МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Управление корпоративными информационными процессами
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование и разработка методики оценки пользовательского интерфейса»

Обучающийся А.Н. Мамаев
(Инициалы Фамилия) (личная подпись)

Руководитель канд.пед.наук, доцент, О.В. Оськина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Оглавление

Введение

Актуальность работы. Проблема создания качественных интерфейсов остается сложной задачей, так как с одной стороны находятся ожидания пользователей к качеству интерфейса (удобство использования, удобство обучения, понятность), а с другой ожидания разработчиков к качеству интерфейса (функциональность, надежность, переносимость), и очень часто данные характеристики находятся в противоречии друг другу.

Научная проблема исследования – заключается в сложности при оценке пользовательского интерфейса разными заинтересованными людьми, так как разработка интерфейса является комплексной задачей, находящейся на стыке многих технических дисциплин: дизайн, математика, техническая эстетика, психология, эргономика и в магистерской выпускной квалификационной работе требуется разработать методику оценки качества пользовательских интерфейсов, которая позволит учесть все аспекты.

Цель работы – повысить скорость создания качественных пользовательских интерфейсов на начальных стадиях разработки программных продуктов с помощью методики оценки пользовательского интерфейса.

Объект исследования – пользовательские интерфейсы.

Предмет исследования работы – методика оценки пользовательских интерфейсов программных приложений.

Гипотеза исследования – предполагается что, использование алгоритма анализа текста интерфейса до этапа внедрения программ в опытную эксплуатацию, позволит улучшить существующие методики анализа пользовательского интерфейса.

Для поставленной цели необходимо решить следующие задачи:

– провести анализ источников литературы-по теме «исследование и разработка методики оценки пользовательского интерфейса»;

– дать краткую характеристику организации, для которой разрабатывается методика оценки пользовательского интерфейса;

– проанализировать инструменты создания пользовательских интерфейсов;

– сделать анализ существующих методик оценки пользовательских интерфейсов;

– проанализировать вопросы качества и создания набора унифицированных метрик для оценки качества пользовательских интерфейсов;

– разработать методику оценки пользовательских интерфейсов;

– разработать модуль программы для новой методики оценки пользовательских интерфейсов;

– и провести апробацию методики оценки пользовательских интерфейсов.

Теоретическая основа диссертационного исследования магистерской квалификационной работы состоит в систематизации информации по разработке пользовательских интерфейсов, обобщении материала по методике оценки качества пользовательских интерфейсов.

Методологическая основа исследования в основу исследования положен метод экспертной оценки.

Основные этапы исследования включают изучение существующих методик оценки пользовательских интерфейсов, выработка на основе лучших методик оценки собственной методики оценки пользовательских интерфейсов, а также разработку системы совместной работы над пользовательским интерфейсом, с учетом предложенной методики.

Апробация исследования по результатам выполнения магистерской работы опубликовано три статьи.

Результаты работы докладывались на научно-практических конференциях, публиковались в отчетах научно-исследовательской работы.

Научная новизна исследования заключается в разработке методики оценки пользовательских интерфейсов, которая впервые позволит проводить качественный и количественный анализ пользовательских интерфейсов.

Теоретическая значимость диссертационного исследования состоит в описании качественных и количественных показателей пользовательских интерфейсов.

Практическая значимость диссертационного исследования состоит в разработке модуля программы для предложенной методики оценки качества пользовательских интерфейсов.

Положения, выносимые на защиту:

– Методика оценки пользовательского интерфейса,

– Модуль программы для предложенной методики оценки качества пользовательских интерфейсов(Приложение А).

Объем и структура диссертации:

– 79 страниц,

– 31 рисунок,

– 3 таблицы,

– 1 приложение.

.

Глава 1. Теоретико-методологические основы диссертационного исследования

1.1. Обзор литературы по проблеме исследования

Публикации по теме разработки и оценки пользовательских интерфейсов начались с момента создания графических пользовательских интерфейсов еще в прошлом веке. Сегодня количество публикаций огромно, этой проблемой занимаются как российские, так и зарубежные ученые, делятся своим опытом разработчики, маркетологи и копирайтеры, а также большое количество публикаций посвящено дизайну интерфейсов.

