает необходимость обговорить какой-то интересующий их вопрос с преподавателем, расписание которого можно узнать, только подойдя к кафедре.

Кроме того, расписание может быть составлено не очень удобно как для преподавателя, так и для студентов, из-за чего появляется необходимость переноса пары на другое время.

С большим количеством предметов трудно управлять своим свободным временем, поэтому возможность добавления задач с изменяемым приоритетом, могли бы помочь студентам структурировать свою занятость.

Исходя из сформулированных требований, ни одно из ныне существующих средств нельзя назвать удовлетворительным решением обозначенной проблемы – быстрый просмотр расписания занятий МИЭТ. В результате, актуальной является задача создания нового ПО, позволяющего упростить взаимодействие с расписанием МИЭТ в рамках ВКР бакалавра по направлению 09.03.04 «Программная инженерия» [1] (в соответствии с ГОСТ 19.701-90 [2], 19.505-78 [3] и 19.201-79 [4]).

Тема данной работы: разработка мобильного приложения для просмотра расписания МИЭТ.

Цель данной работы: ускорение просмотра расписания студенческих групп и преподавателей с возможностью поиска окна для переноса занятия.

Программный продукт должен обеспечивать следующие возможности:

* авторизация пользователя;
* просмотр своего расписания даже при отсутствии интернета;
* добавление задач, у которых автоматически меняется приоритет;
* просмотр расписания нужного преподавателя;
* просмотр расписания другой группы;
* поиск окна для групп и преподавателя, при необходимости с аудиторией.

Пояснительная записка состоит из введения, трёх разделов, заключения, списка литературы и трех приложений: технического задания, руководства пользователя и текста программы, и состоит из 77 страниц.

Исследовательский раздел включает в себя обзор существующих решений в области сервисов персональной аналитики, выбор архитектуры разрабатываемого ПО, схему данных, алгоритм работы.

Конструкторский раздел содержит информацию о выборе языка программирования, средств и технологий разработки ПО.

Технологический раздел состоит из отладки и тестирования программного модуля.

1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

* 1. Анализ существующих решений

В настоящее время существует большое количество программных средств для просмотра расписания ВУЗов. Большинство из программных средств реализовано на мобильных платформах. Это позволяет минимизировать время доступа к расписанию, упростить процесс взаимодействия с ним.

Рассмотрены следующие программные решения, реализованные под мобильные устройства (телефон или планшет) с сенсорным экраном, а также сайт МИЭТ:

* расписание на сайте miet.ru;
* «Orioks» от разработчика MIET;
* «Расписание занятий – Sked» от разработчика SKED LLC;
* «Кампус» от разработчика Ryabushko inc.;
* «СтудЖурнал» от разработчика Roman Sytnyk.

Эти приложения являются популярными среди пользователей устройств под управлением OC Android. Данные программы предоставлены в открытом доступе и для их скачивания необходимо стандартное приложение «Play Маркет». Однако, ни в одном из предложенных программных решений нельзя просмотреть расписание МИЭТ. А для просмотра расписания на сайте необходимо больше времени, чем хотелось бы, а также необходим доступ в интернет.

Расписание на сайте miet.ru

На сайте можно просмотреть расписание МИЭТ своей и других групп на сегодня и на неделю. Данные удобно расположены в виде таблицы. Для доступа к расписанию требуется доступ в интернет. По данным опроса студентов, данный способ просмотра расписания не очень удобен.

Приложение «Orioks»

Данное приложение позволяет просматривать свое расписание на сегодня, завтра, выбранный день, а также можно просмотреть расписание на неделю, где помечается числитель или знаменатель. Расписание загружается с ОРИОКСа. Также можно просмотреть расписание других групп, закрепив их в меню поиска для дальнейшего быстрого доступа. Кроме того, в приложении можно увидеть текущие дисциплины и полученные баллы за них и за контрольные мероприятия. На данный момент приложение не работает: не выгружает расписание и текущие дисциплины.

Работоспособность приложения гарантируется на устройствах с операционной системой Android версии 5.0 и более поздней версией. Для работы программы требуется Интернет-соединение.

Язык интерфейса: русский.

ПО предоставляется в магазине приложений «Play Маркет» и его модификация технически невозможна.

Программа предоставляется бесплатно.

Приложение «Расписание занятий – Sked»

Данное приложение позволяет просматривать свое расписание на каждый день, но нельзя посмотреть расписание на неделю. Расписание загружается с ОРИОКСа/сайта miet.ru. Кроме того, можно просмотреть расписание других групп и преподавателя. Также в поиске указываются все существующие преподаватели и группы. Приложение обеспечивает возможность добавление задач, которые видны всей группе. На данный момент приложение не работает.

Работоспособность приложения гарантируется на устройствах с операционной системой Android версии 5.0 и более поздней версией. Для работы программы требуется Интернет-соединение.

Язык интерфейса: русский.

ПО предоставляется в магазине приложений «Play Маркет» и его модификация технически невозможна.

Программа предоставляется бесплатно.

Приложение «Кампус»

Данное приложение загружает расписание с сайта ВУЗов. Просмотреть можно расписание групп и преподавателей. К плюсам относится то, что можно добавить виджет расписания на рабочий стол. Также есть возможность получать объявления, которые, скорее всего, публикует ВУЗ, но на данный момент там расположена реклама. Кроме того, есть вкладка «Отзывы», где можно прочитать и оставить отзыв о преподавателе, а также просмотреть рейтинг. Отзыв формируется из оценки компетентности, справедливости оценивания, умения донести материал, актуальности материала и общей оценки, а также тегов, характеризующих преподавателя, которые помогают составить портрет о нем и не влияют на рейтинг, и самого комментария. Основным минусом является то, что МИЭТа в списке ВУЗов нет.

Работоспособность приложения гарантируется на устройствах с операционной системой Android версии 5.0 и более поздней версией. Для работы программы требуется Интернет-соединение.

Язык интерфейса: русский.

ПО предоставляется в магазине приложений «Play Маркет» и его модификация технически невозможна.

Программа предоставляется бесплатно.

Приложение «СтудЖурнал»

В данном приложении расписание составляется вручную. Есть возможность разделить недели на числитель/знаменатель(четную/нечетную). Можно скопировать составленное расписание со знаменателя на числитель и наоборот. Карточка предмета состоит из названия, корпуса/аудитории, преподавателя, чек-бокса лекция/практика, что немного не удобно, так как МИЭТе есть еще и лабораторные работы, ссылки на урок и стрелок вверх/вниз, чтобы передвинуть предмет на пару вверх/вниз соответственно. Кроме того, есть вкладка заметки, где можно оставить небольшую запись, и домашнее задание, в которое добавляется предмет, название которого подгружается из расписания, дата и текст задания. Также можно настроить время пар и перемен между ними. Составленным расписание можно поделиться с одногруппниками.

Работоспособность приложения гарантируется на устройствах с операционной системой Android версии 5.0 и более поздней версией. Для работы программы требуется Интернет-соединение.

Язык интерфейса: русский.

ПО предоставляется в магазине приложений «Play Маркет» и его модификация технически невозможна.

Программа предоставляется бесплатно.

Сравнение существующих решений

В результате анализа существующих решений составлена таблица 1.1, в которой перечислены ключевые параметры для сравнения продуктов между собой.

В результате анализа вышеперечисленных программ, составлен перечень функций, которыми обязано обладать искомое решение:

* работоспособность приложения/сайта c расписанием МИЭТ;
* загрузка расписания;
* быстрый доступ к просмотру своего расписания/удобство работы;
* просмотр своего расписания без доступа в интернет;
* просмотр расписания преподавателя;
* поиск окна;
* наличие интерфейса в цветах «ОРИОКС».

Таблица 1.1 – Сравнение существующих решений

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Программный продукт  Параметры | Расписание на сайте miet.ru [5] | «Orioks» [6] | «Sked» [7] | «Кампус» [8] | «Студ  Журнал» [9] | МП ПР |
| Версия ОС | Любая | Android 5.0 и более поздние версии | Android 5.0 и более поздние версии | Android 5.0 и более поздние версии | Android 5.0 и более поздние версии | Android 5.0 и более поздние версии |
| Возможность работы с расписанием МИЭТ | Да | Нет | Нет | Нет | Да | Да |
| Загрузка расписания | Да | Да | Да | Нет | Нет | Да |
| Удобство работы | Нет | Да | Да | Да | Нет | Да |
| Просмотр своего расписания без доступа в интернет | Нет | Нет | Нет | Да | Да | Да |
| Просмотр расписания преподавателя | Нет | Нет | Да | Да | Нет | Да |
| Поиск окна | Нет | Нет | Нет | Нет | Нет | Да |
| Цвет приложения в соответствии с корпоративными цветами компании | Да | Да | Нет | Нет | Нет | Да |

Так как ни одно из существующих решений полностью не удовлетворяет описанным выше требованиям, то актуальной является задача создания нового программного обеспечения – МП ПР.

## Концептуальное описание предметной области МП ПР

Архитектура МП ПР описывается при помощи Use case диаграммы, ИЛМ и ДЛМ, которые направлены на облегчение разбора связей элементов сложного объекта. Данные диаграммы показывают, как система работает, скрывая детальную реализацию. UML – это язык графического описания для объектного моделирования в области разработки ПО, моделирования бизнес-процессов, системного проектирования и отображения организационных структур [10].

Use case диаграмма МП ПР указана на рисунке 1.1.

На схеме представлены различные сценарии, которые соответствуют требованиям, предъявляемым к программному обеспечению. Один из основных сценариев – авторизация пользователя. После успешной авторизации пользователь получает доступ к функционалу приложения.

При первом входе пользователя в приложение, его расписание сохраняется в локальном хранилище, чтобы обеспечить быстрый доступ к нему в последующих сеансах. Пользователь может просмотреть свое расписание, которое извлекается из локальной базы данных и отображается на экране.

Пользователь также имеет возможность просматривать расписание других групп. Для этого он выбирает соответствующую группу, и расписание этой группы также получается из локального хранилища и отображается на экране. Также имеется возможность просмотра расписания преподавателя. Для этого сначала загружаются расписания групп из локального хранилища, а затем формируется расписание преподавателя на основе этих данных.

Для нахождения окна в расписании для переноса задачи, пользователю необходимо указать группы и преподавателя, для которых нужно найти свободное время. Система использует эти данные для анализа расписания и предоставляет информацию о доступных окнах.

Пользователь также может добавлять задачи в свое расписание. Для этого он вводит название, текст задачи, предмет и срок выполнения. Задача сохраняется в локальном хранилище и становится доступной для просмотра и управления.

Обновление расписания происходит путем загрузки данных из локального хранилища, если они были сохранены ранее, или из базы данных, если нет сохраненных данных. Это позволяет обновлять и отображать актуальное расписание пользователя.

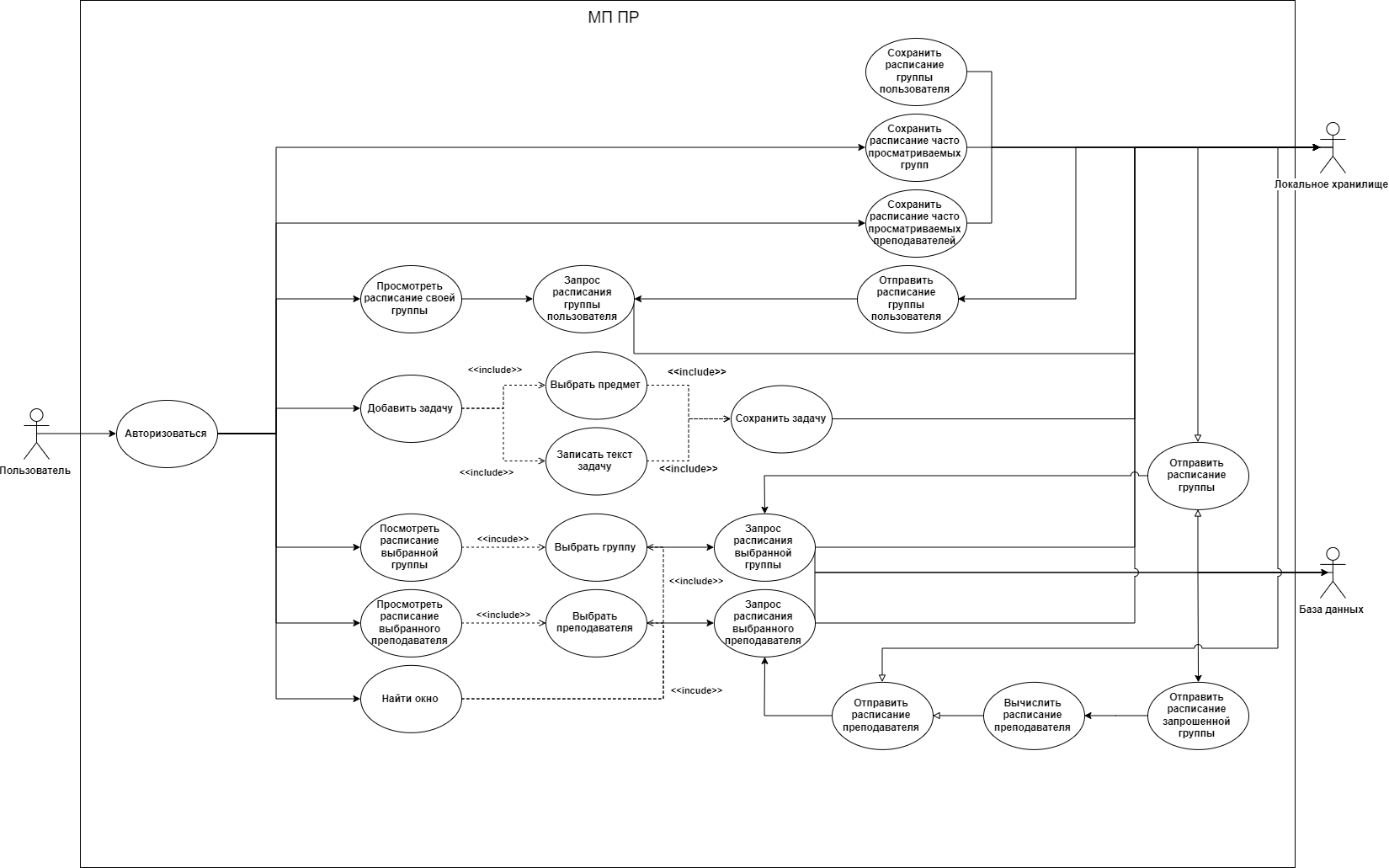


Рисунок 1.1 – Use case диаграмма

Инфологическая модель представлена на рисунке 1.2.

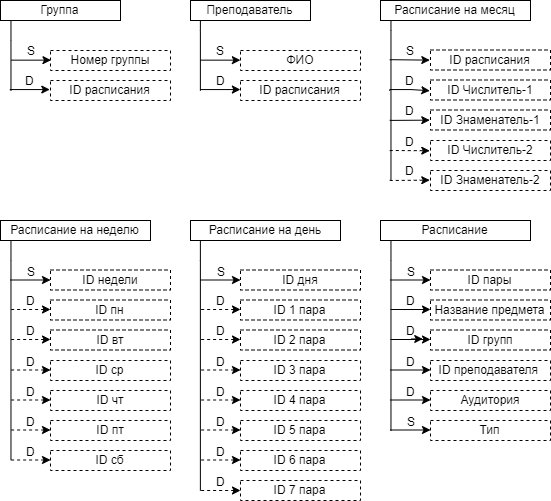


Рисунок 1.2 – Инфологическая модель предметной области

На инфологической модели предоставлены сущности с атрибутами:

* «Группа» со статическим атрибутом «Номер группы» и динамическим «ID расписания»;
* «Преподаватель» со статическим атрибутом «ФИО» и динамическим «ID расписания»;
* «Расписание на месяц» со статическим атрибутом «ID расписания» и динамическими «ID Числитель-1», «ID Знаменатель-1», «ID Числитель-2», «ID Знаменатель-2», причем ID последних двух недель могут отсутствовать, если расписание одинаковое для числителя 1(знаменателя 1) и числителя 2(знаменателя 2);
* «Расписание на неделю» со статическим атрибутом «ID недели» и динамическими для ID каждого дня недели, при этом ID дней может отсутствовать, так как в данный день отсутствуют занятия;
* «Расписание на день» со статическим атрибутом «ID дня» и динамическими для ID каждого пары, причем ID пар может отсутствовать;
* «Расписание» со статическими атрибутами «ID пары» и «Тип», который может иметь три значения: «Лекция», «Семинар», «Лабораторная», и динамическими «Название предмета», «ID групп», которых может быть несколько, «ID преподавателя» и «Аудитория».

