Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий
Кафедра Компьютерные интеллектуальные технологии

«Допустить к защите»
Зав. каф. КИТ, к.т.н.
_____ А.В. Речинский
«____» _____ 20___ г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

СОЗДАНИЕ ПРОТОТИПА СИСТЕМЫ ДЛЯ ТЕСТИРОВАНИЯ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ ПРИ ПОМОЩИ СТАТИСТИЧЕСКИХ МЕТРИК

направление: 02.03.03 «Математическое обеспечение и администрирование информационных систем»

Выполнила:
Кузнецова Анастасия Сергеевна
Подпись_____

Руководитель:
доцент каф. КИТ ИКНТ, к.ф.-м.н.,
Черкасова Танзиля Халитовна
Подпись_____

Санкт-Петербург
2016

РЕФЕРАТ

«Создание прототипа системы для тестирования интерфейса пользовательских приложений при помощи статистических метрик»

Работа содержит: 36 стр., 4 ил., 5 табл., 16 библ.

Ключевые слова: пользовательский интерфейс, оценка качества, графический интерфейс, разработка для Android.

Тема данной работы относится к области разработки и проектирования пользовательского интерфейса. В работе создается приложение для смартфонов на платформе Android, используемое для оценки интерфейса других приложений. Данная система может быть полезна для разработчиков.

Логическая структура работы содержит введение, четыре главы, заключение и приложения.

Во введении раскрыта актуальность и практическая значимость работы.

В первой главе рассмотрены существующие современные методы оценки интерфейса и проведен их сравнительный анализ, а также сформулирована цель работы.

Вторая глава содержит постановку задач для выполнения работы, предъявленные к ней требования и обзор технологий для реализации.

В третьей главе описана разработка структурных единиц приложения.

В четвертой главе приведены результаты тестирования интерфейса при помощи разработанного приложения.

Заключение содержит описание задач, выполненных в ходе работы.

В приложениях содержится исходный код всех структурных единиц системы.

«Creating a prototype system for testing the application's user interface based on statistical metrics»

This thesis includes: 36 pages, 4 figures, 5 tables, 16 references.
Keywords: user interface, quality control, application interface, graphical interface, Android development.

The theme of this thesis relates to the field of development application’s user interface. An application for smartphones, used to evaluate the interface of other applications, is created in this work. This system can be useful for developers.

The logical structure of the work contains an introduction, four chapters, conclusion and annexes.

The relevance and practical significance of the work are disclosed in the introduction.

The first chapter describes the existing methods of interface evaluation and comparative analysis was carried out, as well as the purpose of the work is formulated.

The second chapter contains the statement of objectives for the work, presented requirements and an overview of technologies for implementation.

The third chapter describes the development of the structural units of the application.

The fourth chapter presents the results of testing the interface by means of the developed application.

Conclusion contains a description of the tasks performed during operation.

The annexes provide the source code of all structural units of the system.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В данной пояснительной записке к выпускной квалификационной работе применяют следующие термины с соответствующими определениями:

- Интерфейс – имеется в виду пользовательский интерфейс, то есть совокупность программных средств, с помощью которых человек может взаимодействовать с программой;
- Среда разработки – система программных средств, которая используется программистами для разработки программного обеспечения;
- База данных – это информационная модель, которая позволяет хранить данные об объектах в упорядоченном виде;
- Библиотека – совокупность программных объектов, используемых для разработки программного обеспечения;
- Activity – аналог окон Windows в операционной системе Android.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

SUMI – Software Usability Measurement Inventory.

GOMS – Goals, Operators, Methods, and Selection rules.

KLM – Keystroke-Level Model.

ПО – программное обеспечение.

ГОСТ – государственный стандарт.

ВВЕДЕНИЕ

В современном мире высокоразвитых технологий и их лёгкой доступности благодаря сети Интернет практически любой желающий может заняться разработкой программного обеспечения. Естественно, у программы должен быть интерфейс для взаимодействия с пользователем. Но, к сожалению, далеко не каждый начинающий разработчик способен адекватно оценить, насколько эргономична его программа. Порой и опытные программисты при создании интерфейса могут допустить ошибки, усложняющие работу пользователей.

Непродуманный интерфейс способен испортить впечатление от программного продукта, а в некоторых случаях даже вызвать желание отказаться от его использования. Крайне важно подходить к разработке интерфейса ответственно и стараться своевременно обнаружить его недостатки.

При этом оценка качества интерфейса является достаточно сложной задачей, ведь понятие качества и удобства использования весьма субъективно.

Именно поэтому целью моей работы стало создание системы для оценки пользовательского интерфейса на основе временных измерений производимых пользователем операций. Поскольку некоторые из уже существующих методик оценивания рассчитаны на работу исключительно с персональными компьютерами, мною было принято решение создать систему для определения качества интерфейса приложений на смартфонах. Эта система позволит дать объективную оценку интерфейсу приложения, базирующуюся на статистике его использования.

1 Анализ предметной области

1.1 Определение качества интерфейса

Вследствие субъективности представлений об удобстве и простоте невозможно дать общее и однозначное определение хорошего интерфейса. Тем не менее, можно сделать это для конкретного программного продукта [1]. В каждом случае необходимо определить наиболее значимые качества интерфейса, исходя из особенностей целевой аудитории и основных задач программы, а на основе этих качеств уже формировать требования к интерфейсу. Так, например, приложение, рассчитанное на использование пожилыми людьми, должно иметь достаточно крупный шрифт, что не является критичным качеством для приложений, используемых молодёжью.