В магистерской работе большое внимание будет уделено рассмотрению публикаций, в которых поднимаются вопросы оценки пользовательских интерфейсов исследовательских, образовательных проектов, так как основным местом внедрения работы будет ООО «Национальные проекты», компанией занимающейся продвижением стартапов и научных работ в ИТ-сфере.

В публикациях Тиханычев О.В., показана практика по созданию и оценке программ для автоматизированных систем управления. А также рассмотрены вопросы которые затрагивают оптимизацию непосредственно разработки интерфейсов с помощью механизмов унификации и стандартизации элементов пользовательского интерфейса.

Подробно рассмотрены нормативные акты и регламенты на которые основана разработка программных интерфейсов в России, а именно ГОСТ серии 34 (для АСУ) и серии 19 (для отдельных программ). В ГОСТ описывается техническое задании на опытно-конструкторскую работу (ОКР) и детализируются в форме руководящих указаний главного конструктора ОКР.

В целом подходы, ориентированные на унификацию, упрощают работу по созданию пользовательских интерфейсов в рамках каждого отдельного

проекта. В то же время такой подход обеспечивают лишь частичное решение проблемы создания эффективного пользовательского интерфейса.

В публикациях С. В. Шибанов, большой акцент делается на унификации процесса разработки интерфейсов с помощью паттернов. «Паттерны помогают сэкономить время на организацию кода и сосредоточиться на решении задачи. Существуют паттерны, позволяющие грамотно реализовать отдельные решения в коде, например, паттерн «перечисление», есть паттерны, которые позволяют настраивать общение различных модулей, например паттерн «инверсия контроля». Применение паттернов позволяет упростить реализацию проекта, обеспечить читаемость кода, позволит использовать отдельные компоненты повторно, а также увеличит эффективность совместной разработки. Самые используемые и общие паттерны объединяют в порождающие (фабрика, модуль, строитель, одиночка), структурные (адаптер, мост, компоновщик) и поведенческие (посредник, итератор, наблюдатель). При разработке приложения с использованием пользовательского интерфейса используются паттерны, описывающие, как эффективно разделить код приложения по папкам, что реализовывать в конкретных файлах, а также как настраивать связи между компонентами» [22]. Примером подобного паттерна является паттерн MVC (Model-View-Controller) и его аналоги MVP (Model-View-Presenter) и MVVM (Model-View-ViewModel) (рисунок 1).

Рисунок 1 – Базовые паттерны интерфейсов

Но и как любая унифицированная разработка, данный подход не дает ответа на вопрос, какой интерфейс более удобный и может быть признан лучшим.

В работах В. Головач предлагает проводить оценку пользовательских интерфейсов основываясь на мотивах пользователей. Суть оценок мотивов можно свести к следующему:

Интерфейс, ориентированный на мотивы пользователей (GoalCenteredDesign) позволяет преодолеть бесконечный рост функциональности. Составив список мотивов целевых пользователей и просто сравнив их со списком задач- получим оценку адекватности и полноту списка задач.

С помощью этих концепций разработчик может провести подробный и глубокий анализ, который послужит основой для эффективного дизайна интерфейса.

Все чаще в работах стали уделять внимание оценки пользовательских интерфейсов мобильных приложений и web—разработки (TomTullis, BillAlbert, Нильсен Я..). В этих работах основной акцент на измерении качества интерфейсов сосредоточен на следующих показателях:

– время отклика программы,
– адаптируемость к различным аппаратным средствам,
– скорость загрузки страницы и т.д.

Так как проблема интерфейсов находится на стыке программирования и дизайна, то множество советов по созданию легких, юзабельных интерфейсов относят к советам дизайна и должны быть использованы на фазе проектирования приложений и на этом этапе лучшими инструментами является модульирование.