Даталогическая модель представлена на рисунке 1.3.

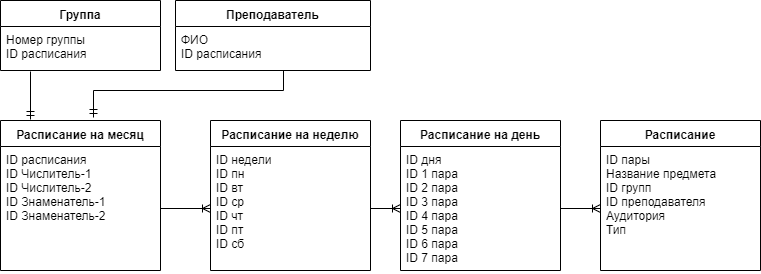


Рисунок 1.3 – Даталогическая модель базы данных

На данной даталогической модели представлены различные сущности и связи между ними, которые описывают систему расписания для групп и преподавателей. В модели присутствуют следующие сущности: "Группа", "Преподаватель", "Расписание на месяц", "Расписание на неделю", "Расписание на день" и "Расписание".

Сущности "Группа" и "Преподаватель" имеют связь один к одному с сущностью "Расписание на месяц". Это означает, что каждая группа и преподаватель связаны с одним расписанием на месяц.

Сущность "Расписание на месяц" имеет связь один ко многим с сущностью "Расписание на неделю". То есть каждое расписание на месяц содержит несколько расписаний на неделю.

Сущность "Расписание на неделю" в свою очередь имеет связь один ко многим с сущностью "Расписание на день". Это означает, что каждое расписание на неделю содержит несколько расписаний на день.

Наконец, сущность "Расписание на день" имеет связь один ко многим с сущностью "Расписание". То есть каждое расписание на день содержит несколько расписаний.

## Схема данных приложения

Схема данных представлена на рисунке 1.4. В течение работы приложения происходит постоянное взаимодействие пользователя с базой данных.



Рисунок 1.4 – Схема данных приложения

На схеме видно, что в процессе взаимодействия с пользователем система демонстрирует различные состояния. Одним из таких состояний является отображение пользователю окон с соответствующим расписанием. Это позволяет пользователю просматривать информацию о своем расписании и осуществлять необходимые операции. Кроме того, при нажатии кнопок "Поиск окна" и "Задачи" система также переходит в определенные состояния. При нажатии кнопки "Поиск окна" пользователю предоставляются возможные варианты окон для переноса расписания для выбранных групп и преподавателя. Это позволяет пользователю найти подходящие временные интервалы для переноса занятий. Кнопка "Задачи" ведет к отображению добавленных пользователем задач. В этом состоянии система демонстрирует список задач, которые были введены пользователем, и предоставляет возможность просмотра и управления ими.

* 1. Алгоритм работы приложения

Схема алгоритма представлена на рисунке 1.5. Для использования приложения необходимо пройти процесс авторизации с использованием системы ОРИОКС.

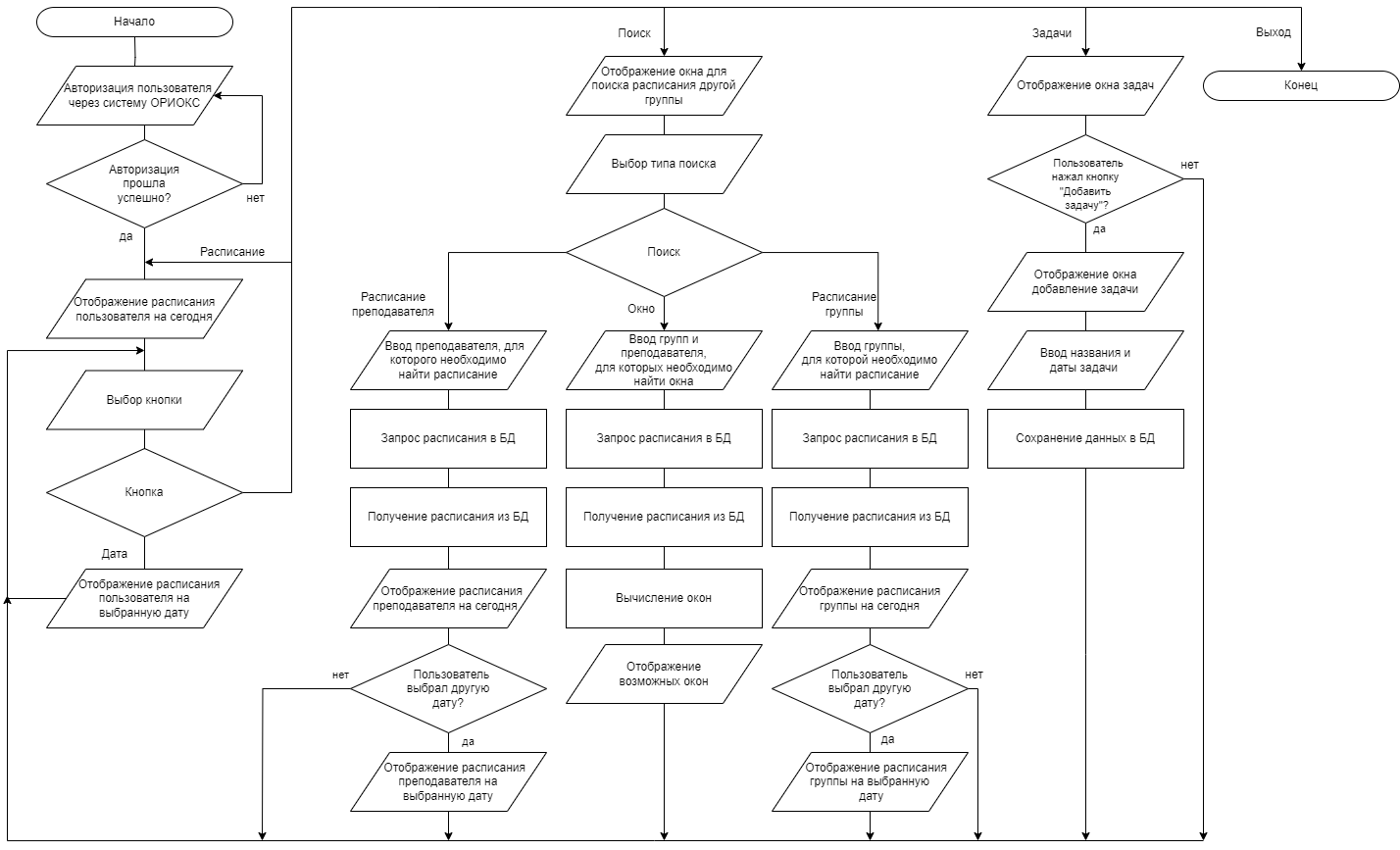


Рисунок 1.5 – Схема алгоритма работы приложения

После успешной авторизации пользователь получает доступ к функционалу приложения. Основные возможности включают просмотр собственного расписания на текущий день и другие дни. При нажатии кнопки "Поиск" пользователю предоставляется возможность выполнить поиск расписания для определенных групп или преподавателей, а также найти доступные окна для переноса занятий. Пользователь может ввести номер группы или фамилию преподавателя для запроса соответствующей информации из базы данных.

При нажатии кнопки "Задачи" пользователь может просмотреть список существующих задач и добавить новую задачу. Для добавления задачи необходимо указать название предмета, срок выполнения и саму задачу. Задачи сохраняются в системе для последующего отображения и управления.

Для возврата к просмотру собственного расписания пользователю достаточно нажать кнопку "Расписание".

Таким образом, приложение предоставляет пользователю удобные инструменты для работы с расписанием, поиском информации и управлением задачами.

# Выводы по исследовательскому разделу

В результате исследования предметной области и приложений-аналогов, принято решение о разработке собственного мобильного приложения, обладающего возможностью просмотра своего расписания, расписания других групп и преподавателей, а также поиск окна для переноса занятия и добавление задач. Сформулирован перечень требований к функционалу разрабатываемого приложения.

Мобильное приложения для просмотра расписания МИЭТ реализуется для мобильных устройств c операционной системой Android. Для облегчения этапа написания кода спроектированы Use case диаграмма, а также ИЛМ и ДЛМ. Разработаны схема данных, которая призвана облегчить понимание логических связей по данным между рабочими экранами приложения и схема алгоритма, на которой подробно описан процесс работы приложения.

1. КОНТРУКТОРСКИЙ РАЗДЕЛ

## Выбор языка программирования

Выбор языка программирования основывался на сравнении доступных средств и инструментов языка и требований к мобильному приложению.

Перечень требований, которые могут повлиять на выбор языка программирования:

* наличие транслятора под платформу Android;
* наличие документации;
* наличие бесплатных инструментов для разработки;
* наличие декларативного фреймворка;
* наличие возможности контроля потоком данных;
* читабельность кода;
* опыт работы с языком разработчика;
* удобный инструментарий отладки кода.

Так как приложение реализовано на мобильной платформе проводился анализ только тех языков, которые пригодны для разработки нативных приложений. В качестве платформы выбрана операционная система Android, так как его доля на мировом рынке составляет 75% [11].

Язык С++

Язык C++  является [компилируемым](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), статически типизированным языком программирования общего назначения. В нем существует множество возможностей, отсутствующих в C: поддержка объектно-ориентированного программирования (ООП) и шаблонов. У него есть возможности для использования низкоуровневых операций, таких как манипулирование битами и поддержка многопоточного программирования.

По аналогии с C, C++ широко используется в системном программирование и разработке приложений для операционных систем, создании компиляторов, драйверов устройств и игр. Также используется в разработке приложений с графическим интерфейсом GUI, научных и инженерных компьютерного программного обеспечения, разработке веб-серверов и серверов внутреннего назначения.

Это сложный язык программирования, который требует от разработчиков глубокого понимания языка и опыта работы с ним. Он также имеет репутацию более уязвимого для ошибок программирования, таких как переполнение буфера, по сравнению с более современными языками программирования.

Для разработки Android-приложения существует Android NDK. С помощью данного набора средств можно написать библиотеки на С++, которые впоследствии можно подключить к приложению с помощью Java, так как Android NDK не работает с графическим интерфейсом.

Язык С#

Язык С# является объектно-ориентированным языком общего назначения. Язык имеет статическую типизацию, поддерживает полиморфизм и перегрузку операторов. C# поддерживает абстрактные типы данных, делегаты, события и многопоточность. Это один из ключевых языков для разработки приложений на платформе Windows. Язык стал ответом на появление Java и других языков программирования, которые становились все более популярными в те годы.

Одной из основных характеристик C# является его близость к языку Java. Кроме того, C# имеет множество функций. Мощная поддержка типов: C# – это статически типизированный язык, который помогает избежать многих ошибок во время выполнения. Объектно-ориентированный дизайн: C# позволяет создавать классы и объекты, упрощая разработку сложных систем. Управление памятью: C# имеет сборщик мусора, который автоматически удаляет неиспользуемые объекты и освобождает память. Поддержка многопоточности: C# поддерживает создание многопоточных приложений, что позволяет создавать более быстрые и эффективные системы. Поддержка LINQ: язык C# включает поддержку Language Integrated Query, которая позволяет работать с данными из разных источников, используя общий синтаксис запросов.

C# стал основным языком программирования для разработки приложений на платформе .NET Framework. Он широко используется для создания веб-приложений, настольных приложений, игр, мобильных приложений и других систем. C# также позволяет создавать библиотеки классов, которые можно использовать в различных приложениях.

Для разработки Android-приложения существует Xamarin, фреймворк для кроссплатформенной разработки мобильных приложений с использованием языка C#. На данный момент Xamarin SDK, включая библиотеки и инструменты командной строки, стал открытым и доступным для всех. Но приложения, создаваемые с помощью фреймворка, обычно больше, чем нативные. И так как Xamarin сравнительно новый инструмент возникают сложности с интеграцией сторонних ресурсов.

Язык Python

Язык Python является высокоуровневым языком программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика за счет читаемости кода. Синтаксис языка минималистичен, за счет чего редко возникает необходимость обращаться к документации.

К особенностям Python относятся. Интерпретируемость: Python выполняется с помощью интерпретатора, позволяющего сразу проверить код на наличие возможных ошибок, а также запустить его без предварительной компиляции. Динамическая типизация: переменные в Python могут принимать различные типы данных в течение жизни программы. Объектно-ориентированная парадигма: Python поддерживает создание классов, создание объектов и наследование. Мультипарадигма: Python поддерживает как процедурное, так и функциональное программирование. Большая стандартная библиотека: Python имеет обширную библиотеку модулей для сетей, баз данных, графики, научных вычислений и машинного обучения. Поддержка нескольких ОС: Python работает на Windows, macOS, Linux и многих других платформах.

Python используется в различных областях, например, научное и инженерное вычисление, разработка сайтов и приложений, машинное и искусственное обучение, анализ и анализ данных, автоматизированные задачи и скрипты, разработка игр.

Для разработки Android приложений существует два фреймворка Kivy и BeeWare Project. Kivy – это библиотека Python, предназначенная для разработки кроссплатформенных GUI приложений. Его недостатком является отсутствие нативного набора инструментов пользовательского интерфейса. BeeWare Project предлагает набор инструментов, которые можно использовать для написания мобильных и десктопных приложений. Но данный фреймворк находится в разработке, поэтому часто происходит изменения API, ошибки и отсутствия необходимых функций.

Язык Java

Язык Java является строго типизированным объектно-ориентированным языком программирования общего назначения.

Приложения Java могут работать в любой операционной системе, в которой установлена ​​виртуальная машина Java (JVM). Код Java проверяется компилятором на наличие ошибок во время компиляции, что позволяет избежать многих ошибок во время выполнения. Все элементы языка (классы, объекты, методы и т. д.) являются объектами. Java автоматически освобождает память, когда объект больше не используется. В Java есть механизмы для эффективной работы с многопоточностью. Java уже поставляется со многими стандартными библиотеками, что упрощает и ускоряет разработку. Благодаря своей особой архитектуре Java обеспечивает высокую производительность во многих приложениях.

Java используется для создания многих типов программного обеспечения, включая мобильные приложения, веб-приложения, игры и настольные программы. Он также используется в крупных бизнес-приложениях и больших системах обработки данных.

Java в разработке считается фундаментальным, а также поддерживается многими популярными средами разработки, такие как Eclipse, Netbeans и JetBrains.

Язык Kotlin

Язык Kotlin является статически типизированным, объектно-ориентированным языком программирования, работающий поверх Java Virtual Machine. У языка Kotlin более быстрая компиляция и лучшая поддержка языка в IDE. Kotlin полностью совместим с Java, что позволяет для существующего Android-приложения внедрять новые функции на Kotlin без переписывания приложения целиком.

Kotlin – это язык программирования со статической типизацией, который позволяет обнаруживать ошибки компиляции. Kotlin поддерживает функциональное программирование и имеет такие конструкции, как лямбда-выражения и функции более высокого порядка. Kotlin можно использовать с Java, что упрощает перенос проектов с Java на Kotlin. Kotlin имеет чистый и простой синтаксис, который позволяет быстро писать и читать код. Kotlin поддерживает расширения для добавления функциональности к существующим классам. Kotlin может работать с Java и использовать библиотеки Java.

Сравнение языков программирования

Чтобы наглядно продемонстрировать возможности рассмотренных языков программирования составлена таблица 2.1.