Однако всё же есть некоторые общие системы показателей качества интерфейса. При выполнении данной работы я опиралась на систему показателей Шнейдермана [2]. Согласно ей, интерфейс характеризуют:

а) скорость работы пользователя;

б) количество человеческих ошибок;

в) субъективная удовлетворённость пользователя;

г) скорость обучения работе с интерфейсом.

Эту модель сложно назвать полной, однако в ней сформулированы достаточно точные характеристики, не оперирующие такими расплывчатыми понятиями, как «удобство».

1.2 Обзор существующих методик оценки интерфейса

В настоящее время уже существуют различные методики определения качества интерфейса. Я приведу в качестве примера несколько наиболее распространённых, разделив их по критерию участия в оценке пользователей.

а) методики, требующие участия пользователей;
    1) анкетирование пользователей;
    2) тестирование с участием фокус-групп;
б) методики, не требующие участия пользователей;
    1) расчет среднего времени выполнения операции;
    2) оценка эксперта.

Далее я рассмотрю подробнее каждую из этих методик.

1.2.1 Анкетирование пользователей

Группа пользователей программного продукта должна ответить на вопросы, касающиеся его использования. Список вопросов может быть составлен самостоятельно, но также можно использовать готовую анкету, например SUMI – Software Usability Measurement Inventory [3].

Недостатком использования этой методики является то, что в результате будет получено лишь субъективное мнение пользователей, без учёта временной статистики выполнения операций в приложении. К тому же, многие международно одобренные анкеты-опросники содержат большое число вопросов (к примеру, упомянутая выше SUMI состоит из 50 пунктов). Это делает обработку результатов достаточно трудоёмкой, а также создаёт неудобство для респондентов.

1.2.2 Тестирование с участием фокус-групп

Данный метод подразумевает оценивание некоторых характеристик произведения операций пользователем в тестируемом программном продукте. Этими характеристиками могут быть: время выполнения операции, количество пользовательских ошибок, среднее время перехода к следующему пункту меню и т.д.

Благодаря этой методике можно получить объективные данные об эффективности интерфейса на основе его сравнения с другими версиями. Для более углублённой оценки фокус-группы могут быть разделены по возрасту или уровню технической подготовки. Это даст понимание о широте спектра потенциальных пользователей программного продукта.

1.2.3 Расчет среднего времени выполнения операции

Основная идея этого метода заключается в том, чтобы произвести расчет времени выполнения операции на основе средних значений времени выполнения её составляющих. Существует несколько моделей, реализующих эту методику. Самая известная их них – GOMS (Goals, Operators, Methods, and Selection rules), разработанная тремя американскими учеными в 1983 году [4]. Эта модель достаточно сложна и лишь косвенно касается моей темы, поэтому я рассмотрю её упрощенную вариацию – KLM (Keystroke-Level Model) [4].

Суть этой техники в разбиении операции на атомарные действия и вычислении суммы времени их выполнения. Среднее время выполнения этих действий было выведено в ходе исследований.

Рассмотрим простой пример. Допустим, пользователю необходимо ввести 4 символа в диалоговое окно и нажать кнопку ОК. Время, которое займет эта операция, рассчитывается по формуле (1.1):

$$\mathbf{4*К + Н + П + Л}, \quad (1.1)$$

где **К** = 0,2 с – время нажатия клавиши клавиатуры,

П = 1,1 с – время перемещения указателя мыши,

Н = 0,4 с – время перемещения руки с клавиатуры на мышь,

Л = 0,1 с – время, затрачиваемое на клик левой кнопкой мыши.

Таким образом, согласно технике KLM, данная операция займёт 2,4 с.

Главным достоинством этой методики является то, что её можно применять ещё на стадии планирования интерфейса, не затрачивая время на программную разработку нескольких вариантов. Также использование этой техники не требует участия третьих лиц (экспертов или пользователей). Но следует учитывать тот факт, что полученные результаты будут носить чисто теоретический характер. Поэтому, на мой взгляд, оптимальным решением будет использование этой методики в совокупности с тестированием конечного продукта.

1.2.4 Оценка эксперта

Существует множество правил, стандартов и рекомендаций по разработке интерфейса программного обеспечения. Например ГОСТ Р ИСО/МЭК 9126-93 [5] позволяет оценить качество ПО в целом, в том числе и удобство его использования.

Эксперт производит оценку, опираясь на эти стандарты и некоторые типовые решения. Он должен определить наиболее важные правила для конкретного приложения и оценить степень их соблюдения.

Данный метод в большей мере полагается на компетентность эксперта в области информационных технологий, поэтому необходимо тщательно выбирать профессионала.

1.3 Сравнение существующих методик оценки

В своем кратком обзоре я постаралась указать все существенные достоинства и недостатки каждого из четырёх рассматриваемых методов оценки интерфейса.

Далее мною были выбраны критерии для сравнения этих методов и проведён их анализ.

1.3.1 Критерии сравнения методик

Для сравнения я выбрала следующие критерии:

- объективность оценки;
- необходимость участия третьих лиц в оценке;
- оценка на основе реального использования программного продукта;
- возможность применения на стадии планирования;
- учёт показателей Шнейдермана при оценке.