1.2 Современное состояние проблемы оценки пользовательских интерфейсов

Под понятием пользовательский интерфейс в выпускной квалификационной работе магистра будем понимать - наличие связи между пользователем и компьютером. Пользовательский интерфейс должен решать задачи пользователя.

У пользовательского интерфейса выделяют следующие составные части:

– часть пользовательского интерфейса, связанная с аппаратно-программной реализацией;
– часть пользовательского интерфейса, связанная с активностями со стороны пользователя с его реакциями с помощью мыши и клавиатуры, а также сейчас необходимо включать обработку и сенсорного нажатия.

Качество интерфейсов, как составной части программного обеспечения определяется в стандарте ISO 9126. Качественный пользовательский интерфейс – это совокупность всех характеристик, которые могут удовлетворить высказанные или подразумеваемые потребности пользователя.

Целью создания пользовательского интерфейса программных продуктов является отображение данных и подписей эффективно, чтобы пользователь мог адекватно все воспринимать, и отображения на экране

гаджетов (мониторов, экране смартфона и планшета, умных часов) привлекало внимание к наиболее важной части информации.

Основные принципы создания пользовательского интерфейса, на которые будут сделан акцент в выпускной квалификационной работе магистра:

– Естественность – качественный интерфейс не вызывает сложностей;

– Непротиворечивость – качественный интерфейс не противоречит никакой части собственных модулей;

– Неизбыточность – качественный интерфейс требуем минимум для управления;

– Доступ к системе помощи – качественный интерфейс содержит инструкции, понятные пользователю;

– Гибкость – качественный интерфейс должен быть ориентирован на пользователей с разными уровнями подготовки.

Качество программного обеспечения определяется в стандарте ISO 9126 (рисунок 2).

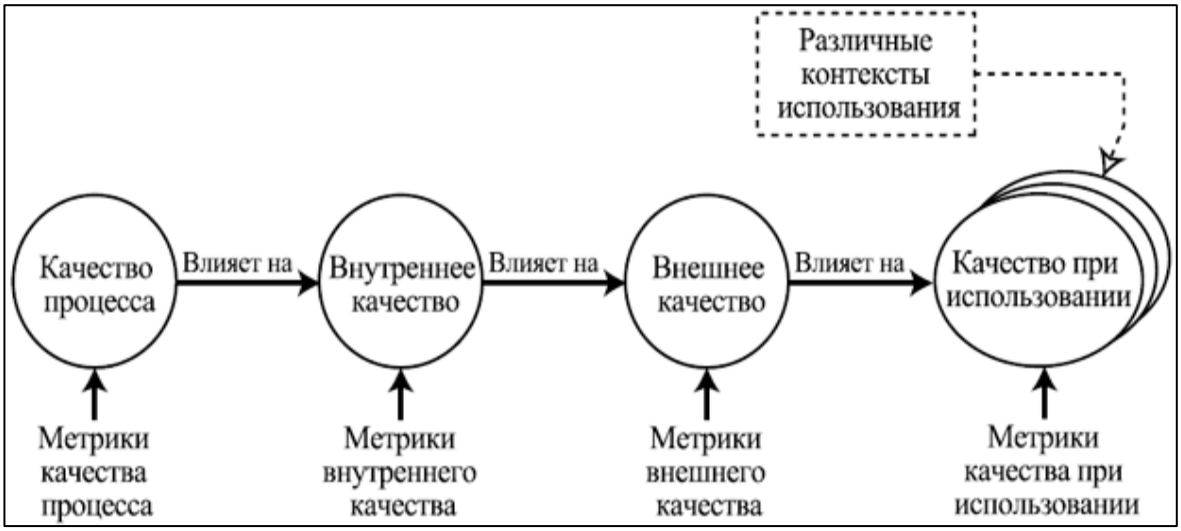

Рисунок 2 – Качество интерфейса

В выпускной квалификационной работе магистра будем рассматривать три вида качества пользовательских интерфейсов:

– внутреннее качество, которое связанно с характеристиками интерфейса без учета поведения;

– внешнее качество , которое связанно с характеристиками поведения интерфейса;

– и качества пользовательский интерфейс при использовании в различных контекстах, которое связанно с характеристиками при конкретных сценариях работы программами.