Таблица 2.1 – Сравнение языков программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Язык  Критерий | С++ [12] [13] | С# [14] [15] | Python [16] [17] | Java [18] | Kotlin [19] |
| Наличие транслятора под платформу Android | Нет | Да | Нет | Да | Да |
| Наличие документации | Да | Да | Да | Да | Да |
| Наличие бесплатных инструментов для разработки | Да | Да | Да | Да | Да |
| Наличие декларативного фреймворка | Нет | Нет | Нет | Нет | Да |
| Наличие возможности контроля потоком данных | Нет | Нет | Нет | Нет | Да |
| Удобный инструментарий отладки кода | Нет | Нет | Да | Да | Да |
| Опыт работы | 1 год | 2 года | 1 год | 2 года | 2 года |

В результате сравнения для разработки МП ПР выбран язык программирования Kotlin, так как в нем присутствует транслятор под платформу Android и под Kotlin существует декларативный фреймворк Jetpack Compose, который дает возможность контроля потоком данных.

## Выбор среды разработки

После выбора языка необходимо выбрать среду разработки. К основным требованиям относится:

* наличие эмулятора для запуска Android-приложения;
* наличие бесплатных инструментов для разработки;
* сборка приложений, основанная на Gradle;
* инструменты для работы с UI;
* инструменты для тестирования.

JetBrains предоставляет официальный плагин Kotlin для двух интегрированных сред разработки (IDE): IntelliJ IDEA и Android Studio.

Другие IDE и редакторы исходного кода, такие как Eclipse, Visual Studio Code и Atom, имеют плагины, поддерживаемые сообществом Kotlin.

Среда разработки IntelliJ IDEA

IntelliJ IDEA – IDE для языков JVM, предназначенная для максимальной производительности разработчиков. Она обеспечивает интеллектуальное завершение кода, статистический анализ кода и рефакторинг. Среда предлагает различные инструменты автоматизации сборки, включая Maven, Gradle, Gant, NPM и Grunt, которые помогают членам команды выполнять такие действия, как компиляция кода, упаковка, тестирование и многое другое.

IntelliJ IDEA предоставляет простой для понимания и использования пользовательский интерфейс. Его интуитивно понятный интерфейс и интеллектуальные функции позволяют разработчикам работать более эффективно и быстро. IntelliJ IDEA имеет инструменты управления зависимостями проекта, позволяющие разработчикам управлять библиотеками, платформами и другими зависимостями проекта в удобном интерфейсе. IntelliJ IDEA предоставляет инструменты для рефакторинга кода, такие как переименование переменных, методов и классов, извлечение методов, упрощение выражений и т. д. Это улучшает качество и читаемость кода. IntelliJ IDEA поддерживает широкий спектр языков программирования, включая Java, Kotlin, Python, JavaScript, PHP и многие другие. IntelliJ IDEA интегрируется со многими системами контроля версий, включая Git, Mercurial, Subversion и другими, что упрощает работу с кодом, контроль изменений и управление версиями. IntelliJ IDEA имеет встроенные инструменты для автоматической проверки кода, такие как проверки, которые обнаруживают проблемы в вашем коде и предлагают способы их устранения. IntelliJ IDEA обладает широкими возможностями расширения и может быть интегрирован с другими инструментами и плагинами, что позволяет настраивать IDE в соответствии с конкретными потребностями разработчика.

IntelliJ IDEA – одна из самых популярных IDE для разработки приложений на Java и Kotlin, обладающая множеством функций и инструментов, которые делают ее таковой.

Среда разработки Eclipse

Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. В его основе лежат фреймворк OSGi и SWT/JFace. Eclipse служит платформой для разработки расширений. Существуют расширения для добавления в IDE возможность работы с базой данных, серверами приложение и другое. Главные преимущества Eclipse бесплатность и открытый код. Изначально Eclipse разрабатывалась для программирования на языке Java, но уже давно доступна для разработки на других языках программирования.

Eclipse поддерживает множество языков программирования, включая Java, C/C++, PHP, Python, Ruby, Perl, JavaScript и другие. Каждый язык имеет соответствующие инструменты и функции, такие как подсветка синтаксиса, автозаполнение, отладчик и т. д. Eclipse можно использовать в различных операционных системах, включая Windows, Linux и macOS. Также его можно настроить для работы с различными серверами приложений, такими как Apache Tomcat, JBoss, WebSphere и другими. Eclipse предлагает широкий спектр инструментов разработки, включая визуальные редакторы, инструменты анализа кода, поддержку автоматизированного тестирования, инструменты баз данных, инструменты разработки мобильных приложений и многое другое. Eclipse имеет модульную архитектуру, которая позволяет разработчикам создавать свои собственные плагины и инструменты, расширяя функциональность среды разработки по своему усмотрению.

К недостаткам данной среды разработки относятся возможность подключать плагины, так как не все плагины совместимы, и не интуитивность в интерфейсе [20].

Плагин Kotlin для Eclipse основан на плагине JetBrains Kotlin и содержит небольшие улучшения. Эти улучшения в основном связаны с адаптацией для последних версий Java [21].

Среда разработки Visual Studio Code

Visual Studio Code – среда разработки для разных языков программирования. Он немного весит, гибкий и прост в освоении. VS Code позволяет писать, редактировать и форматировать код разных языков программирования. Также в нем есть встроенный интеллектуальный помощник и инструменты для отладки. Редактор легко расширяется с помощью плагинов. Недостатки IDE ограниченная функциональность и медленная работа.

VS Code предоставляет мощный и настраиваемый редактор кода, который поддерживает множество языков программирования и позволяет работать с различными форматами файлов. Он изначально поддерживает систему контроля версий Git, что делает процесс контроля версий более удобным и позволяет быстро отслеживать изменения в ваших проектах. VS Code предоставляет инструменты для отладки кода, включая поддержку точек останова, отображение значений переменных, пошаговое выполнение кода и многое другое. Он позволяет работать с командной строкой прямо в среде разработки, что значительно ускоряет процесс разработки и упрощает работу с проектами. VS Code поддерживает расширения, которые позволяют добавлять новые функции и инструменты для расширения возможностей разработки и повышения производительности. VS Code предоставляет инструменты для быстрого создания новых проектов, включая шаблоны и генераторы кода. VS Code позволяет подключаться к различным облачным службам, таким как Azure и AWS, что упрощает разработку приложений для облачных служб.

Visual Studio Code – это мощная и гибкая среда разработки, позволяющая быстро и удобно создавать разнообразные приложения и проекты.

Плагин предоставляет интеллектуальную сборку кода, линтинг, отладку, форматирование и многое другое для Kotlin с использованием языкового сервера Kotlin и адаптера отладки Kotlin [22].

Среда разработки Atom

Atom – это устаревший бесплатный редактор текста и исходного кода с открытым исходным кодом. Редактор можно настроить с помощью HTML, CSS и JavaScript. Недостатками являются не очень удачные решения в дизайне и некоторые ошибки.

Atom имеет встроенную поддержку Git, что делает его идеальным выбором для разработки программного обеспечения с использованием этой системы контроля версий. В Atom есть удобный менеджер пакетов, который позволяет быстро и легко устанавливать и управлять плагинами и темами. Atom имеет функцию завершения кода, которая может значительно ускорить процесс кодирования и уменьшить количество ошибок. Atom поддерживает множество языков программирования, что делает его универсальным инструментом для разработки многих приложений. Несмотря на все преимущества, Atom также имеет некоторые недостатки, такие как немного медленная работа с большими файлами или проектами, а также меньшее количество пользователей, чем в других популярных средах разработки.

Среда разработки Vim

Vim – свободный текстовый редактор. Он значительно легче, чем другие IDE, поддерживает все языки программирования и любые необходимые для программирования инструменты.

Vim работает в терминале, что делает его удобным для удаленного использования и на серверах, которые могут не иметь графического интерфейса. Он также имеет множество плагинов и настроек, которые можно настроить для облегчения написания кода. К основным особенностям Vim можно отнести мощную систему настроек и плагинов, наличие интегрированной системы управления файлами и поиска в каталогах, быстрый и эффективный интерфейс командной строки, позволяющий легко выполнять команды без использования мыши, множество форматов и подсветку синтаксиса. варианты различных языков программирования, поддержка большинства операционных систем, включая Unix, Linux, Windows и Mac OS X.

Однако Vim – довольно сложный редактор, для изучения и освоения его функций требуется время. Он также имеет довольно высокий порог входа для новичков в программировании, которым могут быть интересны более простые и интуитивно понятные редакторы.

Особенности плагина Kotlin для Vim подсветка синтаксиса, базовые отступы, поддержка синтаксиса.

Среда разработки Android Studio

Android Studio – официальная среда разработки приложений для Android. Помимо мощного редактора кода и инструментов разработчика, IDE предлагает еще больше функций, повышающих продуктивность разработчика, таких как:

* гибкая система сборки на основе Gradle;
* быстрый и многофункциональный эмулятор;
* применение изменений для push-кода и изменений ресурсов в работающем приложении без перезапуска приложения;
* шаблоны кода и интеграция с GitHub;
* обширные инструменты и фреймворки для тестирования;
* инструменты Lint для выявления проблем с производительностью.

В Android Studio есть удобный редактор макетов, который позволяет создавать пользовательские интерфейсы с помощью графического интерфейса. Android Studio включает эмулятор Android, который можно использовать для тестирования и отладки приложений на различных устройствах. Android Studio включает инструменты разработки игр, такие как платформа разработки игр Unity. Android Studio поддерживает множество языков программирования, включая Java, Kotlin, C++, Python и другие. Android Studio поддерживает интеграцию с системами контроля версий, такими как Git, что упрощает контроль и управление исходным кодом приложения. Android Studio включает в себя мощный профилировщик для оптимизации производительности приложения, а также для обнаружения и исправления ошибок. Android Studio использует систему сборки Gradle, которая позволяет управлять зависимостями проекта и настройками сборки, упрощая процесс разработки.

Данная среда разработки доступна для Windows, macOS и GNU/Linux.

Android Studio бесплатна и имеет открытый исходный код, и ее можно загрузить с официального сайта разработчиков Android.

Сравнение сред разработки

По итогам изучения возможностей различных сред разработки составлена таблица 2.2.

Таблица 2.2 – Сравнение сред разработки

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Среда разработки  Критерий | IntelliJ IDEA [23] | Eclipse [24] | Visual Studio Code [25] | Atom [26] | Vim [27] | Android Studio [28] |
| Наличие эмулятора для запуска Android-приложения | Нет | Нет | Нет | Нет | Нет | Да |
| Наличие бесплатных инструментов для разработки | Да | Да | Да | Нет | Да | Да |

Продолжение таблицы 2.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Среда разработки  Критерий | IntelliJ IDEA [23] | Eclipse [24] | Visual Studio Code [25] | Atom [26] | Vim [27] | Android Studio [28] |
| Сборка приложений, основанная на Gradle | Да | Нет | Нет | Нет | Нет | Да |
| Инструменты для работы с UI | Нет | Нет | Нет | Нет | Нет | Да |
| Инструменты для тестирования | Да | Да | Да | Нет | Да | Да |
| Интуитивно-понятный интерфейс | Да | Нет | Нет | Нет | Нет | Да |
| Удобство работы с плагинами | Да | Да | Да | Нет | Да | Да |
| Простота освоения | Да | Нет | Да | Нет | Нет | Да |

В результате сравнения, выбрана интегрированная среда разработки Android Studio, которая является официальной и рекомендованной средой разработки под OC Android, так как в ней, в отличие от IntelliJ IDEA, имеет эмулятор для запуска Android-приложения и инструменты для работы с UI.

* 1. Особенности разработки Android приложений

Для разработки мобильного приложения для просмотра расписания МИЭТ использовались инструменты Android Studio, а также Compose Navigation и Jetpack Compose.

В Jetpack Compose построение интерфейса основывается на графе Composable функций. После построения граф изменяется на основе состояний, при этом Jetpack Compose определяет какую часть графа необходимо, если нужно, изменить.

Compose Navigation необходим для переходов между экранами, с помощью построенного графа навигации.

Также для создания архитектуры приложения использовался паттерн MVC.

Набор инструментов Jetpack Compose

Jetpack Compose – это фреймворк для разработки пользовательских интерфейсов на языке программирования Kotlin для платформы Android. Он позволяет создавать пользовательские интерфейсы на основе модели декларативного программирования, что упрощает и ускоряет для разработчиков создание и поддержку пользовательских интерфейсов.

Jetpack Compose позволяет разработчикам создавать пользовательские интерфейсы с такими компонентами, как текстовые поля, кнопки и изображения, используя язык Kotlin и функциональное программирование. Он предоставляет ряд компонентов, таких как Column, Row, Box и другие, для создания гибких пользовательских интерфейсов.

Одним из ключевых преимуществ Jetpack Compose является модель декларативного программирования. Это означает, что вместо написания множества строк кода для создания пользовательского интерфейса разработчик может использовать объявления компонентов и их свойств для описания того, что отображается на экране.

Jetpack Compose также предоставляет множество встроенных компонентов, таких как текст, кнопка, изображение, строка, столбец, блок и другие, которые можно использовать для создания интерфейсов. Эти компоненты настраиваются и могут использоваться для создания любого пользовательского интерфейса.

Jetpack Compose также легко интегрируется со средами разработки на основе IntelliJ IDEA, такими как Android Studio. Он поддерживает создание пользовательских компонентов, что позволяет создавать гибкие и масштабируемые пользовательские интерфейсы.

Для создания интерфейса с помощью Jetpack Compose используются следующие компоненты.

1. Row [29]

Контейнер для размещения вложенных компонентов в строку. Функция Row принимает параметры:

* modifier, который позволяет настроить внешний вид и поведение компонента;
* verticalAlignment, который устанавливает выравнивание компонента по вертикали;
* horizontalArrangement, который устанавливает выравнивание компонента по горизонтали;
* content, который представляет вложенное содержимое.

Например, Row использовалось для размещения кнопок (рисунок 2.1).

Row(modifier = Modifier

.fillMaxSize(),

horizontalArrangement = Arrangement.SpaceBetween

){

ScheduleButton(navController, alpha1)

TaskButton(navController,alpha2)

}



Рисунок 2.1 – Кнопки

1. Column [30]

Контейнер для размещения вложенных компонентов в столбец. Функция Column принимает параметры:

* modifier, который позволяет настроить внешний вид и поведение компонента;
* verticalArrangement, который устанавливает выравнивание компонента по вертикали;
* horizontalAlignment, который устанавливает выравнивание компонента по горизонтали;
* content, который представляет вложенное содержимое.

Как один из примеров использования: расположение ФИО с группой в шапке (рисунок 2.2).

Column(modifier = Modifier

.padding(10.dp)

.weight(0.85f)

) {

Text(

name,

fontSize = 16.sp,

fontWeight = FontWeight.Normal,

fontFamily = centurygothic,

color = Color.White

)

Text(

group,

fontSize = 16.sp,

fontWeight = FontWeight.Normal,

fontFamily = centurygothic,

color = Color.White,

modifier = Modifier.alpha(0.9f)

)

}



Рисунок 2.2 – Шапка

1. Box [31]

Простой контейнер, который позволяет позиционировать вложенные компоненты. Функция Box принимает параметры:

* modifier, который позволяет настроить внешний вид и поведение компонента;
* contentAlignment, который устанавливает расположение компонента;
* propagateMinConstraints, который указывает, надо ли применять к содержимому ограничения по минимальным размерам
* content, который представляет вложенное содержимое.

В МП ПР Box использовался для создания кнопки поиска (рисунок 2.1)

Box(modifier = Modifier

.offset(0.dp, 30.dp)

.background(colorResource(R.color.orioks), CircleShape)

.fillMaxWidth(0.27f)

.aspectRatio(1f, true),

contentAlignment = Alignment.Center

Image(

painter = painterResource(R.drawable.search\_inactive),

contentDescription = "search",

modifier = Modifier

.align(Alignment.Center)

.offset(0.dp, -15.dp)

)

}

Объект Modifier используется для настройки внешнего вида компонентов. Основные и часто используемые его функции, и область их применения представлены в таблице 2.3.