Объективность оценки – критерий, определяющий, насколько независима методика от субъективного мнения человека (пользователя или эксперта).

Необходимость участия третьих лиц в оценке может повлиять на выбор методики, так как требует вовлечения дополнительных ресурсов, в том числе и материальных.

Оценка на основе реального использования программного продукта важна, поскольку она даёт нам не только теоретические сведения об использовании интерфейса, но и фактические результаты.

Возможность применения на стадии планирования позволяет прогнозировать успешность интерфейса ещё до его реализации.

Учёт показателей Шнейдермана при оценке – это один из основных критериев сравнения, так как эти показатели, на мой взгляд, дают наиболее точную характеристику интерфейса.

Заявленные значения выбранных критериев приведены в таблице 1.1.

Таблица 1.1 – Необходимые значения выбранных критериев

Критерий	**Заявленное значение**
Объективность оценки	Высокая
Участие третьих лиц в оценке	Не требуется
Оценка на основе реального использования	Да
Возможность применения на стадии планирования	Опционально
Учет показателей Шнейдермана	Минимум 2 из 4

1.3.2 Сравнение методик с учетом выбранных критериев

В соответствии с выбранными критериями я произвела сравнительный анализ существующих методик оценки интерфейса. Результаты этого анализа предоставлены в таблице 1.2.

Таблица 1.2 – Сравнение методик оценки интерфейса

	Анкетирование пользователей	**Тестирование с участием фокус-групп**	**Расчет времени выполнения операции**	**Оценка эксперта**
Объективность оценки	Средняя	Высокая	Высокая	Средняя
Участие третьих лиц в оценке	Требует участия пользователей	Требует участия пользователей	Не требуется	Требует участия эксперта
Оценка на основе реального использования	Да	Да	Нет	Нет
Возможность применения на стадии планирования	Отсутствует	Отсутствует	Есть	Отсутствует
Учет показателей Шнейдермана	1 из 4 (в)	3 из 4 (а, б, г)	1 из 4 (а)	0 из 4

Как видно из таблицы, наиболее соответствующая заданным критериям методика – тестирование с участием фокус-групп. Именно она будет реализована программно в этой работе.

1.4 Постановка цели

Итак, цель данного проекта – создание прототипа системы для тестирования пользовательского интерфейса мобильных приложений с участием фокус-групп. Инструментом оценивания эффективности является статистика временных показаний производимых операций.

2 Теоретическая часть

2.1 Постановка задач

Для выполнения поставленной цели необходимо выполнить следующие задачи:

- определение требований к системе;
- выбор среды разработки;
- создание алгоритма функционирования;
- исследование механизмов, позволяющих реализовать необходимый функционал;
- проектирование и разработка системы;
- тестирование системы.

2.2 Требования к разрабатываемой системе

Разрабатываемая система создаётся для того, чтобы оценивать скорость выполнения операций в приложении, сохранять результаты и предоставлять их разработчику.

Система должна быть универсальной. Иными словами, она способна произвести оценку вне зависимости от функциональности предоставленного ей приложения.

В системе должно быть хранилище результатов тестирования для их дальнейшего анализа.

Должен быть реализован вывод требуемых разработчику результатов на экран.

Данная система является прототипом, поэтому к ней предъявляются упрощенные требования с точки зрения дизайна.

2.3 Выбор среды разработки

Наиболее популярными платформами для смартфонов на данный момент являются Android от компании Google и iOS от Apple.

Операционная система Android лидирует как на мировом, так и на российском рынке смартфонов [6]. Однако целевая аудитория моего проекта – разработчики, а не конечные пользователи смартфонов.

Мною было изучено множество интернет-ресурсов, посвященных разработке приложений для этих систем. Авторы наиболее полных и развернутых сравнительных анализов в большинстве случаев утверждают, что для разработчиков более благоприятной средой является Xcode от Apple [7].

Однако решающим аспектом моего выбора стало отсутствие техники компании Apple, поскольку разработка в Xcode под Windows возможна только посредством виртуальной машины, что значительно усложняет процесс [8]. Также тестирование удобнее и быстрее проводить не на эмуляторе, а на реальном устройстве [9].

Поэтому, при имеющемся в моем распоряжении оборудовании, я приняла решение писать приложение для платформы Android. Наиболее приспособленной для этой цели мне показалась официальная интегрированная среда разработки Android Studio [10].

2.4 Разработка алгоритма функционирования

Разрабатываемое мною приложение должно по запросу пользователя начинать работу в фоновом режиме и считывать прикосновения к экрану устройства. Время прикосновения нужно записывать в таблицу базы данных.

Основываясь на записанных в базу данных, приложение вычисляет необходимые характеристики, такие как общее время совершения операции и среднее время между прикосновениями.

Пользователь должен иметь возможность просмотреть на экране полученные результаты тестирования.

Схематически этот алгоритм изображен на рисунке 2.1.