Юзабилити. В выпускной квалификационной работе магистра особое и отдельное внимание уделим эффективности и удобству пользования интерфейсом (юзабельности), в него входят следующие атрибуты:

– удобство использования (usability) или практичность,

– понятность (understandability),

– удобство обучения (learnability),

– удобство работы (operability),

– привлекательность (attractiveness),

– соответствие стандартам удобства использования (usabilitycompliance).

Определение качественного интерфейса программных продуктов будет состоять из набора качеств и можно говорить о качестве интерфейса, если он объединяет следующие понятия:

– пользовательский интерфейс программного продукта - «удобный», «простой» и «юзабельный»;

– пользовательский интерфейс программного продукта обладает высокими эргономическими показателями;

– пользовательский интерфейс программного продукта оптимизирован под своих пользователей;

– пользовательский интерфейс программного продукта оптимизирован под задачи пользователей;

– пользовательский интерфейс программного продукта оптимизирован под мотивы пользователей;

– пользовательский интерфейс программного продукта обладает высокими показателями юзабилити;

– пользовательский интерфейс программного продукта адекватен деятельности пользователей;

– пользовательский интерфейс программного продукта коммерчески успешен.

«Качественный пользовательский интерфейс является «лицом» программы, определяющим удобство её использования и, как результат, желание пользователя её эксплуатировать в процессе выработки управленческих решений» [15].

«Существование проблемы разработки качественных пользовательских интерфейсов и их оценки определяется целым рядом факторов. В выпускной квалификационной работе магистра остановимся на следующих:

– низкая унификация интерфейсов;

– мало взаимодействия между специалистами смежных областей, способных оценить потребности пользователя и сформулировать пользовательские качества интерфейса уже на этапе разработки;

– отсутствия единых метрик ,

– неопределенность с единой методикой оценки качества пользовательских интерфейсов» [16].

«В совокупности, эти факторы не позволяют обеспечить требуемое качество интерфейсов разрабатываемых прикладных программ, чем определяется общее снижение эффективности автоматизированной поддержки принятия решений» [1].

Вывод по 1 главе

В первой главе были рассмотрены существующие публикации по теме исследования. Определенно, что тема является актуальной и подходы к изучению проблемы можно рассматривать с разных сторон.

В результате рассмотрения вопроса «Теоретико-методологические основы диссертационного исследования» в первой главе был проведен анализ публикаций по данному направлению работы, рассмотрены основополагающие вопросы понятия интерфейса и качества интерфейсов.

2. Анализ существующих методов и подходов к решению сформулированной проблемы исследования

2.1. Методы оценки качества пользовательского интерфейса

«На процесс проектирования качественного пользовательского интерфейса программного продукта, набольшее влияние оказывают субъективные представления проектировщика о понятности, удобстве и красоте.

Существует целый ряд подходов к оценки качества пользовательских интерфейсов программных продуктов. В выпускной квалификационной работе магистра все методы разделим на две группы:

– методы оценки качества пользовательских интерфейсов программных продуктов с помощью тестирования

– методы оценки качества пользовательских интерфейсов программных продуктов без тестирования, основанные на формальных расчетах.

И те, и другие методы одинаково применимы как для оценки интерфейса традиционных программных продуктов, так и Web-приложений» [3].

Оценка качества пользовательского интерфейса программных продуктов процесс достаточно субъективный и трудно формализуемый. В выпускной квалификационной работе магистра попытаемся разработать методику для создания хорошего интерфейса.

Рассмотрим методы, которые можно формализовать и использовать при оценке научных проектов, реализованных в виде программных приложений.

Стандартизированность и соответствие рабочей среде – выбор разработчиком стандарта типа пользовательского интерфейса, адекватного предметной области и используемой ОС.

Бенчмаркинг - практика сравнения продукта с его прямыми конкурентами. Бенчмаркинг — важный этап разработки программного продукта в соответствии с философией интерфейсоянного управления

качеством. Бенчмаркинг обеспечивает в большей степени получение информации о функциональных недочетах, а не о количественных метриках (Таблица 1).