Таблица 2.3 – Функции Modifier

|  |  |  |
| --- | --- | --- |
| Область применения | Функция | Назначение [32] |
| Любая | border() | Добавление границы |
| Любая | alpha() | Рисование компонента с прозрачностью |
| Любая | background() | Рисование компонента с заданным цветом фона |
| Любая | clip() | Рисование компонента в заданной форме |
| Любая | padding() | Добавление дополнительного пространства вдоль края компонента |
| Любая | offset() | Смещение компонента |
| Любая | aspectRatio() | Изменение размера компонента в соответствии с заданным соотношением сторон |
| Любая | fillMaxSize() | Компонент принимает максимально возможный размер по ширине и высоте |
| Любая | fillMaxHeight() | Компонент принимает максимально возможный размер по высоте |
| Любая | fillMaxWidth() | Компонент принимает максимально возможный размер по ширине |
| ColumnScope/  RowScope | weight() | Размер высоты/ширины компонента пропорционален его отношению к другим взвешенным компонентам |
| Любая | clickable() | Настройка компонента для получения кликов через событие ввода или нажатия |

Таким образом Jetpack Compose значительно упрощает разработку пользовательского интерфейса, не ограничивая разработчика в возможностях создания интерфейса разнообразного дизайна. Кроме того, Jetpack Compose предоставляет возможность создания сложной анимации, что практически невозможно сделать с помощью XML.

Состояния

Состояния – это данные, которые могут изменяться во время работы приложения и влиять на поведение пользовательского интерфейса. В Jetpack Compose состояния задаются с помощью функции remember.

@Composable

fun Count

() {

val counterState = remember { mutableStateOf(0) }

Button(onClick = { counterState.value++ }) {

Text("Вы нажали ${counterState.value} раз")

}

}

В этом примере функция remember используется для создания изменяемого состояния counterState, содержащего значение счетчика. Когда пользователь нажимает кнопку, значение счетчика увеличивается на единицу, и изменение автоматически отображается в пользовательском интерфейсе.

Состояния Jetpack Compose бывают нескольких видов. MutableState – это изменяемое состояние, которое может привести к перерисовке пользовательского интерфейса при изменении его значения. DerivedState – состояние, вычисляемое на основе другого состояния или данных из внешнего источника. Он не имеет внутренней ценности и перерисовывается только при изменении его зависимостей. SideEffect – состояние, используемое для выполнения побочных эффектов, таких как загрузка данных из сети или обновление базы данных. Это не приводит к перерисовке пользовательского интерфейса, но может повлиять на его состояние.

Jetpack Compose предоставляет множество встроенных функций состояния, таких как mutableStateOf, derivedStateOf и onCommit. Они позволяют управлять поведением пользовательского интерфейса в зависимости от состояния приложения и создавать интерактивные и анимированные элементы пользовательского интерфейса.

Compose Navigation

Состояния необходимы для изменения отдельных компонентов на экране. Для переходов между окнами необходимо использовать Compose Navigation. Компонент Navigation обеспечивает поддержку приложений Jetpack Compose, перемещаясь между составными объектами, используя преимущества инфраструктуры и функций компонента навигации [33].

Jetpack Compose Navigation – это библиотека, которая позволяет создавать навигационную структуру в приложениях, использующих Jetpack Compose. Она позволяет легко создавать и управлять навигационным графом приложения, переходить между экранами и передавать данные между ними.

Для начала создается граф навигации. Для этого задается navController, начальное окно, а также функция, отвечающая за отображение каждого экрана

NavHost(navController = navController,

startDestination = Screen.Auth.route

) {

composable(

route = Screen.Auth.route

){

AuthScreen(navController)

}

composable(

route = Screen.Schedule.route

){

ScheduleScreen(navController)

}

composable(

route = Screen.Task.route

){

TaskScreen(navController)

}

}

Также между экранами можно передавать агрументы. Для этого помимо пути в composable указывается название и тип аргумента.

composable(

route = Screen.TaskInfo.route,

arguments = listOf(

navArgument("name"){

type = NavType.StringType

}

)

){

val name = it.arguments?.getString("name")

TaskInfoScreen(navController, name)

}

Для перехода на другой экран в функции clickable объекта указывается путь перехода, а также опция, чтобы экраны не наслаивались друг на друга [34].

modifier = Modifier

.clickable {

navController.navigate(Screen.Task.route) {

popUpTo(Screen.Task.route) {

inclusive = true

}

}

}

Паттерн MVC

MVC (Model-View-Controller) – это паттерн, обычно используемый для реализации пользовательских интерфейсов, данных и управляющей логики. Он подчеркивает разделение между бизнес-логикой программного обеспечения и отображением [35].

Три части паттерна проектирования MVC:

* Model: управляет данными и бизнес логикой;
* View: управляет макетом и отображением;
* Controller: направляет команды к Model и View.

Model определяет, какие данные содержит приложение и при изменении данных уведомляет View и Controller.

View определяет, как отображаются данные приложения.

Controller содержит логику, которая обновляет модель и представлений в ответ на ввод данных от пользователя.

Шаблон MVC (Model-View-Controller) упрощает разделение приложения на компоненты, что позволяет их независимую модификацию и управление состоянием. Он также обеспечивает расширяемость приложения и упрощает управление и расширение проекта.

* 1. Разработка мобильного приложения МП ПР

Разработка интерфейса приложения

После успешной авторизации в системе ОРИОКС, пользователь сразу видит свое расписание на текущий день. Чтобы просмотреть расписание на другие дни, достаточно нажать на соответствующие дни в верхней части экрана. Если пользователь хочет просмотреть расписание на другую неделю, он может открыть календарь, нажав на соответствующую кнопку внизу. Вверху календаря указан тип текущей недели. Для управления аккаунтом и выхода из него, пользователь может нажать на панель с его именем и фамилией. Повторное нажатие на данную панель вернет пользователя на предыдущий экран (рисунок 2.3).

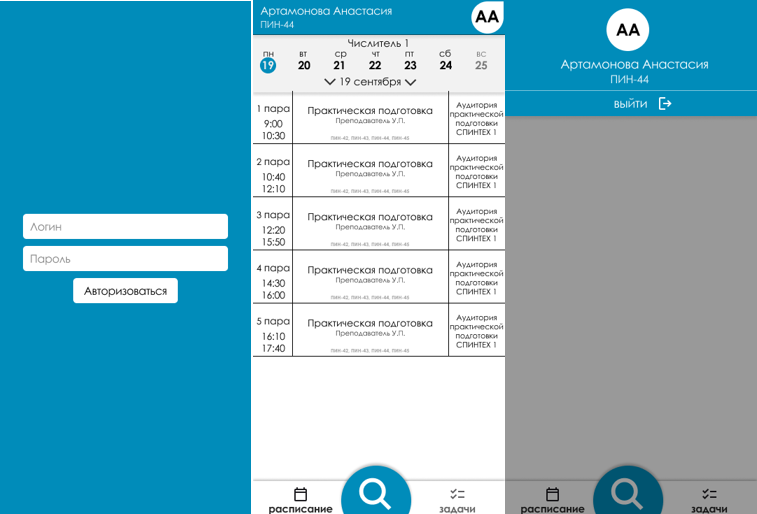


Рисунок 2.3 – Экранные формы авторизации

При активации кнопки "Задачи" пользователь получает доступ к удобному окну, где он может ознакомиться со списком добавленных задач и добавить новые. Задачи отображаются в порядке возрастания времени до их выполнения, что помогает пользователю организовать свою работу. При выборе определенной задачи открывается окно с полным описанием и дополнительными комментариями, предоставляя более подробную информацию о задаче. В правом верхнем углу экрана размещена кнопка, позволяющая пользователю редактировать задачу, внося изменения по необходимости. Простым свайпом задачи влево пользователь может удалить ее, обеспечивая удобный способ управления своим списком задач (рисунок 2.4).

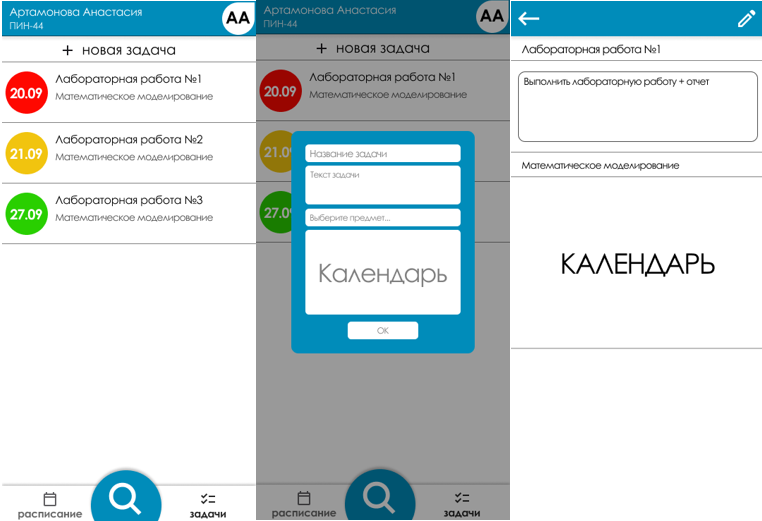


Рисунок 2.4 – Экранные формы задач

При выполнении свайпа вверх или при нажатии кнопки "Поиск" пользователь открывает окно поиска, где ему предоставляется широкий функционал для поиска расписания конкретной группы, расписания определенного преподавателя или нахождения свободных окон. При активации кнопки поиска система производит соответствующий поиск, а при совершении свайпа вниз пользователь возвращается к предыдущему экрану с задачами или рассмотрением расписания (рисунок 2.5).

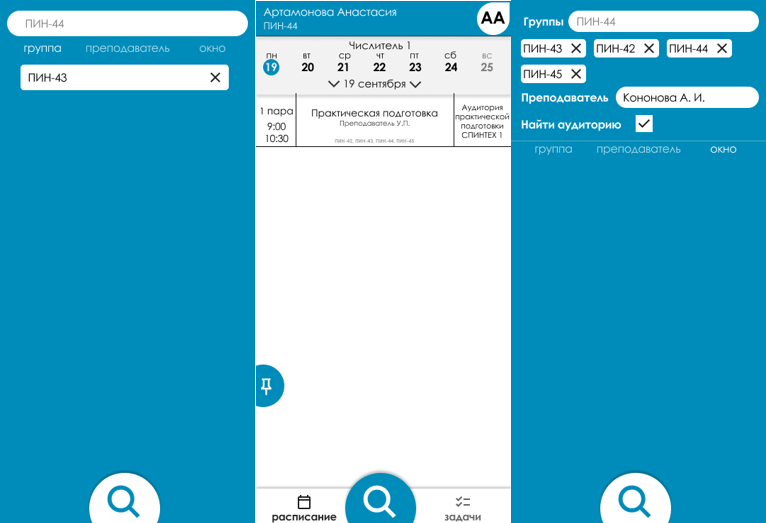


Рисунок 2.5 – Экранные формы поиска

Важно отметить, что, если обнаруживается расписание для другой группы или преподавателя, пользователь имеет возможность закрепить его для быстрого и удобного доступа в будущем

Разработка моделей приложения

Программная реализация МП ПР стоит из классов двух видов (классы для отрисовки графических интерфейсов активностей и классы-контейнеры), манифеста и файлов ресурсов.

На рисунке 2.6 представлена обобщенная UML диаграмма классов мобильного приложения для просмотра.

Связи между классами показаны стрелками. Стрелка с ромбом означает связь агрегации, стрелка с концом треугольником – связь наследования, пунктирная стрелка – связь зависимости [36]. Например, TaskScreen и TaskInfoScreen используют объект класса Task в своих методах.

В архитектуре приложения применяются различные классы-модели, включая Group, Schedule, Week, Day, Lesson и Task. Каждый из этих классов выполняет свою функцию в рамках логики приложения. Например, класс Task используется для хранения информации о задачах, а классы Group, Schedule, Week, Day, Lesson предназначены для хранения и представления расписания пользователей, групп и преподавателей.

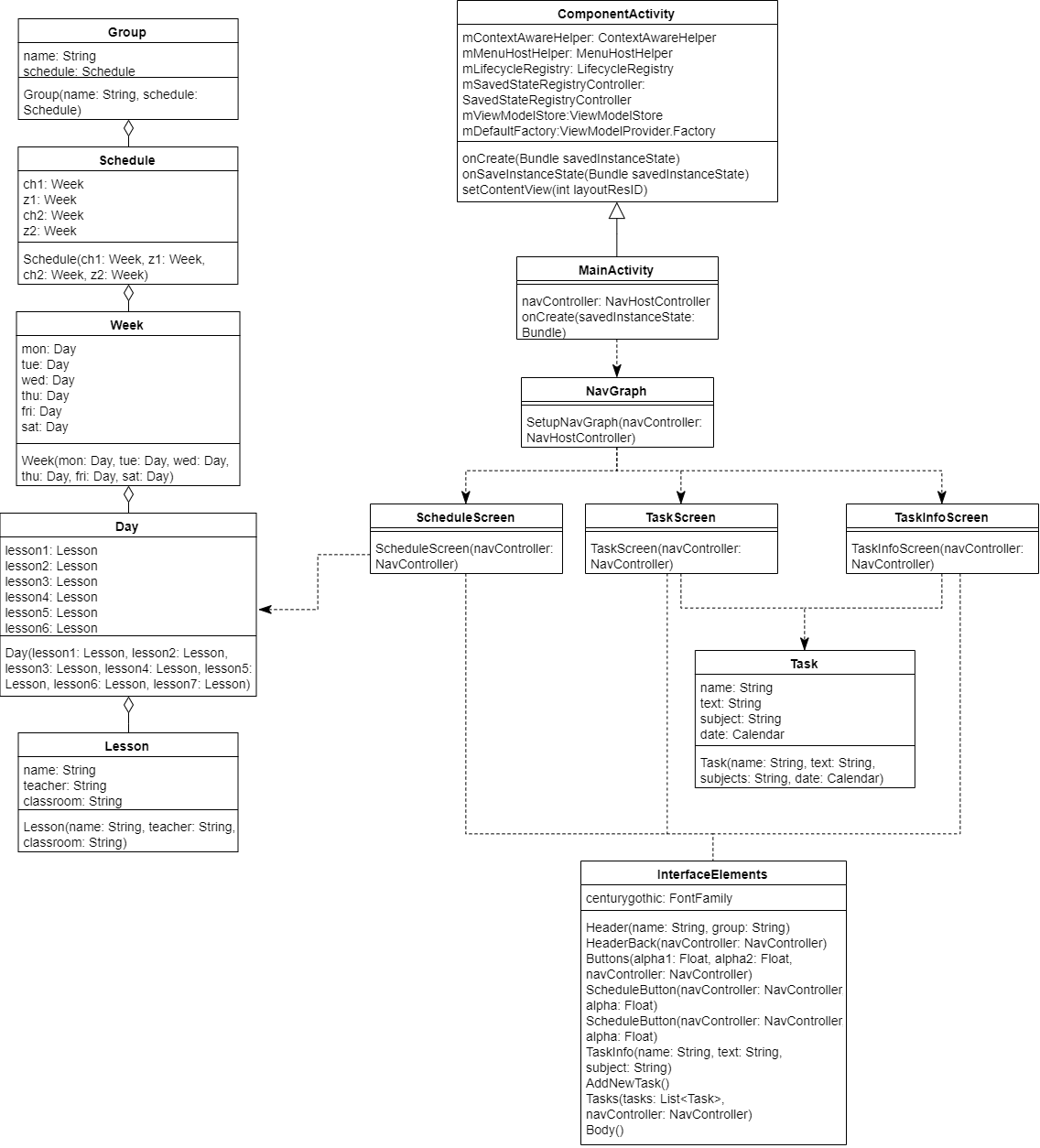
Главной активностью приложения является MainActivity, который наследуется от ComponentActivity. Он играет ключевую роль и отображается в самом начале работы приложения [37].

Рисунок 2.6 – Диаграмма классов

Для управления навигацией между экранами используется класс NavGraph, который представляет собой граф переходов между различными экранами приложения.

Каждое окно приложения имеет свой собственный класс, отвечающий за его отображение и функциональность. Например, ScheduleScreen отображает расписание пользователя и содержит кнопки для перехода на другие экраны, а TaskScreen позволяет пользователю просматривать список задач и создавать новые задачи. Для подробного просмотра информации о задаче используется класс TaskInfoScreen, где пользователь может увидеть все детали задачи и внести необходимые изменения.

Класс InterfaceElements содержит различные элементы интерфейса, которые используются всеми классами-окнами приложения, обеспечивая единообразный стиль и функциональность интерфейса.

Выводы по конструкторскому разделу

В качестве языка реализации МП ПР выбран язык программирования Kotlin, поскольку он отвечает выдвинутым требованиям. Для разработки мобильного приложения выбрана среда разработки Android Studio.