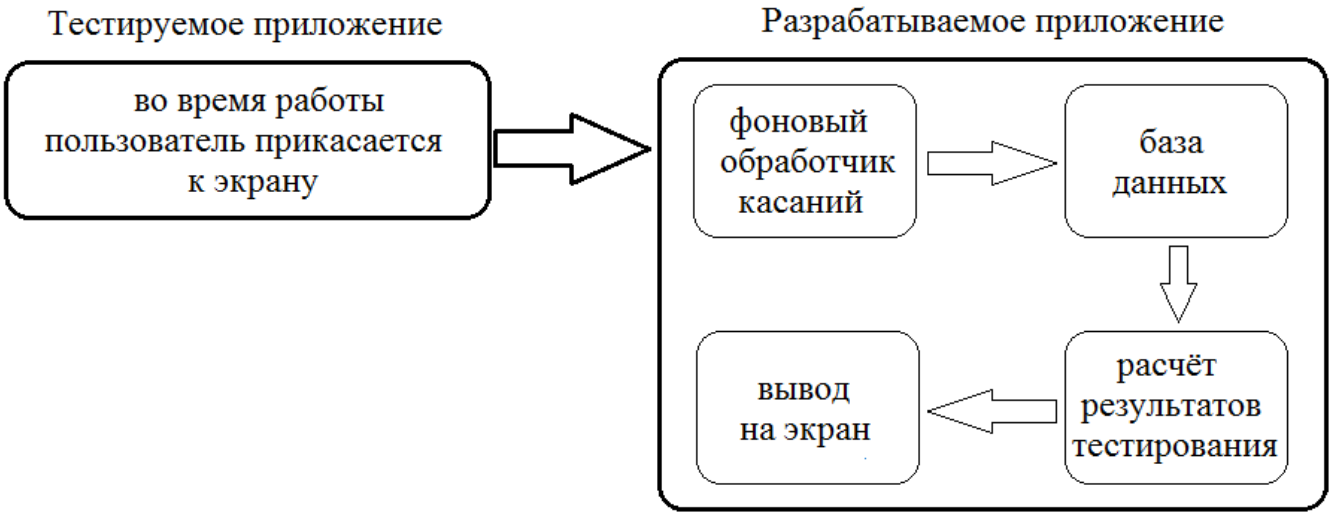

Рисунок 2.1 – Схема работы создаваемого приложения

2.5 Исследование механизмов, необходимых для реализации приложения

Для разработки приложений в Android Studio используется язык Java [11]. Изучение данного языка программирования (как и среды разработки) не было включено в курс дисциплин нашей специальности, поэтому я выделю некоторые специфические темы, которые необходимо изучить для создания проектируемого приложения:

- работа с базами данных для хранения и обработки результатов тестирования;
- работа приложения в фоновом режиме для отслеживания операций, производимых с тестируемым приложением.

Рассмотрим механизмы среды разработки, позволяющие осуществить работу желаемого функционала.

2.5.1 Библиотека SQLite

Для работы с базами данных в Android Studio используется библиотека SQLite. Это компактная встраиваемая база данных, особенностью которой является то, что она не имеет своего серверного процесса (в отличие от многих других баз данных) [12]. SQLite представляет собой библиотеку, что позволяет включать её в код программы, в результате чего приложение и база данных являют собой единое целое [13].

2.5.2 Сервисы Android

Работа в фоновом режиме осуществляется в Android при помощи так называемых сервисов (в некоторых переводах – служб) [14]. В разрабатываемом мною приложении необходимо использовать сервис для того, чтобы отслеживать прикосновения пользователя к экрану во время тестирования. Иными словами, сервис будет осуществлять свою работу даже в то время, когда вызвавшее его приложение находится в свёрнутом или закрытом состоянии.

3 Проектирование и реализация приложения

3.1 Разработка интерфейса приложения

Здесь я опишу создание файла main_activity.xml, отвечающего за внешний вид приложения.

3.1.1 Структура Android приложения

Для начала рассмотрим структуру приложения под Android. Приложение состоит из нескольких окон, называемых Activity. В определенный момент времени на экране отображается только одно Activity, но пользователь может переключаться между ними. Activity содержит в себе представления (View), являющиеся конкретными элементами на экране, например: текстовое поле, кнопка, картинка, чек-бокс и т.д. Графически эта структура изображена на рисунке 3.1.