Таблица 1 – Бенчмаркинг сравнение

Инструменты навигации	Проект 1	Проект 2	Проект 3	Проект 4	Проект 5	Проект 6	Проект 7	Необходимость в критерии
Временная шкала	+	+	+	+	+	-	+	Обязательно
Поисковая строка	+	+	+	+	+	+	-	Обязательно
Фильтрации каналов	-	-	+	-	-	+	+	Обязательно
Календарь	+	-	+	+	+	+	+	Не критично
Фильтрация по времени	+	+	+	+	-	+	-	Не критично
Превью передач	-	+	-	-	+	-	+	Не обязательно
Фильтрация по каналам	-	+	+	-	+	+	-	Не обязательно

Достоинством этого метода является то, что можно использовать рабочие и эффективные практики на основе примера других компаний. Это позволяет избежать большего количества ошибок при поиске других практик, которые могут быть менее эффективны.

Экспертная оценка. В выпускной квалификационной работе магистра метод экспертной оценки качества интерфейса является знаковым , поэтому рассмотрим его более подробно. «Суть метода заключается в исследовании, насколько анализируемый интерфейс соответствует известным правилам, рекомендациям и методикам. В ходе такой оценки выявляются несоответствия и противоречия, которые и должны быть устранены» [2].

Вначале процесса экспертной оценки эксперт составляет список характеристик интерфейса в порядке их значимости. При оценке проверяют насколько тот или иной интерфейс соответствует списку требований.

Данный метод во многом зависит от опыта, компетентности и профессионализма экспертов. Поэтому возникает первоначальная сложности с подбором экспертов.

Эвристическая оценка - была разработана Якобом Нильсеном и Рольфом, Моличем, которые надеялись с ее помощью сократить продолжительность проведения проверки по контрольному списку. При эвристической оценке вместо десятков и сотен конкретных требований интерфейс проверяется на соответствие всего нескольким общим принципам.

Анкетирование пользователей по итогам взаимодействия с системой (метод фокус групп) – чаще всего их проводят на основе ранее разработанных анкет SUMI, QUIS,MUMMS, IsoMetrics и др.

«С помощью метода анкетирования с вопросами о качестве пользовательского интерфейса программного продукта пользователей можно получить достаточно глубинную информацию об особенностях поведения конечных пользователей, которую просто невозможно выяснить другими методами» [2]. Этот метод позволяет и лучше понять пользователей – выявить волнующие их проблемы и пожелания (рисунок 3, 4).

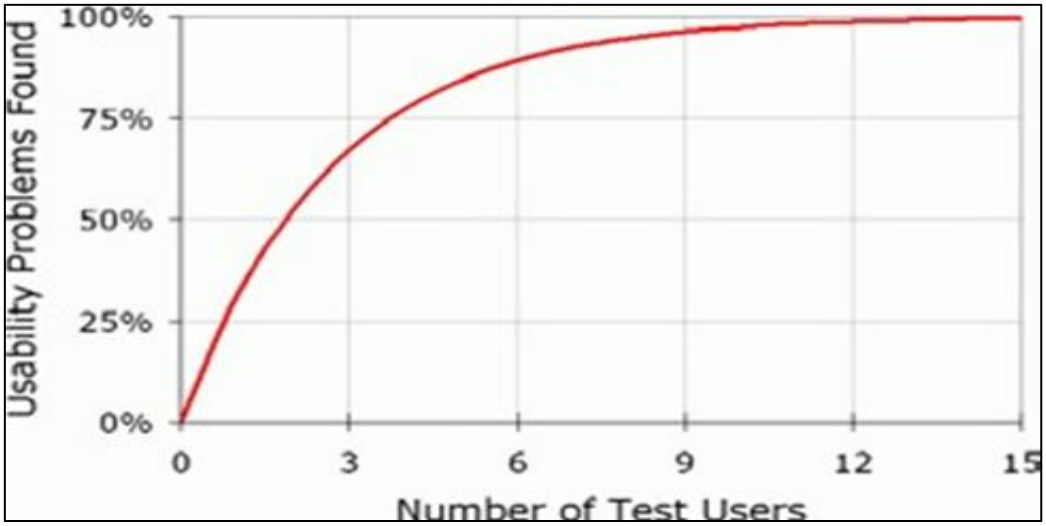

Рисунок 3 – Зависимость количества выявленных проблем от количества пользователей