В ходе реализации МП ПР изучены особенности разработки под операционную систему Android. Обозначены составные части нативного Android-приложения. Разработаны экранные формы пользовательского интерфейса МП ПР. Сгенерирована UML-диаграмма классов приложения, демонстрирующая программную реализацию поставленной задачи.

МП ПР прошло апробацию путем опубликования тезисов доклада: Артамонова А. Разработка мобильного приложения для просмотра расписания МИЭТ. Актуальные проблемы информатизации в цифровой экономике и научных исследованиях. Ⅲ Научно-практическая конференция с международным участием: тезисы докладов. М.: МИЭТ, 2022. 8 с.

1. ИСПЫТАТЕЛЬНЫЙ РАЗДЕЛ
   1. Отладка

Отладка – это процесс поиска и исправления ошибок или неполадок в исходном коде какого-либо программного обеспечения [38].

Android Studio позволяет отлаживать приложения на эмуляторе и на реальном устройстве. Кроме того, в IDE есть возможность просмотреть системный журнал логов, установить точку, проверять значения переменных и вычислять выражения во время работы, делать скриншоты и видеозаписи.

Точка остановы

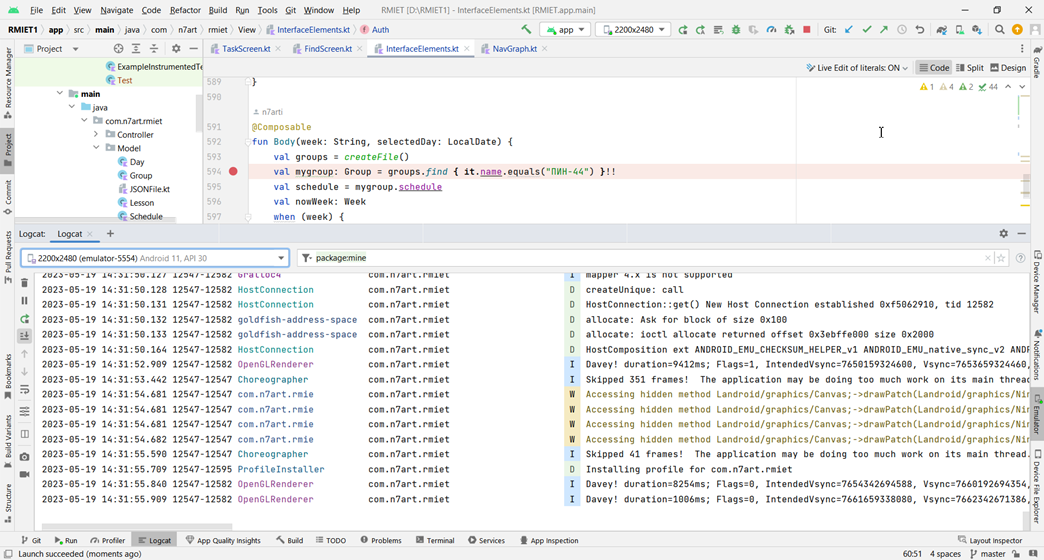
Точка останова является мощным инструментом отладки, который позволяет контролировать выполнение программы и анализировать ее состояние на конкретной строке кода. Она позволяет приостановить выполнение программы в заданной точке, что дает возможность проверить текущие значения переменных, выполнить дополнительные вычисления или проверить правильность выполнения определенного выражения. Пример работы с точкой остановы показан на рисунке 3.1.

Рисунок 3.1 – Точка остановы

В отличие от полной остановки, с помощью точки остановы после работы в отладчике программу можно прервать или продолжить с того же места, где она остановлена.

Отслеживание потребления памяти

Android Studio позволяет отслеживать потребление памяти объектами и показывает, какие классы и потоки используют объекты. С помощью данного инструмента можно найти утечки памяти и выявить неэффективное ее использование. Также можно сделать один или несколько снимков управляемой и собственной памяти в куче, чтобы понять влияние использования памяти типов объектов.

Android Device Monitor

Для анализа потребления памяти, сетевого трафика, поведения приложений при входящих звонках можно использовать графический инструмент Android Device Monitor (рисунок 3.2).

Для профилирования процессора, памяти и использования сети вашим приложением используется Android Profiler.

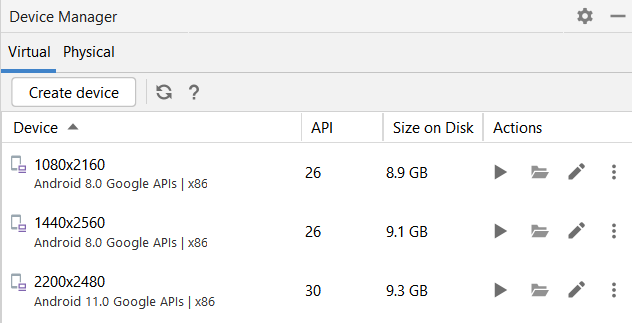
Чтобы проверить файлы .trace, захваченные путем инструментирования вашего приложения с помощью класса Debug, захватить новые трассировки методов, экспортировать файлы .trace и проверить использование ЦП процессами вашего приложения в режиме реального времени, используется CPU Profiler в Android Studio.

Рисунок 3.2 – Device Manager

Чтобы управлять иерархией внешнего вида приложения во время выполнения, используйте инспектор компоновки.

Для профилировки скорость отрисовки вашего приложения, используется Window.OnFrameMetricsAvailableListener.

* 1. Тестирование

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

В мобильном тестировании проверяется функциональность, производительность, безопасность и удобство. Есть ручное и автоматизированное тестирование.

Инструменты для автоматизированного тестирования Android приложений: Appium, Selendroid, Espresso, Robotium, MonkeyRunner, BrowserStack, Ranorex, UI Automator, OWASP Zed Attack Proxy и инструменты семейства Android Studio [39].

Этапы тестирования Android приложений

Планирование. На этом этапе описывается масштаб/охват тестирования, тестовое покрытие, покрытие девайсов, ресурсы, необходимые для тестирования, примерные дедлайны и другие, зависящие от проекта вещи. Также решается, в каком объеме проводится автоматизация и какие тестовые сценарии автоматизировать.

Настройка тестового окружения. Необходимо подобрать соответствующий набор тестовых устройств. Также на этом этапе настраиваются эмуляторы/симуляторы и облачные девайсы.

Написание тест-кейсов и сценариев. После решения, какая функциональность покрыта тестами, пишутся тест-кейсы. Тест-кейс – это список этапов проверки, ведет ли себя приложение, как положено в некой ситуации.

Выполнение тестов. Этап тестирования начинается с функционального тестирования, где проверяются, что вся ключевая функциональность приложения работает как положено.

На этапе тестирования интерфейса и используемости проверяют user experience и удостоверяются, что приложение простое и интуитивное в использовании, а также, что в интерфейсе нет явных ошибок.

Тестирование совместимость проверяет, правильно ли работает приложение на предполагаемых устройствах и операционных системах.

Тестирование производительности важный этап в тестировании любого приложения. Типы тестирования производительности: volume-тестирование; стресс-тестирование; тестирование стабильности; нагрузочное тестирование; спайк-тестирование. Тестирование производительности проверяет поведение приложение при ожидаемой средней рабочей нагрузке и при нагрузке значительно выше средней. Это тестирование выполняется с использованием специальных инструментов.

Дальше выполняется тестирование безопасности. Тестирование безопасности – это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным [40].

Кроме упомянутых тестов, могут выполнятся: инсталляционное тестирование, тестирование обновлений, тестирование прерываний, тестирование восстановления, тестирования ресурсов устройства, тестирование сетевых конфигураций.

На последнем этапе создается итоговый отчет по результатам тестирования. Найденные и исправленные ошибки анализируется и определяются самые проблемные модули приложения.

Особенности тестирования Android приложений

Разработка Android приложений имеет свои особенности.

Во-первых, необходимо тестировать приложение на большом количестве устройств, так как производители Android имеют возможность модифицировать операционную систему.

Во-вторых, устройства Android поставляются с разными размерами и типами экранов, поэтому необходимо протестировать приложение на различных разрешениях.

В-третьих, устройства выпускаются с разными характеристиками, поэтому нужно учитывать, как работает приложение с аппаратными ресурсами устройства.

В-четвертых, существуют устройства с разными версиями ОС, поэтому надо подразумевать, чтобы приложение работало на большинстве версий.

В-пятых, приложения для Android устройств можно получить не только из официального магазина, но и из альтернативных источников, поэтому необходимо тщательно протестировать приложение на безопасность.

Инструменты для модульного тестирования

Junit

JUnit – это фреймворк, предназначенный для автоматического тестирования программ. Его основное назначение – unit-тестирование.

Модульное тестирование с помощью JUnit – это тип тестирования, который обрабатывает каждый компонент по отдельности. Это помогает избежать ошибок на самом низком уровне, то есть на уровне работоспособности отдельных объектов.

Один из недостатков Junit – отсутствие встроенных mock-объектов.

Mockk

MockK – это библиотека для языка Kotlin, которая имитирует поведение реальных объектов и позволяет легко модифицировать зависимости в тестируемых классах с помощью заглушек и моков (Stubs and Mocks).

kotlinx-coroutines-test

kotlinx-coroutines-test – это тестовые утилиты для .kotlinx.coroutines. Этот пакет предоставляет утилиты для эффективного тестирования сопрограмм. Тестирование функции приостановки в TestCoroutineScope, предоставленную функцией runBlockingTest, изменив основной диспетчер в тесте [41].

turbine

Небольшая, но полезная библиотека для тестирования Kotlinx.coroutines.Flow.

Robolectric

Robolectric – стандартная среда модульного тестирования Android. Robolectric позволяет запускать тесты в смоделированной среде Android внутри JVM без накладных расходов и нестабильности эмуляторов. Тесты Robolectric обычно выполняются в 10 раз быстрее, чем эмуляторы с холодным запуском.

* + 1. Инструменты тестирования пользовательского интерфейса

Appium

Appium построен на идее, что тестирование приложения не требует включения SDK или его перекомпиляции. В Appium есть возможность использовать любимые методы тестирования, фреймворки и инструменты. Это проект с открытым исходным кодом, в котором приняты решения по дизайну и инструментам для поощрения активного сообщества [42].

Appium стремится автоматизировать любое мобильное приложение на любом языке и в любой тестовой среде с полным доступом к серверным API и базам данных из тестового кода.

Appium работает значительно медленнее драйверов, так как по сути является дополнительной прослойкой между кодом тестов и самими драйверами. При большом количестве тестов проблема становится заметнее.

Selendroid

Selendroid – это платформа автоматизации тестирования для нативных и гибридных приложений Android и мобильных веб-интерфейсов пользователя. Тесты пишутся с использованием клиентского API Selenium 2 [43].

Selendroid можно использовать на эмуляторах и реальных устройствах, а также интегрировать как узел в Selenium Grid для масштабирования и параллельного тестирования.

Недостаток этого инструмента в том, что он довольно медленный, и на некоторых машинах с объемом оперативной памяти менее 4 ГБ его невозможно использовать.

Espresso

Espresso – это инструмент для автоматизации тестирования пользовательского интерфейса в Android-приложениях [44].

Он предлагает разработчикам возможность создавать компактные и надежные тесты, которые обеспечивают проверку ожиданий, взаимодействие с элементами интерфейса и проверку утверждений.

Основной API Espresso является лаконичным, интуитивно понятным и простым в освоении. Он предлагает гибкую настройку и позволяет точно управлять тестовыми сценариями, не отвлекаясь на шаблонный контент или детали реализации. Espresso также предлагает эффективное выполнение тестов, минимизируя ожидания, синхронизацию и задержки.

Однако, стоит отметить некоторые ограничения Espresso. Для использования этого инструмента требуется доступ к исходному коду тестируемого приложения. Кроме того, Espresso не предоставляет возможности для взаимодействия с другими приложениями, что может быть ограничением в некоторых сценариях тестирования. Также следует учитывать, что при работе с сложными асинхронными интерфейсами или списками, стабильность работы Espresso может быть немного нарушена.

Robotium

Robotium – это среда автоматизации тестирования для платформы Android, которая обладает полной поддержкой нативных и гибридных приложений. Она позволяет разработчикам создавать эффективные, надежные и автоматизированные тесты пользовательского интерфейса для приложений на Android. С помощью Robotium можно легко создавать разнообразные тестовые сценарии, включая функциональные, системные и пользовательские сценарии приемочных испытаний, которые охватывают различные действия в Android-приложениях [45].

Однако, у Robotium также есть некоторые недостатки. Например, он не может обрабатывать Flash-элементы или веб-компоненты, и может работать только с одним приложением за раз. Также он ограничен в имитации нажатий на клавиатуру.

UI Automator

UI Automator - это среда для тестирования пользовательского интерфейса, которая предназначена для функционального тестирования взаимодействия между приложениями в системе и установленными приложениями. Его API позволяет выполнять различные действия, такие как открытие настроек или запуск приложений на тестовом устройстве, что позволяет взаимодействовать с элементами пользовательского интерфейса независимо от конкретного сценария.

Одним из преимуществ UI Automator является возможность легкого нахождения компонентов пользовательского интерфейса с использованием удобных идентификаторов, таких как текстовые значения или описания содержимого. Это облегчает создание тестовых сценариев и проверку корректности отображения элементов на экране.

Однако, среди недостатков UI Automator можно выделить сложный и универсальный операционный механизм, который может повлиять на производительность из-за использования IPC Binder. Это может вызывать определенные проблемы в некоторых случаях.

* 1. Выбор инструмента тестирования

В результате анализа существующих инструментов тестирования составлена таблица 3.1, в которой сравниваются инструменты для модульного тестирования, и таблица 3.2, в которой сравниваются инструменты для UI тестирования.

В результате анализа вышеперечисленных программ, составлен перечень функций, которыми обязан обладать искомый инструмент модульного тестирования:

* параллельный запуск тестов;
* стабильность работы инструмента;
* популярность инструмента тестирования;
* знакомство разработчика с инструментом тестирования.

Таблица 3.1 – Сравнение инструментов для модульного тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Инструмент  Критерий | Robolectric [46] | Mokk [41] | kotlinx-coroutines-test [47] | turbine [48] | JUnit [49] |
| Параллельный запуск тестов | Нет | Нет | Нет | Нет | Да |
| Стабильность | Нет | Да | Да | Да | Да |
| Популярность инструмента | Нет | Нет | Нет | Нет | Да |
| Знакомство с инструментов | Нет | Нет | Нет | Нет | Да |

После проведения сравнения доступных инструментов для модульного тестирования, был выбран инструмент Junit. Этот выбор основан на соответствии Junit перечисленным требованиям и на знакомстве разработчика с этим инструментом. Junit является широко распространенным и надежным фреймворком для модульного тестирования в экосистеме Android. Он предоставляет простой и понятный синтаксис для написания тестов и обладает мощным функционалом для проверки различных аспектов программного кода. Благодаря своей популярности и широкой поддержке сообщества, Junit является надежным выбором для разработчиков при модульном тестировании и обеспечивает уверенность в качестве кода.

Для искомого инструмента тестирования пользовательского интерфейса составлены следующие требования:

* принадлежность к семейству Android Studio;
* быстрое выполнение тестов;
* обработка Flash компонентов;
* обработка нескольких приложений одновременно;
* стабильность работы инструмента;

Таблица 3.2 – Сравнение инструментов тестирования пользовательского интерфейса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Инструмент  Критерий | Appium [42] | Selendroid [43] | Espresso [44] | Robotium [45] | UI Automator [50] |
| Принадлежность к семейству Android Studio | Нет | Нет | Да | Нет | Да |
| Быстрое выполнение тестов | Нет | Нет | Да | Да | Да |
| Обработка Flash компонентов | Да | Да | Да | Нет | Да |
| Обработка нескольких приложений одновременно | Да | Да | Да | Нет | Да |
| Стабильность работы | Да | Да | Нет | Да | Да |

Для тестирования пользовательского интерфейса выбран UI Automator, так как инструмент принадлежит семейству Android Studio, а, следовательно, быстро подключается к среде разработки.