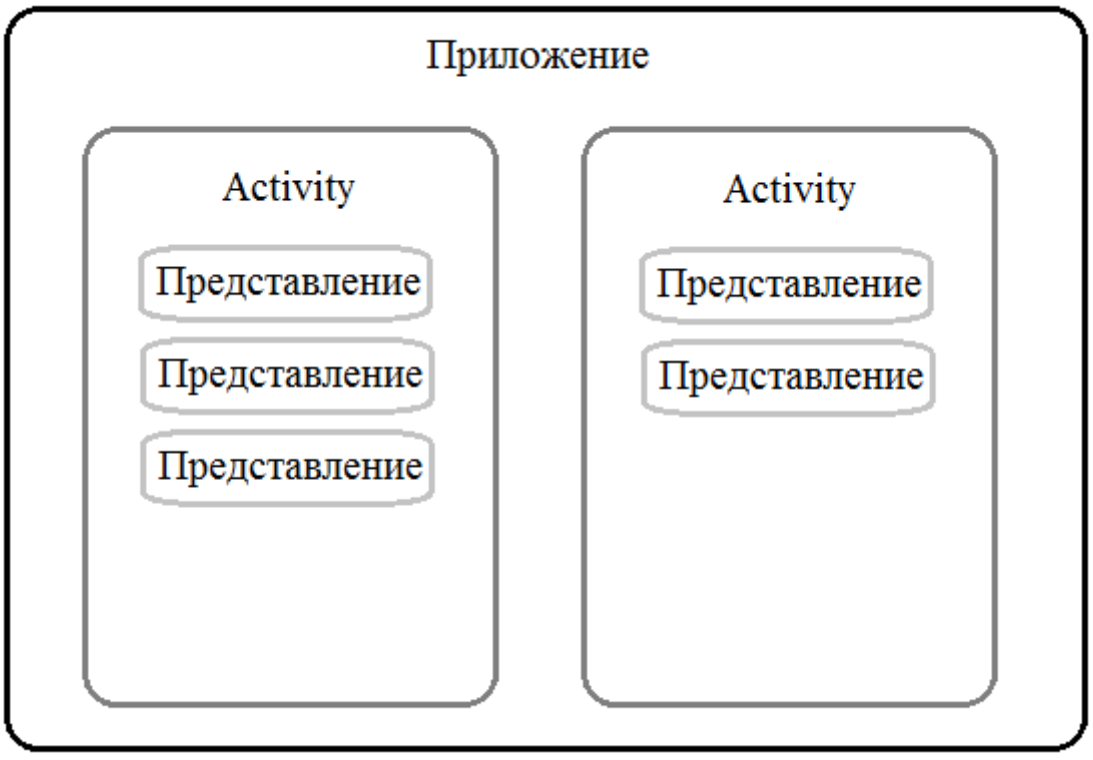

Рисунок 3.1 – Структура Android приложения

3.1.2 Создание интерфейса для Activity

Поскольку разрабатываемое приложение является прототипом, требования к дизайну интерфейса предъявляются минимальные, то есть нужно реализовать лишь самые необходимые для функционирования элементы.

Для корректной работы приложения нужны кнопки запуска и остановки тестирования, а также кнопка и поле для вывода результатов на экран. Исходный код, реализующий вышеперечисленные элементы, представлен в приложении 1.

На рисунке 3.1 изображен созданный интерфейс. Он содержит одно Activity и пять представлений: текстовое поле для ввода названия тестируемого приложения, три кнопки и текстовое поле для вывода результатов.

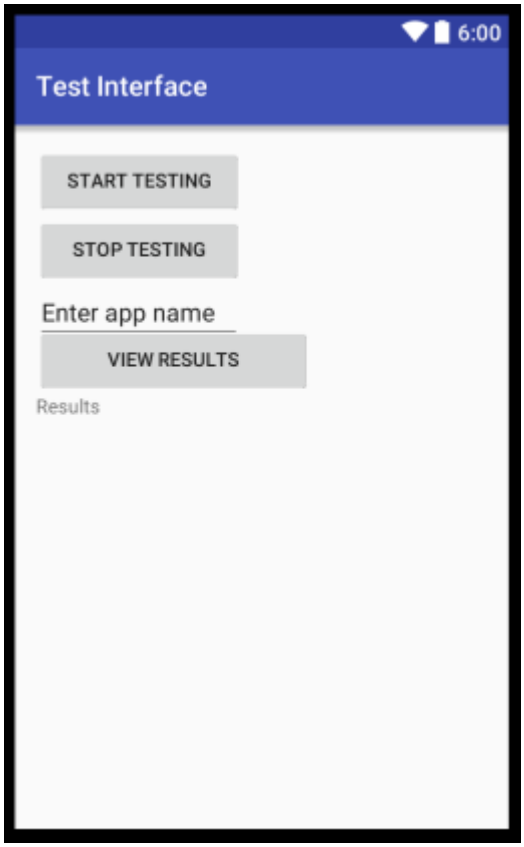

Рисунок 3.2 – Главный экран приложения

3.2 Разработка Activity приложения

В данном разделе я опишу создание файла MainActivity.java, который реализует функционал работы приложения. Исходный код файла приведен в приложении 2.

3.2.1 Создание базы данных

База данных состоит из двух таблиц: одна используется непосредственно во время тестирования приложения, вторая – для вывода результатов (таблица 3.1).

Таблица 3.1 – Структура таблиц базы данных

Таблица	**Назначение**	**Список полей**
Test	Таблица для записи временных показаний при тестировании	Clickno (int) – номер касания Clicktime (int) – время касания
Results	Таблица, содержащая результаты тестирования	App (text) – название тестируемого приложения Totaltime (int) – общее время выполнения операции Avetime (real) – среднее время между касаниями

3.2.2 Создание обработчиков нажатия кнопок

При нажатии кнопок «Start testing» и «Stop testing» соответственно запускается или останавливается сервис, производящий считывание касаний экрана.

После остановки сервис возвращает собранные данные в Activity. Для их просмотра нужно нажать кнопку «My results». Обработчик нажатия этой кнопки создает запрос к данным таблицы test, извлекает необходимые значения и помещает их в курсор. Далее с помощью этого курсора в цикле собираются данные, которые

затем выводятся на экран в поле для вывода. Также полученные результаты заносятся в таблицу results.

3.3 Разработка сервиса GlobalTouch

Создание сервиса глобальной обработки всех касаний экрана оказалось весьма нетривиальной задачей.