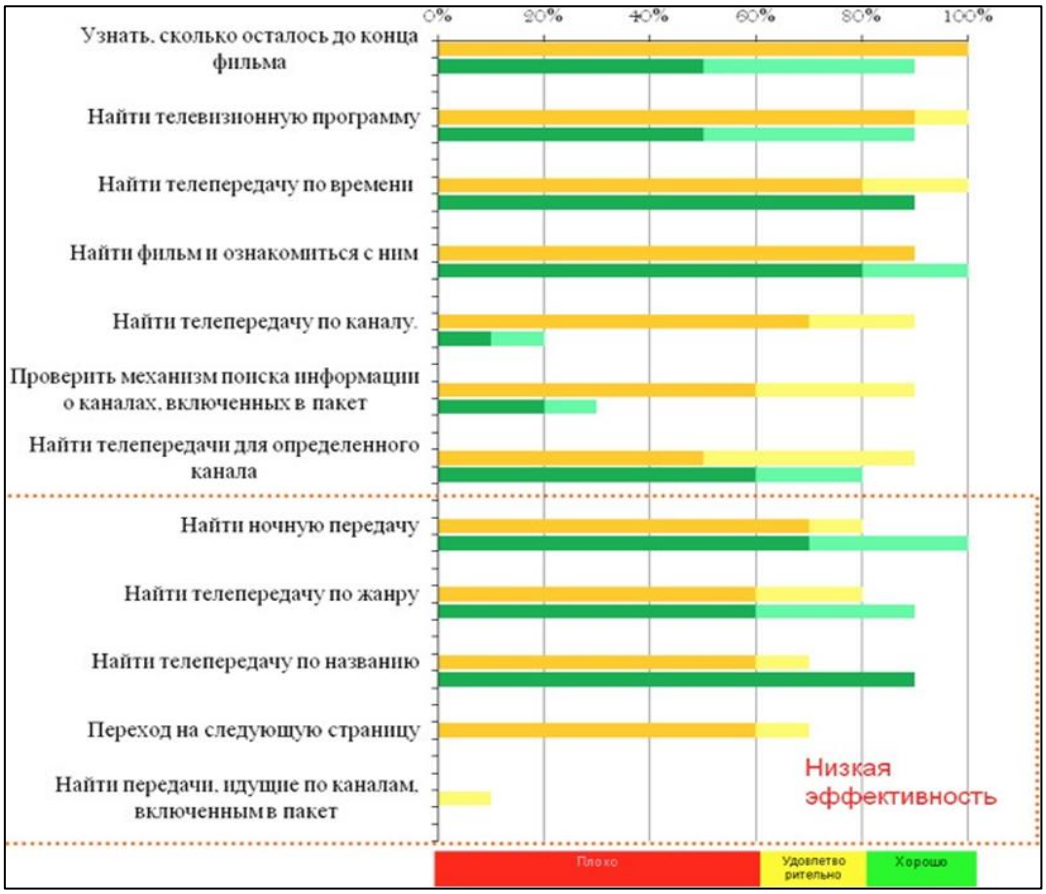

Рисунок 4 – Результаты оценки качества пользовательских интерфейсов на основании анкетирования пользователей

Метод анкетирования пользователей всегда актуален как средства оценки интерфейса.

Количественные оценки качества пользовательского интерфейса программного продукта (рисунок 5) базируются на экспериментальных данных.

Количественные оценки качества пользовательского интерфейса программного продукта основаны на следующих характеристиках:

– количество ошибок, найденных при тестировании интерфейса,

– средний уровень выполнения задач,

– объединенная метрика (single usability metric, sum).