Методы и техники тестирования

Тестирование методом черного ящика

Тестирование методом «черного ящика» или поведенческое тестирование – это метод тестирования функционального поведения объекта с точки зрения внешнего мира без использования каких-либо знаний о тестируемом внешнем объекте. Стратегия относится к систематическому способу выбора и построения тестов для набора тестов. Стратегия эксплуатационных испытаний основывается на технических требованиях и их характеристиках.

Под «черным ящиком» понимается объект исследования, внутренняя структура которого неизвестна. Некоторые поиски можно выполнять, манипулируя только входными и выходными данными.

«Черный ящик» – это сложные гомоморфные модели кибернетических систем, учитывающие разнообразие. Удовлетворительная модель системы может быть получена только в том случае, если она содержит достаточно информации, чтобы отразить разнообразие системы. Чем больше количество возмущений, действующих на входы модели системы, тем больше разнообразие регулятора.

Техники тестирования метода чёрного ящика

Эквивалентное разбиение (Equivalence partitioning) – это метод тестирования черного ящика. Идея разделенного тестирования классов эквивалентности состоит в том, чтобы исключить наборы входных данных, которые заставляют системы вести себя одинаково и дают одинаковый результат при тестировании программы.

Разбиение на разделы – это распространенный метод тестирования «черного ящика», целью которого является сокращение числа избыточных тестовых случаев за счет исключения тестовых наборов, которые дают одинаковые результаты и не обязательно выявляют дефекты в функциональности программы.

Цель тестирования – найти дефекты, поэтому успешный тест-кейс – это тест, обнаруживший дефекты.

Процесс проверки разделения на эквивалентность идентифицирует наборы данных как входные условия, дающие одинаковый результат при запуске программы, классифицирует их как эквивалентные наборы данных и группирует их из другого эквивалентного набора данных [51].

Анализ граничных значений (Boundary value analysis) – это метод проверки работы продукта на предельных значениях входных данных, которые могут быть вне допустимого диапазона. В рамках граничного тестирования также могут быть проведены тесты, которые проверяют, как система обрабатывает такие входные данные, включая их обработку через исключительные ситуации или сообщения об ошибках.

Граничное значение – это значение в конце допустимого диапазона входных данных, которое может вызвать изменение поведения программы. Это минимальные или максимальные значения данного типа данных, значения, близкие к ним, или значения, вызывающие переполнение буфера или другие ошибки. Использование граничных значений в тестах может помогает выявить ошибки при обработке граничных условий. Тестирование с этими значениями выявляет ошибки, связанные с манипулированием граничными значениями.

Цель этой техники – найти ошибки, связанные с граничными значениями [52].

Анализ доменов (Domain analysis) – это методика тестирования, применяемая в методе черного ящика для эффективного и эффективного тестирования множественных параметров. Она основана на методах эквивалентного разбиения и анализа граничных значений.

Суть доменного тестирования заключается в том, чтобы разделить условия тестирования на значения, которые можно считать одинаковыми, и тестировать их более эффективно. К достоинствам метода можно отнести обнаружение ошибок при минимальном количестве тестов и простоту подхода. Однако метод имеет некоторые недостатки, такие как низкую вероятность обнаружения ошибок за пределами границ допустимых значений и сложность разделения условий тестирования на группы.

Цель доменного тестирования заключается в предоставлении стратегии для выбора минимального набора репрезентативных тестов [53].

Тестирование всех пар (All pairs testing, Pairwise) – это методика тест-дизайна черного ящика, при которой создаются тест-кейсы, включающие все возможные комбинации каждой пары входных параметров. В приложениях результат работы зависит от различных факторов, таких как входные параметры, переменные состояния и конфигурации среды. Для определения возможных значений могут использоваться техники, такие как анализ граничных значений и классы эквивалентности. Тестирование всех возможных комбинаций значений для всех факторов непрактично, поэтому генерируется подмножество комбинаций, которые охватывают все комбинации для каждой пары входных параметров.

Попарное тестирование помогает при разработке тестов для приложений, содержащих множество параметров, и позволяет существенно уменьшить количество комбинаций проверок, необходимых для обеспечения необходимого уровня качества программного обеспечения. Это эффективная техника тест-дизайна, которая гарантирует качественное покрытие тестирования при минимальном количестве тестов.

Цель попарного тестирования заключается в том, чтобы уменьшить количество полученных тестов, но при этом сохранить покрытие [54].

Тестирование таблицы решений (Decision table testing) – это метод представления модели со сложной логикой и упорядочения бизнес-требований, которые реализованы в продукте. Он основан на взаимосвязи между множеством условий и действий и позволяет проверять каждый функциональный аспект приложения с помощью таблицы принятия решений.

Такая таблица имеет две оси – вертикальную и горизонтальную – где вертикальная ось содержит функциональные аспекты приложения, а горизонтальная ось перечисляет различные варианты использования. Каждый элемент таблицы указывает, следует ли проверять соответствующий функциональный аспект для данного варианта использования приложения. Таблица принятия решений содержит набор условий, выполнение которых приведет к определенному действию.

Обычно таблица принятия решений разделена на 4 квадранта, где условия – список возможных условий, варианты выполнения действий – комбинация выполнения или невыполнения условий этого списка, а действия – список возможных действий. Необходимость действий указывается для каждой комбинации условий [55].

Тестирование диаграммы переходов состояний (State transition testing) – это метод тестирования "черного ящика", который используется в случаях, когда система может быть описана в терминах "конечного автомата", где система может находиться в разных состояниях, а переходы между ними определяются правилами "машины". Эта модель состоит из четырех основных частей: состояний, переходов, действий, вызывающих переходы и действий возникающих в результате переходов. Этот метод заключается в получении тестовых сценариев из модели переходов между состояниями. Хотя процесс измерения покрытия тестами в этом методе похож на процесс тестирования "белого ящика", тестирование переходных состояний обычно рассматривается как метод "черного ящика" [56].

Графики причинно-следственных связей (Cause-Effect Graphing) – это метод тестирования черного ящика, который визуально демонстрирует связь между результатом и всеми факторами, влияющими на него. Данный метод используется для создания тестовых сценариев, предоставляя формальную запись логических условий и соответствующих действий. Он базируется на автоматном подходе к решению задачи.

При использовании данного метода, на первом шаге для тестируемой программы или модуля определяются причины и следствия, каждому из которых присваивается идентификатор. На втором шаге создается граф причинно-следственных связей, а на третий этот граф преобразуется в таблицу решений. На четвертом шаге столбцы таблицы преобразуются в тестовые варианты. Таким образом, метод графиков причинно-следственных связей позволяет создать набор тестовых сценариев на основе взаимосвязи между различными факторами и результатом, что помогает эффективно протестировать программное обеспечение [57].

Тестирование сценариев использования (Use Case Testing) – это способ тестирования программного обеспечения, который позволяет определить тестовые случаи, охватывающие всю систему от начала до конечной точки, по каждой транзакции. Этот метод основан на анализе путей и переводе требований в тестовые случаи. Каждый сценарий использования описывает несколько различных сценариев поведения, и каждый сценарий может быть потенциальным тестовым случаем, так как он определяет набор ответных реакций системы на заданные внешние воздействия.

Для тестов методом Use Case используется технология «черного ящика». Однако для того, чтобы тест прошел успешно, тестер должен иметь полную информацию обо всех внешних интерфейсах тестируемого модуля. Этот метод тестирования помогает выявить ошибки, которые могут возникнуть при использовании программного обеспечения реальными пользователями. Тестирование вариантов использования можно проводить на любом этапе жизненного цикла разработки программного обеспечения, но оно наиболее эффективно на этапе функционального тестирования [58].

Тестирование историй использования (User Story Testing) – это один из методов тестирования программного обеспечения, основанный на использовании пользовательских историй. Пользовательские истории – это краткие описания функций, которые система реализует с точки зрения пользователей. Каждая пользовательская история обычно состоит из заголовка и описания, содержащего информацию о требованиях и ожиданиях пользователя от системы.

Тестирование вариантов использования состоит в создании тестовых случаев на основе пользовательских историй и проверке того, что реализация системы соответствует этим требованиям. При тестировании вариантов использования тестировщики создают тестовые сценарии, включающие предварительные условия, действия пользователя и ожидаемые результаты. Это помогает убедиться, что система работает так, как описано в пользовательской истории.

Тестирование истории использования может быть выполнено в методологиях гибкой разработки, таких как Scrum, где пользовательские истории используются для определения задач, которые нужно выполнить в итерации. Кроме того, тестирование вариантов использования может быть полезно позже в жизненном цикле разработки, когда требования к системе уже уточнены и необходимо убедиться, что система соответствует этим требованиям.

Основное преимущество тестирования истории использования заключается в том, что тестирование выполняется с точки зрения конечного пользователя системы. Это помогает обнаруживать баги и ошибки в системе, которые могут быть пропущены другими методами тестирования [59].

Тестирование методом белого ящика

Тестирование методом «белого ящика» или структурное тестирование – это тестирование, которое рассматривает внутренние механизмы системы или компонента.

Обычно он включает в себя тестирование ветвей, маршрутов и операторов. Во время тестирования выбираются входные данные и выполняются различные части кода для определения ожидаемого результата.

Критерии покрытия кода:

* область действия объявления: выполняется и тестируется ли каждая строка исходного кода;
* покрытие условий: выполняется и проверяется ли каждая точка принятия решения (вычисление истинности или ложности утверждения);
* покрытие пути: выполнены ли и протестированы ли все возможные пути через данный фрагмент кода;
* функциональная область: выполняется ли каждая функция программы;
* область ввода/вывода: выполняются ли все вызовы функций и какие-либо возвраты из них;
* диапазоны значений параметров: проверены ли все типовые значения и пределы параметров.

Тестирование методом «белого ящика» традиционно выполняется на уровне модулей, но еще используется для системной интеграции и системного тестирования, тестирования внутри устройства и тестирования путей между устройствами. Этот метод тестирования не может выявить недостающие части спецификации, отсутствующие требования или неправильную структуру приложения.

Техники тестирования метода белого ящика

Тестирование покрытия операторов (Statement testing and coverage) – это метод тестирования программного обеспечения, который измеряет, сколько программных инструкций выполнено в процессе тестирования. Он используется для оценки качества тестов и определения того, насколько хорошо тестируется код.

Для проверки полноты программных инструкций используются специальные инструменты, способные отслеживать, какие инструкции выполнялись при выполнении тестового скрипта. Эти инструменты создают отчеты о покрытии кода, которые помогают разработчикам и тестировщикам понять, какие части кода не тестировались и нуждаются в дальнейшем тестировании.

Тестирование покрытия операторов помогает выявить проблемы в коде, такие как отсутствующие проверки ошибок или неиспользуемый код. Это также помогает определить, какие тесты добавить для улучшения покрытия кода.

Покрытие операторов не является полным индикатором качества теста и не гарантирует, что код является безошибочным. Это только помогает разработчикам и тестировщикам определить, какие части кода нуждаются в дополнительном тестировании, чтобы увеличить вероятность обнаружения ошибок [60].

Тестирование покрытия условий и решений (Decision testing and coverage) – это один из методов оценки эффективности тестирования программного обеспечения. Он помогает определить, насколько тесты охватывают все возможные ветви логики приложения, и выявить ошибки, связанные с принятием решений.

Для проверки покрытия условий и решений используется метод, называемый покрытием условий/решений. Он включает в себя два типа покрытия: покрытие условий и покрытие решений.

Покрытие условий измеряет, насколько хорошо тесты покрывают все возможные значения истинности для каждого логического выражения в приложении. Для достижения полного условного покрытия каждый логический оператор должен быть проверен как минимум дважды: true и false.

Покрытие решений измеряет, насколько хорошо тесты охватывают все возможные пути, которые могут быть выбраны в результате решений, принятых в приложении. Для получения полного покрытия решений необходимо протестировать все возможные комбинации значений для всех логических операторов.

При тестировании покрытия условий и решений следует использовать специальные инструменты, помогающие оценить покрытие кода. Например, JaCoCo – это инструмент, который можно использовать для оценки покрытия кода в Java и Kotlin. Предоставляет подробную информацию о покрытии кода, включая покрытие условий и решений.

Для тестирования покрытия условий и решений важно разработать хорошие тестовые примеры, которые охватывают все возможные ветви логики приложения. Это может занять много времени и усилий, но позволит найти больше ошибок и улучшить качество программного обеспечения [61].

Тестирования покрытия условий и значений операторов (The value of statement and decision testing) – это метод тестирования программного обеспечения, используемый для проверки правильности работы выражений и условных операторов в коде. Этот метод позволяет определить, насколько хорошо тесты покрывают все возможные пути выполнения кода и какие условия и значения операторов не тестировались.

Для проверки условия и покрытия значения оператора используется несколько методов, таких как метод ветвей и границ, метод покрытия условий и решений и метод покрытия операторов.

Метод ветвей и границ используется для проверки всех возможных путей выполнения кода, включая все ветви и границы. Этот метод позволяет убедиться, что все ветви и границы кода выполняются корректно.

Метод покрытия условий и решений проверяет, что все возможные комбинации условий и решений в коде протестированы. Этот метод позволяет убедиться, что код работает корректно во всех возможных сценариях.

Метод покрытия значения оператора используется для проверки всех возможных значений, которые могут принимать операторы в коде. Этот метод позволяет убедиться, что код правильно обрабатывает все возможные значения операторов.

Тестирование методом серого ящика

Тестирование серого ящика – это особый метод тестирования с неполным знанием внутренней работы программного обеспечения. Для запуска таких тестов не нужен доступ к исходному коду программного обеспечения.

Все тесты основаны на простом знании алгоритмов, архитектур и других высокоуровневых характеристик поведения продукта.

Тестирование серого ящика включает в себя преимущества тестирования черного ящика и белого ящика.

Техники тестирования, основанные на опыте

Техника тестирования предугадывание ошибок (Error guessing) основана на интуиции и опыте тестировщика. Она заключается в поиске ошибок в программном продукте на основе предварительных знаний и опыта тестировщика. Это означает, что тестировщику необходимо иметь глубокое понимание тестируемой системы, ее потенциальных проблем и типичных ошибок, которые могут возникнуть в процессе эксплуатации системы.

Чтобы применить метод прогнозирования ошибок, тестировщик должен использовать свой профессиональный опыт и логику, чтобы угадать, где могут быть ошибки. Например, если приложение имеет функцию отправки сообщений, тестер может ожидать возникновения ошибок при попытке отправить пустое сообщение или сообщение с неправильным форматом.

Техника прогнозирования ошибок может быть особенно полезна в ситуациях, когда нет формальных требований или тестовых случаев, так как она позволяет тестировщику находить ошибки, которые могут быть упущены при тестировании формальными методами. Однако, поскольку он полагается на интуицию, он может не обнаруживать некоторые типы ошибок.

Методы прогнозирования ошибок можно использовать в сочетании с другими методами тестирования, такими как тестирование покрытия, краевое тестирование и регрессионное тестирование, чтобы обеспечить более полное и эффективное тестирование программного обеспечения.

Исследовательское тестирование (Exploratory testing) – это методология тестирования программного обеспечения, в которой тестировщики могут свободно выбирать, как они ищут ошибки и какие тестовые примеры они используют. Вместо заранее определенного плана тестирования, который строго следует набору инструкций, исследовательское тестирование позволяет тестировщикам использовать свой интуитивный подход и опыт для разработки тестовых случаев.

Исследовательское тестирование используется для обнаружения ошибок, которые могут быть упущены при традиционном тестировании. Этот метод может быть полезен в случаях, когда информация о тестируемом программном обеспечении ограничена или, когда программное обеспечение имеет высокий уровень сложности.

Процесс исследовательского тестирования начинается с изучения тестируемой системы и формирования гипотез о возможных ошибках. Затем тестировщики разрабатывают тестовые примеры, используя гипотезы в качестве ориентира. Тестировщики могут повторно протестировать отдельные элементы после обнаружения ошибок, чтобы убедиться, что они исправлены.

Исследовательское тестирование может быть эффективным методом обнаружения дефектов, но документирование процесса и результатов тестирования может быть затруднено. Поэтому очень важно, чтобы команда имела высокий уровень специализации [62].