Сложность заключается в том, что стандартные средства Android для отслеживания касаний представляют собой два метода: onTouch() и onTouchEvent() [15]. Оба этих метода получают события только от представлений, на которые назначен обработчик. Но целью разрабатываемого сервиса является отслеживание всех касаний, вне зависимости от кода тестируемого продукта.

Решением этой проблемы стало создание представления, лежащего поверх всех остальных окон. Именно к этому представлению и привязан обработчик касаний onTouch(). При этом, представление не должно блокировать работу лежащих под ним приложений, поэтому оно представляет собой очень маленький (1x1 пиксель) макет, расположенный в левом верхнем углу экрана.

Обработчик касаний onTouch() записывает время касания в таблицу test базы данных.

Также при запуске и остановке сервиса вызываются соответственно методы onCreate() и onDestroy(), создающие уведомления для пользователя о том, что тестирование начато или окончено.

Исходный код файла GlobalTouch.java представлен в приложении 3.

3.4 Файл манифеста

Приложение Android также содержит так называемый файл манифеста. Этот файл содержит в себе основную информацию о приложении: определение его идентификатора, описание компонент, список разрешений и библиотек [16].

В текущей версии Android Studio (2.1.1) многие изменения вносятся в этот файл автоматически, например при создании сервиса у меня не было нужды вручную прописывать информацию о нём в манифесте. Тем не менее, файл был изменён, поэтому я привожу его исходный код в приложении 4.

4 Тестирование приложения

Для демонстрации работы приложения я выбрала сравнение интерфейсов двух приложений для заказа такси. Назовём их условно Uberetti и Klassetti (названия изменены).

В приложении Uberetti по умолчанию адреса поездки не вводятся пользователем вручную, а выбираются им на карте. Также изначально точкой отправления обозначено текущее местоположение. Приложение Klassetti предлагает вводить адреса текстом, не выводя текущее местоположение в качестве точки отправления.

Заданием для фокус-группы было заказать такси от текущего местоположения до Эрмитажа. Предполагается, что интерфейс приложения Uberetti намного удобнее и проще для реализации этой цели.

Снимок экрана с результатом тестирования представлен на рисунке 4.1.

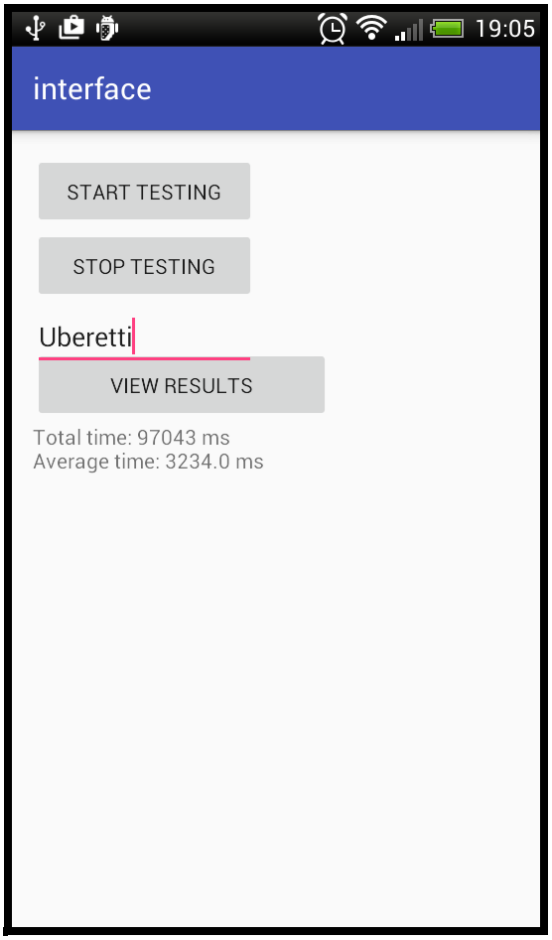

Рисунок 4.1 – Снимок экрана с результатом работы

Тестирование проводилось при участии 7 человек. Результаты приведены в таблице 4.1.

Таблица 4.1 – Результаты тестирования

№	Время заказа (мс)		Среднее время между касаниями экрана (мс)	
	Uberetti	**Klassetti**	**Uberetti**	**Klassetti**
1	109043	287298	2870	3283
2	138290	336092	3241	3713
3	121069	320522	3105	3398
4	162045	348156	3489	3667
5	99856	309808	2811	3917
6	147891	297189	3256	3452
7	117233	371239	3098	3875

Фокус-группа была сравнительно небольшой, но в целом ожидаемый результат удалось получить (таблица 4.2).

Таблица 4.2 – Сравнение тестируемых приложений

Приложение	**Среднее время заказа (с)**	**Среднее время между касаниями экрана (с)**
Uberetti	128	3,12
Klassetti	324	3,62

Из таблицы видно, что заказ такси с помощью приложения Klassetti занял в два раза больше времени, а также потребовал в два раза больше касаний экрана.

ЗАКЛЮЧЕНИЕ

Для достижения цели, поставленной в рамках выпускной квалификационной работы, были выполнены следующие задачи:

- рассмотрены существующие методы оценки интерфейса;
- предъявлены требования к разрабатываемому приложению;
- изучена среда разработки Android Studio;
- исследованы механизмы для реализации системы;
- спроектированы и реализованы все элементы приложения;
- проведено тестирование с участием фокус-группы.