Свободное тестирование (Ad-hoc testing) – это метод тестирования программного обеспечения, основанный на опыте и интуиции тестировщика. Он отличается от тестирования, основанного на заданных требованиях или тестовых примерах.

При исследовательском тестировании тестировщик изучает приложение изнутри и тестирует без заранее определенного плана. Он исследует функциональность, внешний вид и взаимодействие с программой и использует свой опыт и интуицию для поиска потенциальных проблем и ошибок.

Исследовательское тестирование можно использовать в дополнение к более традиционным методам тестирования для обнаружения новых проблем, которые могут быть упущены более формализованными методами тестирования. Однако обратная сторона этого метода может отрицательно сказаться на производительности тестировщиков, если тесты не управляются и не контролируются должным образом.

Тестирование по контрольным спискам (Checklist-based testing) – это один из методов тестирования программного обеспечения, основанный на предварительно составленном списке проверок, которые необходимо выполнить для определенного сценария тестирования. Контрольные списки могут состоять из различных критериев и категорий, которые можно использовать для тестирования на разных уровнях, таких как модульное, интеграционное и системное тестирование.

При тестировании по контрольному списку тестировщик запускает приложение и последовательно проверяет каждый элемент в контрольном списке, если он завершается успешно или обнаруживает ошибку. Некоторые контрольные списки могут также включать инструкции по тестированию, что делать в случае обнаружения ошибки и какова ожидаемая производительность приложения.

Преимущество тестирования с контрольным списком заключается в том, что оно позволяет проводить тестирование более предсказуемым, систематическим и эффективным способом, что может сократить время, затрачиваемое на тестирование. Тестирование по контрольному списку также может быть полезным для обеспечения уровня стандартизации тестирования и улучшения качества продукции.

Однако недостатком этого метода может быть то, что чек-листы могут быть недостаточно полными или не охватывать все возможные ошибки, что может привести к неполному тестированию продукта и пропуску некоторых критических ошибок [63].

Выбор метода тестирования

Для тестирования выбран метод черного ящика. Основная причина такого выбора заключается в том, что тестировщик может рассматривать систему с точки зрения пользователя. Тестирование методом черного ящика позволяет проверить соответствия системы требованиям и ожиданиям пользователя, а также для выявления ошибок.

Таким образом, тестирование черного ящика является важным инструментом для проверки функциональности и соответствия программного продукта, а также для обнаружения ошибок и проблем.

* 1. Проведение тестирования

Тестирование функциональности

Тестирование функциональности проводится для проверки того, что продукт соответствует требованиям и спецификациям, определенным на этапе анализа и проектирования. Это позволяет убедиться, что программа выполняет заданные функции и корректно обрабатывает данные.

Тестирование функциональности включает тестирование основных функций программы, таких как ввод данных, обработка данных, вывод данных и других функций, необходимых для выполнения основной задачи программы. В этом случае проводятся тесты с целью выявления ошибок, несоответствий требованиям и других проблем в программе.

Тестирование функциональности может включать создание тестовых сценариев, воспроизводящих различные сценарии использования программы и проверки ее функциональности.

Таким образом составлены тест-кейсы. Некоторые из них представлены в таблицах 3.3 – 3.7, как пример.

Таблица 3.3 – Тестирование функциональности авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Авторизация | | |
| Предварительные шаги | | Шаги | Ожидаемый результат |
| Открыть приложение | | Ввести логин  Ввести пароль  Нажать кнопку «Авторизоваться» | При вводе правильного логина и пароля переход в окно расписания.  При вводе не правильного логина или пароля сообщение об ошибке |

Тестирование прошло успешно. При вводе правильного логина и пароля переход в окно расписания. При вводе неправильного логина или пароля сообщение об ошибке.

Таблица 3.4 – Тестирование функциональности расписания

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Совпадение расписания по дням | | |
| Предварительные шаги | | Шаги | Ожидаемый результат |
| Открыть приложение  Авторизоваться | | Нажать в календаре на другие дни недели | Расписание на экране совпадает с фактическим расписанием |

Тестирование прошло успешно. Расписание на экране совпадает с фактическим расписанием.

Таблица 3.5 – Тестирование функциональности создания задачи

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Создание задачи | | |
| Предварительные шаги | | Шаги | Ожидаемый результат |
| Открыть приложение  Авторизоваться  Перейти на окно задач | | Нажать кнопку «Новая задача»  Ввести название задачи  Ввести текст задачи  Выбрать предмет  Выбрать срок окончания задачи | Задача появилась в списке текущих задач |

Тестирование прошло успешно. Задача появилась в списке текущих задач.

Таблица 3.6 – Тестирование функциональности поиска расписания группы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 |  | Поиск расписания группы | | |
| Предварительные шаги | | | Шаги | Ожидаемый результат |
| Открыть приложение  Авторизоваться  Перейти на окно поиска | | | Ввести группу «ПИН-45» в поле поиска  Нажать на нее в выпадающем списке | Расписание группы на экране совпадает с расписанием группы ПИН-45 |

Тестирование прошло успешно. Расписание группы на экране совпадает с расписанием введенной группы.

Таблица 3.7 – Тестирование функциональности поиска расписания преподавателя

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | Поиск расписания преподавателя | | |
| Предварительные шаги | | Шаги | Ожидаемый результат |
| Открыть приложение  Авторизоваться  Перейти на окно поиска  Перейти на поиск преподавателя | | Ввести преподавателя «Кононова» в поле поиска  Нажать на преподавателя в выпадающем списке | Расписание преподавателя на экране совпадает с его фактическим расписанием |

Тестирование прошло успешно. Расписание преподавателя на экране совпадает с его фактическим расписанием.

С помощью инструмента UI Automator проведены тесты, представленные выше (рисунок 3.3).

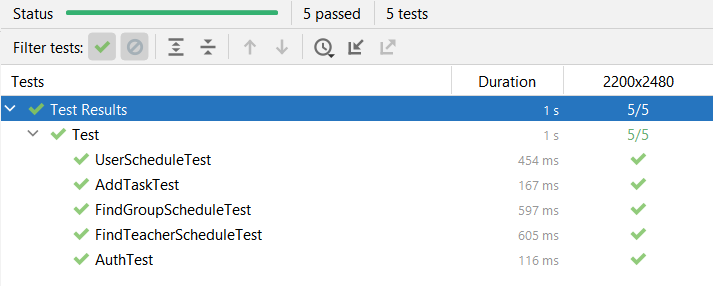


Рисунок 3.3 – Проведение тестирования

В результате тестирования ошибок не выявлено.

Тестирование используемости

Тестирование используемости проводятся для оценки удобства использования продукта. Оно позволяет определить, насколько легко пользователь может выполнять задачи с помощью продукта и насколько интуитивно понятен интерфейс.

Основная цель тестирования – улучшить пользовательский опыт и повысить удобство использования продукта. Процесс тестирования включает в себя наблюдение за тем, как пользователи взаимодействуют с продуктом, и сбор данных о том, как они воспринимают определенные аспекты продукта.

Для тестирования используемости расписания составлена таблица 3.8, для определения удобства пользования расписанием. Для этого, необходимо выполнить ряд действий:

1. Нажать на завтра в календаре.
2. Пролистать расписание.

Таблица 3.8 – Тестирование используемости расписания пользователя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Просмотр расписание на завтра | | | | |
| Польз. 1 | | Польз. 2 | Польз. 3 | Польз. 4 | Среднее время |
| 3.21 с | | 4.10 с | 2.89 с | 3.99 с |  |

Предположительное время прохождение теста 4 секунды. Тест пройден успешно. Интерфейс расписания интуитивно-понятен.

Для тестирования используемости поиска расписания группы составлена таблица 3.9, для определения удобства пользования поиском. Для этого, необходимо выполнить ряд действий:

1. Нажать на кнопку поиска.
2. Ввести в поле поиска группу «ПИН-45».
3. Нажать на группу в выпадающем меню.
4. Просмотреть расписание группы на сегодня.

Таблица 3.9 – Тестирование используемости поиска расписания группы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | Поиск расписания группы | | | | |
| Польз. 1 | | Польз. 2 | Польз. 3 | Польз. 4 | Среднее время |
| 9.65 с | | 11.73 с | 10.83 с | 11.29 с |  |

Предположительное время прохождение теста 11 секунд. Тест пройден успешно. Интерфейс поиска интуитивно-понятен.

Для тестирования используемости поиска расписания преподавателя составлена таблица 3.10, для определения удобства пользования поиском. Для этого, необходимо выполнить ряд действий:

1. Нажать на кнопку поиска.
2. Нажать на кнопку «Преподаватель».
3. Ввести в поле поиска группу «Кононова».
4. Нажать на преподавателя в выпадающем меню.
5. Просмотреть расписание преподавателя на сегодня.

Таблица 3.10 – Тестирование используемости поиска расписания преподавателя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | Поиск расписания преподавателя | | | | |
| Польз. 1 | | Польз. 2 | Польз. 3 | Польз. 4 | Среднее время |
| 14.96 с | | 13.52 с | 14.61 с | 12.88 с | 13.99 с |

Предположительное время прохождение теста 15 секунд. Тест пройден успешно. Интерфейс поиска интуитивно-понятен.

Для тестирования используемости просмотра задачи составлена таблица 3.11, для определения удобства пользования задачами. Для этого, необходимо выполнить ряд действий:

1. Нажать на кнопку задач.
2. Нажать на первую задачу.
3. Просмотреть задачу.

Таблица 3.11 – Тестирование используемости поиска расписания группы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Просмотр задачи | | | | |
| Польз. 1 | | Польз. 2 | Польз. 3 | Польз. 4 | Среднее время |
| 2.97 с | | 3.41 с | 3.11 с | 2.47 с | 2.99 с |

Предположительное время прохождение теста 3 секунды. Тест пройден успешно. Интерфейс задач интуитивно-понятен.

Тестирование совместимости

Тестирование совместимости необходимо провести для проверки того, как различные компоненты взаимодействуют друг с другом. Тестирование также включает в себя проверку правильности отображения компонентов на устройствах с разным разрешением и dpi, а также на платформах разных версий. Что помогает выявить недостатки в отображении пользовательского интерфейса.

В результате тестирования составлена таблица 3.12 для представления проведенных испытаний.

Таблица 3.12 – Результаты тестирования совместимости.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Экранная форма | Устройства | | | | | | | |
| Разрешение | dpi | Разрешение | dpi | Разрешение | dpi | Разрешение | dpi |
| 1080х2340 | 440 | 1440x2560 | 560 | 1080х2160 | 400 | 2200х2480 | 420 |
| Авторизация | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | |
| Расписание | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | |
| Поиск | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | |
| Задачи | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | |
| Добавление задачи | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | |
| Просмотр задачи | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | | Тестирование прошло успешно. Недостатков не выявлено. | |

В результате тестирование недостатков не выявлено.



Рисунок 3.4 – Экранная форма расписания при разрешении 1080х2340



Рисунок 3.5 – Экранная форма расписания при разрешении 1440x2560



Рисунок 3.6 – Экранная форма расписания при разрешении 1080х2160

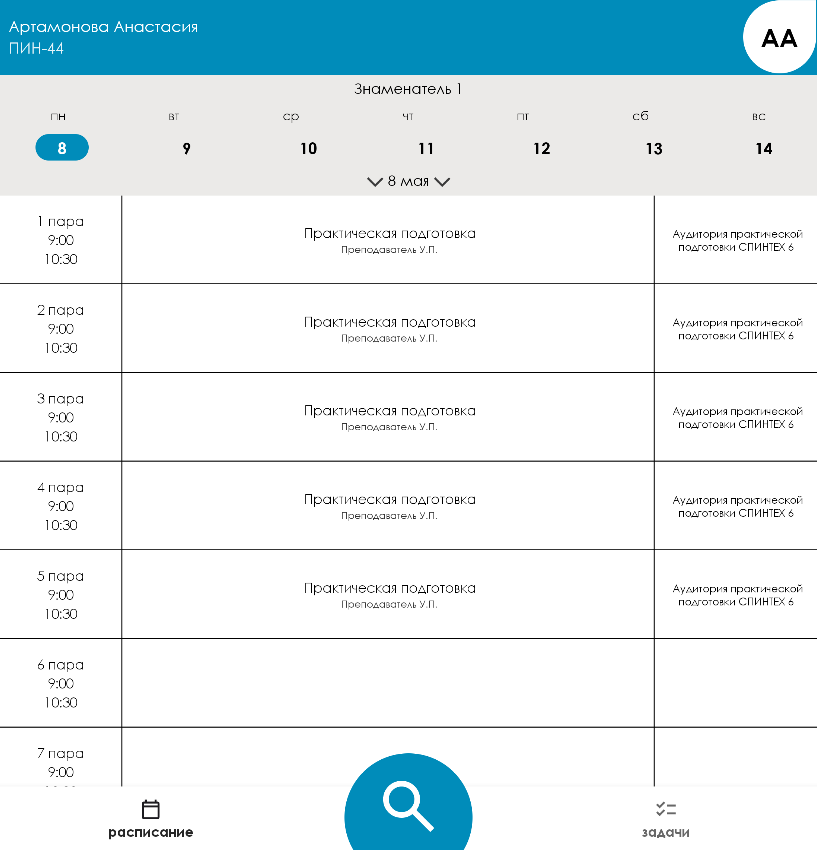


Рисунок 3.7 – Экранная форма расписания при разрешении 2200х2480

На рисунках 3.4 – 3.7 представлено, как выглядит окно расписания на разных устройствах.

Тестирование использования ресурсов

Тестирование использования ресурсов сделано с помощью профилировщика Android Studio (рисунок 3.8). Заметно, что больше всего память возросла на моменте поиска расписания групп и преподавателей. В остальное время приложение не требует значительных энерго- и ресурсо-затрат устройства.

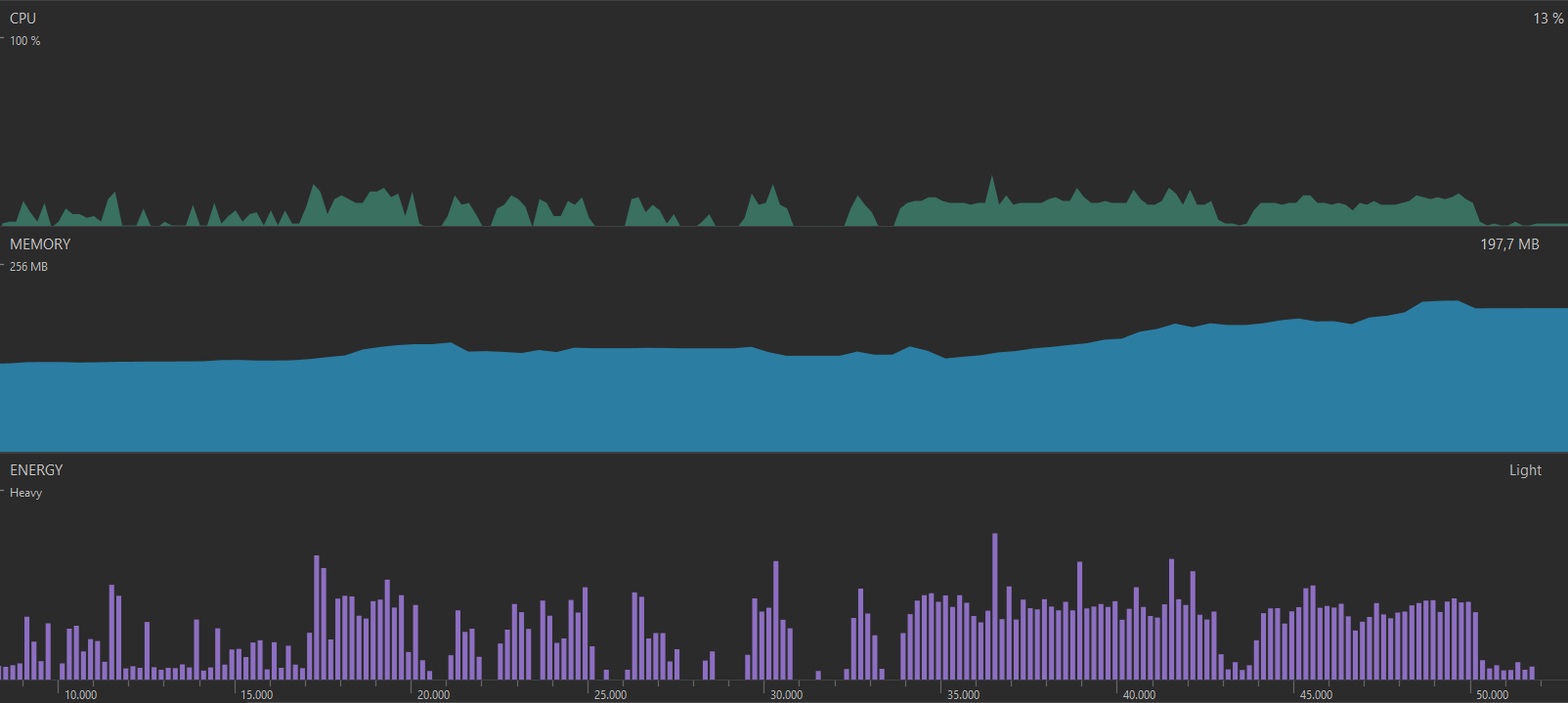


Рисунок 3.8 – Тестирование производительности

В процессе тестирования выявлено, что при пиковой нагрузки приложения затрачивается 13% ресурсов приложения. В техническом задании указано, что минимальные затраты не превышают 256 MB. По результатам тестирования выявлено, что приложению необходимо 197,7 MB. По затратам энергии тест показал «Light», что означает низкие затраты энергии.