Можно сделать вывод, что поставленной цели удалось достичь, поскольку в результате проведенной работы было создано приложение для оценки пользовательского интерфейса с учетом предъявленных ему требований.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Головач, В.В. Дизайн пользовательского интерфейса: Искусство мыть слона: [Электронный документ]. (http://uibook2.usethics.ru/uibookII.pdf). Проверено 18.04.2016.
2. Designing the User Interface: Strategies for Effective Human-Computer Interaction / Shneiderman B., Plaisant C., Cohen M., Jacobs S.: Pearson, 2010. – 624 с.
3. Kirakowski J., Corbett M. SUMI: The software usability measurement inventory // British Journal of Educational Technology. – 1993. – № 24. – 210 - 212 с.
4. Card, S.K. The Psychology of Human-Computer Interaction / S.K. Card, T.P. Moran, Newell A.: Lawrence Erlbaum Associates, 1983. – 469 с.
5. ГОСТ Р ИСО/МЭК 9126-93 // Госстандарт России. – 1993. – № 267. – 13 с.
6. Концаренко Ф. Где разрабатывать хорошо: сравнение Android, iOS и Windows Phone: [Электронный документ]. (https://vc.ru/p/ios-android-iphone). Проверено 25.05.2016.
7. Суягин С. Android vs iOS: какая платформа лучше для разработчиков: [Электронный документ]. (http://lifehacker.ru/2013/11/19/android-vs-ios-kakaya-platforma-luchshe-dlya-razrabotchikov/). Проверено 25.05.2016.
8. Uddin I. How to Install Xcode on Windows 10, 8 or 8.1and 7 for iOS SDK: [Электронный документ]. (https://www.alltechbuzz.net/install-xcode-on-windows-for-ios-sdk/). Проверено 26.05.2016.
9. Виноградов Д. Учебник по Android. Уроки для начинающих: [Электронный документ]. (http://startandroid.ru/ru/uroki/vse-uroki-spiskom/12-urok-3-sozdanie-avd-pervoe-prilozhenie-struktura-android-proekta.html). Проверено 05.06.2016.
10. Android Studio. The Official IDE for Android: [Электронный документ]. (https://developer.android.com/studio/index.html?hl=ru). Проверено 06.0.2016.
11. Study Java. Установка Android Studio: [Электронный документ]. (http://study-java.ru/java-dlya-android/urok-1-ustanovka-android-studio/). Проверено 06.06.2016.
12. About SQLite: [Электронный документ]. (http://www.sqlite.org/about.html). Проверено 06.06.2016.

13. Wikipedia. The Free Encyclopedia. SQLite: [Электронный документ]. (https://en.wikipedia.org/wiki/SQLite). Проверено 06.06.2016.

14. Климов, Александр. Теория Android: [Электронный документ]. (http://developer.alexanderklimov.ru/android/theory/services-theory.php). Проверено 06.06.2016.

15. Developers. Responding to Touch Events: [Электронный документ]. (https://developer.android.com/training/graphics/opengl/touch.html?hl=ru). Проверено 07.06.2016.

16. Developers. Манифест приложения: [Электронный документ]. (https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=ru). Проверено 07.06.2016.

ПРИЛОЖЕНИЕ 1

Исходный код реализации интерфейса приложения

Содержимое файла strings.xml. Включает в себя строковые ресурсы, используемые в activity_main.xml.

```
<resources>
    <string name="app_name">Test Interface</string>
    <string name="StartBtn">Start Testing</string>
    <string name="StopBtn">Stop Testing</string>
    <string name="ResBtn">My Results</string>
    <string name="AppTxt">Enter app name</string>
    <string name="ResTxt">Results</string>
</resources>
```

Содержимое файла activity_main.xml. Включает в себя описание всех представлений, выводимых на экран.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="kas.testinterface.MainActivity"
    android:contextClickable="false">

    <Button
        android:layout_width="150dp"
        android:layout_height="50dp"
        android:text="@string/StartBtn"
        android:id="@+id/button"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:onClick="onClickStart" />

    <Button
        android:layout_width="150dp"
```

```
        android:layout_height="50dp"
        android:text="@string/StopBtn"
        android:id="@+id/button2"
        android:layout_below="@+id/button"
        android:layout_alignLeft="@+id/button"
        android:layout_alignStart="@+id/button"
        android:onClick="onClickStop" />

    <Button
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:text="@string/ResBtn"
        android:id="@+id/button3"
        android:layout_below="@+id/button2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="30dp"
        android:onClick="onClickResults" />

    <TextView
        android:layout_width="350dp"
        android:layout_height="200dp"
        android:text="New Text"
        android:id="@+id/textView"
        android:layout_below="@+id/button3"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true" />

    <EditText
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_below="@+id/button2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:text="@string/AppTxt" />

</RelativeLayout>
```

ПРИЛОЖЕНИЕ 2

Исходный код основного Activity

Содержимое файла MainActivity.java.

```
package kas.testinterface;

// Импорт необходимых компонент
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    // Подключение к базе данных
    DBHelper dbh;
    SQLiteDatabase db = dbh.getWritableDatabase();
    TextView tv;
    EditText et;

    // Метод, вызываемый при открытии приложения
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        dbh = new DBHelper(this);
        tv = (TextView) findViewById(R.id.textView);
        et = (EditText) findViewById(R.id.editText);
    }

    // Обработчик нажатия кнопки Start. Запуск сервиса
    public void onClickStart(View v){
        db.execSQL("delete * from table test;");
        startService(new Intent(this, GlobalTouch.class));
    }
```

```
    // Обработчик нажатия кнопки Stop. Остановка сервиса
    public void onClickStop(View v){
        stopService(new Intent(this, GlobalTouch.class));
    }

    // Обработчик нажатия кнопки просмотра результатов
    public void onClickResults(View v){

        // Создание запроса данных из таблицы тестирования
        // На выходе получаем курсор
        String[] columns = new String[] {"max(clicktime) -
min(clicktime) as total", "(max(clicktime) -
min(clicktime))/max(clickno) as ave"};
        Cursor c = db.query("test", columns, null, null, null, null,
null);
        StringBuilder w = new StringBuilder();

        // Установка позиции курсора на первую строку выборки
        if (c.moveToFirst()) {

            // Определение строк в таблице
            String appname = et.getText().toString();
            int total = c.getColumnIndex("total");
            double ave = c.getColumnIndex("ave");

            // Получение данных из таблицы в цикле
            do {
                // Создание строки для вывода
                w.append("Total time: ").append(total).append(" ms
\n");
                w.append("Average time: ").append(ave).append(" ms
\n");

                // Запись в таблицу results
                cvv.put("app", appname);
                cvv.put("totaltime", total);
                cvv.put("avetime", ave);
                db.insert("results", null, cvv);

            // Переход на следующую строку либо выход из цикла
            } while (c.moveToNext());
        } else
            w.append("0 rows");
        // Закрытие курсора
        c.close();
        // Вывод полученных результатов на экран
        tv.setText(w);
    }
```

```
// Класс для работы с базой данных
class DBHelper extends SQLiteOpenHelper {
        public DBHelper(Context context) {
            // Конструктор класса
            super(context, "myDB", null, 1);
        }

        @Override
        public void onCreate(SQLiteDatabase db){
            // Создание таблиц для записи результатов тестирования
            db.execSQL("create table test ("
                    + "clickno integer,"
                    + "clicktime real " + ");");
            db.execSQL("create table results ("
                    + "app text,"
                    + "totaltime real,"
                    + "avetime real " + ");");
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

        }
    }
}
```

ПРИЛОЖЕНИЕ 3

Исходный код сервиса

Содержимое файла GlobalTouch.java.

```
package kas.testinterface;

// Подключение необходимых компонент
import android.app.Service;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.PixelFormat;
import android.os.IBinder;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.WindowManager;
import android.widget.Toast;

public class GlobalTouch extends Service implements OnTouchListener{
public GlobalTouch() {
}

int i = 0;

// Метод, срабатывающий при запуске сервиса
@Override
public void onCreate() {
    super.onCreate();
    // Вывод на экран уведомления о запуске
    Toast.makeText(this, "Тестирование начато",
Toast.LENGTH_SHORT).show();

    // Создание представления, сканирующего нажатия на экран
    WindowManager.LayoutParams params = new
WindowManager.LayoutParams(
            0,
            0,
            WindowManager.LayoutParams.TYPE_SYSTEM_ALERT,
            WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
                    | WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
                    |
WindowManager.LayoutParams.FLAG_WATCH_OUTSIDE_TOUCH,
            PixelFormat.TRANSLUCENT);
    WindowManager wm = (WindowManager)
getSystemService(WINDOW_SERVICE);
    View tiny = new View(this);
    tiny.setOnTouchListener(this);
    wm.addView(tiny, params);
```

```
    // Открытие базы данных
    db = openOrCreateDatabase("myDB", Context.MODE_PRIVATE, null);

}


// Метод остановки сервиса. Уведомление об остановке
@Override
public void onDestroy(){
    Toast.makeText(this, "Тестирование окончено",
Toast.LENGTH_SHORT).show();
}
@Override
public IBinder onBind(Intent intent) {
    return null;
}

// Метод, обрабатывающий нажатия на экран
@Override
public boolean onTouch(View v, MotionEvent event) {

    // Условие нажатия
    if (ev.getAction() == MotionEvent.ACTION_DOWN) {

        // Создание объекта для данных
        ContentValues cv = new ContentValues();
        // Запись данных в объект cv
        cv.put("clickno", i++);
        cv.put("clicktime", (long) ev.getDownTime());
        // Вставка данных в таблицу
        db.insert("test", null, cv);
    }
    return false;
}
}
```

ПРИЛОЖЕНИЕ 4

Исходный код файла манифеста

Содержимое файла AndroidManifest.xml, инкапсулирующего всю архитектуру приложения.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kas.testinterface">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <uses-permission
android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
        <service
            android:name=".GlobalTouch"
            android:enabled="true"
            android:exported="true"></service>
    </application>

</manifest>
```

ЗАКЛЮЧИТЕЛЬНЫЙ ЛИСТ РАБОТЫ

Выпускная работа бакалавра выполнена мною самостоятельно. Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Список использованных источников: 16 наименований.
Работа выполнена на 36 листах,
включая приложения на 8 листах.

Один экземпляр сдан на кафедру.

Подпись_____/_____/
(фамилия, инициалы)

Дата «_____» _____ 20_____г.