Выводы по испытательному разделу

Отладка МП ПР проводилась средствами IDE Android Studio.

Тестирование приложения включало в себя функциональное тестирование, тестирование используемости, совместимости и использования ресурсов. Функциональное тестирование проводилось с помощью инструмента UI Automator. Тестирование производительности проводилось с помощью профилировщика Android Studio. Тестирование совместимости реализовывалось на платформах с ОС Android версии от 5.0 до 12.0, в том числе, на различных оболочках системы Android.

По результатам тестирования в МП ПР все ошибки выявлены и исправлены.

ЗАКЛЮЧЕНИЕ

В результате исследования предметной области и приложений-аналогов, принято решение о разработке собственного мобильного приложения, обладающего возможностью просмотра своего расписания, расписания других групп и преподавателей, а также поиска окна для переноса занятия и добавления задач. Сформулирован перечень требований к функционалу разрабатываемого приложения.

Мобильное приложения для просмотра расписания МИЭТ реализуется для мобильных устройств c операционной системой Android. Для облегчения этапа написания кода спроектированы Use case диаграмма, а также ИЛМ и ДЛМ. Разработаны схема данных, которая призвана облегчить понимание логических связей по данным между рабочими экранами приложения и схема алгоритма, на которой подробно описан процесс работы приложения.

В качестве языка реализации МП ПР выбран язык программирования Kotlin, поскольку он отвечает выдвинутым требованиям. Для разработки мобильного приложения выбрана среда разработки Android Studio.

В ходе реализации МП ПР изучены особенности разработки под операционную систему Android. Обозначены составные части нативного Android-приложения. Разработаны экранные формы пользовательского интерфейса МП ПР. Сгенерирована UML-диаграмма классов приложения, демонстрирующая программную реализацию поставленной задачи.

Отладка МП ПР проводилась средствами IDE Android Studio.

Тестирование приложения включало в себя функциональное тестирование, тестирование используемости, совместимости и производительности. Функциональное тестирование проводилось с помощью инструмента UI Automator. Тестирование производительности проводилось с помощью встроенного профилировщика Android Studio. Тестирование совместимости реализовывалось на платформах с ОС Android версии от 5.0 до 12.0, в том числе, на различных оболочках системы Android.

По результатам тестирования в МП ПР все ошибки выявлены и исправлены.

Разработанное мобильное приложение позволяет просмотреть расписание студенческих групп и преподавателей. Кроме того, МП ПР дает возможность поиска окна для переноса занятия. Также мобильное приложение позволяет добавлять задачи для упрощения учебного процесса.

# Список литературы

|  |  |
| --- | --- |
| [1] | А. А. Доронина, Р. А. Касимов и Е. Л. Федотова, Методические указания по  подготовке выпускной квалификационной работы по направлению подготовки  бакалавров 09.03.04 "Программная инженерия", Л. Г. Гагарина, Ред.,  Москва: МИЭТ, 2021. |
| [2] | ГОСТ 19.505-79. Руководство оператора. Требования к содержанию и оформлению. |
| [3] | ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению. |
| [4] | ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов,  программ, данных и систем. Условные обозначения и правила выполнения. |
| [5] | «Расписание,» [В Интернете]. Available: https://miet.ru/schedule.  [Дата обращения: 10 10 2022]. |
| [6] | «Приложение GooglePlay - "Orioks",» [В Интернете]. Available:  https://play.google.com/store/apps/details?id=ru.eva.miet.orioks.  [Дата обращения: 10 10 2022]. |
| [7] | «Приложение GooglePlay - "Расписание занятий - SKED",» [В Интернете]. Available:  https://play.google.com/store/apps/details?id=com.sked.core. [Дата обращения: 10 10 2022]. |
| [8] | «Приложение GooglePlay - "Кампус",» [В Интернете]. Available:  https://play.google.com/store/apps/details?id=ru.dewish.campus.  [Дата обращения: 10 10 2022]. |
| [9] | «Приложение GooglePlay - "СтудЖурнал",» [В Интернете]. Available:  https://play.google.com/store/apps/details?id=com.romansytnyk.studentstudio.  [Дата обращения: 10 10 2022]. |
| [10] | «Unified Modeling Language,» [В Интернете]. Available:  https://en.wikipedia.org/wiki/Unified\_Modeling\_Language. [Дата обращения: 15 10 2022]. |
| [11] | «ОС Android: статистика, оценка и перспективы для рекламодателей,» [В Интернете].  Available: https://www.byyd.me/ru/blog/2021/08/android/. [Дата обращения: 15 10 2022]. |
| [12] | «С++,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/C%2B%2B.  [Дата обращения: 15 10 2022]. |
| [13] | «Разработка кроссплатформенных мобильных приложений на языке C++,»  [В Интернете]. Available: https://learn.microsoft.com/ru-ru/cpp/cross-platform/visual-cpp  -for-cross-platform-mobile-development?view=msvc-170. [Дата обращения: 15 10 2022]. |
| [14] | «C Sharp,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/C\_Sharp.  [Дата обращения: 15 10 2022]. |
| [15] | «Подробно о Xamarin,» [В Интернете]. Available: https://habr.com/ru/post/188130/.  [Дата обращения: 15 10 2022]. |
| [16] | «Python,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Python.  [Дата обращения: 15 10 2022]. |
| [17] | «Мобильная разработка на Python: обзор двух фреймворков,» [В Интернете]. Available:  https://proglib.io/p/mobile-python. [Дата обращения: 15 10 2022]. |
| [18] | «Java,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Java.  [Дата обращения: 15 10 2022]. |
| [19] | «Kotlin,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Kotlin.  [Дата обращения: 15 10 2022]. |
| [20] | «Что такое Eclipse? Обзор Eclipse Java,» [В Интернете]. Available:  https://codernet.ru/articles/drugoe/chto\_takoe\_eclipse\_obzor\_eclipse\_java\_sreda\_razrabotki  \_“pod\_sebya”/. [Дата обращения: 1 4 2023]. |
| [21] | «kotlin | Eclipse Plugins, Bundles and Products,» [В Интернете]. Available:  https://marketplace.eclipse.org/content/enhanced-kotlin-eclipse. [Дата обращения: 1 4 2023]. |
| [22] | «Kotlin,» [В Интернете]. Available:  https://marketplace.visualstudio.com/items?itemName=fwcd.kotlin.  [Дата обращения: 1 4 2023]. |
| [23] | «IDEs for Kotlin development,» [В Интернете]. Available:  https://kotlinlang.org/docs/kotlin-ide.html. [Дата обращения: 20 10 2022]. |
| [24] | «Eclipse (среда разработки) - Википедия,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Eclipse\_(среда\_разработки)#:~:text=Eclipse%20(МФА%3A%  20%5Bɪˈklɪps%5D,Развивается%20и%20поддерживается%20Eclipse%20Foundation..  [Дата обращения: 1 4 2023]. |
| [25] | «Visual Studio Code: что это за редактор и для чего он,» [В Интернете]. Available:  https://blog.skillfactory.ru/glossary/visual-studio-code/. [Дата обращения: 1 4 2023]. |
| [26] | «Atom,» [В Интернете]. Available: https://en.wikipedia.org/wiki/Atom\_(text\_editor).  [Дата обращения: 9 5 2023]. |
| [27] | «Vim,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Vim.  [Дата обращения: 9 5 2023]. |
| [28] | «Meet Android Studio,» [В Интернете]. Available:  https://developer.android.com/studio/intro. [Дата обращения: 20 10 2022]. |
| [29] | «Jetpack Compose | Row,» [В Интернете]. Available:  https://metanit.com/kotlin/jetpack/2.3.php. [Дата обращения: 10 2 2023]. |
| [30] | «Jetpack Compose | Column,» [В Интернете]. Available:  https://metanit.com/kotlin/jetpack/2.2.php. [Дата обращения: 10 2 2023]. |
| [31] | «Jetpack Compose | Box,» [В Интернете]. Available:  https://metanit.com/kotlin/jetpack/2.1.php. [Дата обращения: 10 2 2023]. |
| [32] | «List of Compose modifiers | Android Developers,» [В Интернете]. Available:  https://developer.android.com/jetpack/compose/modifiers-list. [Дата обращения: 10 2 2023]. |
| [33] | «Navigating with Compose | Android Developers,» [В Интернете]. Available:  https://developer.android.com/jetpack/compose/navigation. [Дата обращения: 12 2 2023]. |
| [34] | «Навигация на Jetpack Compose by Google,» [В Интернете]. Available:  https://habr.com/ru/post/594621/. [Дата обращения: 12 2 2023]. |
| [35] | «MVC - MDN Web Docs Glossary: Definitions of Web-related,» [В Интернете]. Available:  https://developer.mozilla.org/en-US/docs/Glossary/MVC. [Дата обращения: 5 4 2023]. |
| [36] | «Построение диаграммы классов,» [В Интернете]. Available:  https://flexberry.github.io/ru/gpg\_class-diagram.html. [Дата обращения: 5 4 2023]. |
| [37] | «Создание и вызов Activity - Start Android,» [В Интернете]. Available:  https://startandroid.ru/ru/uroki/vse-uroki-spiskom/58-urok-21-sozdanie-i-vyzov-activity.html#:~:text=MAIN%20показывает%20системе%2C%20что%20Activity,intent..  [Дата обращения: 5 4 2023]. |
| [38] | «Что такое отладка?,» [В Интернете]. Available:  https://aws.amazon.com/ru/what-is/debugging/. [Дата обращения: 7 4 2023]. |
| [39] | «Большой гайд по тестированию Android-приложений,» [В Интернете]. Available:  https://testengineer.ru/bolshoj-gajd-po-testirovaniyu-android-prilozhenij/.  [Дата обращения: 7 4 2023]. |
| [40] | «Виды Тестирования ПО - Тестирование безопасности,» [В Интернете]. Available:  http://www.protesting.ru/testing/types/security.html. [Дата обращения: 7 4 2023]. |
| [41] | «Инструменты разработчика для тестирования Android-приложений,» [В Интернете].  Available: https://habr.com/ru/companies/simbirsoft/articles/592595/.  [Дата обращения: 11 4 2023]. |
| [42] | «Appium,» [В Интернете]. Available: https://appium.io. [Дата обращения: 7 4 2023]. |
| [43] | «Selendroid,» [В Интернете]. Available: http://selendroid.io. [Дата обращения: 7 4 2023]. |
| [44] | «Espresso,» [В Интернете]. Available: https://developer.android.com/training/testing/espresso.  [Дата обращения: 7 4 2023]. |
| [45] | «Robotium,» [В Интернете]. Available: https://github.com/RobotiumTech/robotium.  [Дата обращения: 7 4 2023]. |
| [46] | «Robolectric,» [В Интернете]. Available: https://robolectric.org.  [Дата обращения: 26 4 2023]. |
| [47] | «kotlinx-coroutines-test,» [В Интернете]. Available:  https://kotlinlang.org/api/kotlinx.coroutines/kotlinx-coroutines-test/.  [Дата обращения: 26 4 2023]. |
| [48] | «turbine,» [В Интернете]. Available: https://github.com/cashapp/turbine.  [Дата обращения: 26 4 2023]. |
| [49] | «JUnit | Java: Автоматическое тестирование - Хекслет,» [В Интернете]. Available:  https://ru.hexlet.io/courses/java-testing/lessons/junit/theory\_unit#:~:text=Кроме%20того  %2C%20JUnit%20обладает%20целым,создание%20динамических%20тестов.  [Дата обращения: 26 4 2023]. |
| [50] | «Write automated tests with UI Automator,» [В Интернете]. Available:  https://developer.android.com/training/testing/other-components/ui-automator.  [Дата обращения: 26 4 2023]. |
| [51] | «Тестирование методом эквивалентного разбиения,» [В Интернете]. Available:  https://www.it-courses.by/equivalence-partitioning/#:~:text=Эквивалентное%20разбиение  %20–%20это%20метод%20тестирования,одинаковый%20результат%20при%20  тестировании%20программы.. [Дата обращения: 21 4 2023]. |
| [52] | «Техника анализа граничных значений,» [В Интернете]. Available:  https://qaevolution.ru/testovaya-dokumentaciya/test-dizajn/texnika-analiza-granichnyx-  znachenij/. [Дата обращения: 21 4 2023]. |
| [53] | «Что такое доменное тестирование?,» [В Интернете]. Available:  https://sedtest-school.ru/test-design/chto-takoe-domennoe-testirovanie/.  [Дата обращения: 21 4 2023]. |
| [54] | «Попарное тестирование: суть техники, инструменты,» [В Интернете]. Available:  https://habr.com/ru/companies/otus/articles/592575/. [Дата обращения: 21 4 2023]. |
| [55] | «Таблица принятия решений,» [В Интернете]. Available:  https://qaevolution.ru/testovaya-dokumentaciya/test-dizajn/tablica-prinyatiya-reshenij/.  [Дата обращения: 21 4 2023]. |
| [56] | «Тестирование таблицы переходов состояний.,» [В Интернете]. Available:  https://www.it-courses.by/state-transition-testing/. [Дата обращения: 21 4 2023]. |
| [57] | «Особенности тестирования программного обеспечения,» [В Интернете]. Available:  https://moluch.ru/archive/107/25816/. [Дата обращения: 21 4 2023]. |
| [58] | «Тестирование вариантов использования,» [В Интернете]. Available:  https://pandia.ru/text/78/555/69492.php. [Дата обращения: 21 4 2023]. |
| [59] | «Виды , уровни, методы и техники тестирования,» [В Интернете]. Available:  https://crashtest.by/test-types-methods-levels-techniques/. [Дата обращения: 21 4 2023]. |
| [60] | «Метод покрытия операторов,» [В Интернете]. Available: https://tstu.ru/book/elib3/mm/2015/loskutov/pages/4\_3\_1\_metod\_pokrytija\_operatorov.html.  [Дата обращения: 21 4 2023]. |
| [61] | «Тестирование покрытия решений,» [В Интернете]. Available:  https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/slovar-  testirovaniia-programmnogo-obespecheniia/testirovanie-pokrytiia-reshenii#:~:text=Покрытие  %20решения%20или%20покрытие%20Ветвления,обеспечение%20выполнения%20  всего%20достижимого%. [Дата обращения: 21 4 2023]. |
| [62] | «Исследовательское тестирование: когда его стоит применять и как это делать,» [В Интернете]. Available: https://habr.com/ru/companies/redmadrobot/articles/280618/. [Дата обращения: 22 4 2023]. |
| [63] | «Чеклист для тестирования прототипа: 5 пунктов, которые нужно учесть, чтобы не потратить ресурсы впустую,» [В Интернете]. Available: https://vc.ru/design/613650-cheklist-dlya-testirovaniya-prototipa-5-punktov-kotorye-nuzhno-uchest-chtoby-ne-potratit-resursy-vpustuyu. [Дата обращения: 22 4 2023]. |