Достоинством этого метода является точность и объективность оценки пользовательских интерфейсов. Этот метод допустимо использовать при

тестировании интерфейса для ввода в дальнейшую эксплуатацию с целью исправления ошибок.

«Формальные методы оценки пользовательских интерфейсов семейство методов позволяющих провести моделирование выполнения той или иной задачи пользователем . GOMS это сокращение от английского Goals, Operators, Methods, andSelectionRules – Цели, Операторы, Методы и Правила выбора. Данный способ был предложен S. K. Card, T. P. Moran и A. Newell в 1983 году.

Идея метода заключается в том, что все действия пользователя можно представить как набор типовых составляющих составляющие (например, нажать ту или иную кнопку на клавиатуре, передвинуть мышь, и т.п.). Оценка качества интерфейса заключается в разложение выполняемой задачи на типовые составляющие, и вычисление времени, которое будет в среднем затрачиваться пользователем на выполнение этой задачи» [1] (рисунок 5-6).

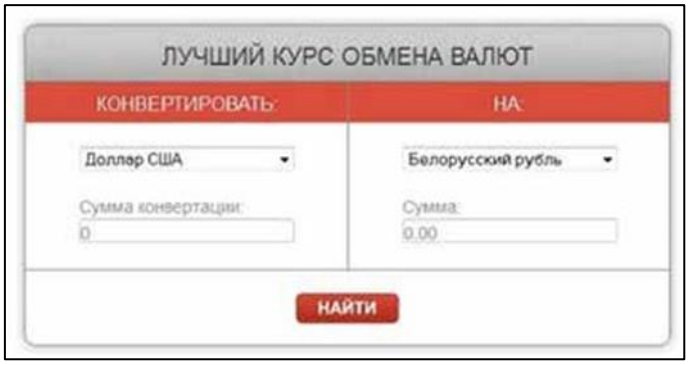

Рисунок 5 – Расчет по модели KLM-GOMS виджета конвертации валют. Шаг 1

При разработке метрики были учтены ограничения: метрика или комбинация метрик должны рассчитываться автоматически с помощью metadata пользовательского интерфейса

Количество контролов (Controls’ Count, CC) сравнивается с количеством строк кода (Lines of Code, LOC). Очевидно, что программа имеющая миллионы строк кода ожидаемо будет более сложной, чем программа, чье

количество строк кода измеряется в тысячах. Точно так же, интерфейс, имеющий большое количество контролов скорее всего более сложен, чем интерфейс, имеющий мало управляющих элементов.

В процессе описания существующих методов оценки пользовательских интерфейсов были выделены семь методов и подходов к оценки пользовательских интерфейсов программных приложений, которые можно условно поделить на три укрупненных группы:

– оценка количественных характеристик интерфейсов,
– оценка качественных характеристик интерфейсов,
– применение стандартов и шаблонов.

К первой группе - оценка количественных характеристик интерфейсов, можно отнести следующие методы:

– количественные оценки базирующиеся на экспериментальных данных,
– формальные методы оценки пользовательских интерфейсов.

Ко второй группе - оценка качественных характеристик интерфейсов, можно отнести следующие методы:

– экспертная оценка,
– эвристическая оценка,
– анкетирование пользователей.

К третьей группе – оценка с помощью применения стандартов и шаблонов интерфейсов. К этой группе можно отнести следующие методы:

– стандартизированность и соответствие рабочей среде,
– бенчмаркинг.

На основе проведенного обзора в предыдущем пункте в каждом методе выделим достоинства и недостатки и полученный анализ приведен в таблице 2.

Таблица 2 – Достоинства и недостатки существующих методов оценки пользовательских интерфейсов

Методы оценки пользовательских интерфейсов	Недостатки	Достоинства
Стандартизированность и соответствие рабочей среде	Не подходит для инновационных проектов	выбор стандарта пользовательского интерфейса, адекватного предметной области
Бенчмаркинг	Бенчмаркинг обеспечивает в большей степени получение информации о функциональных недочетах, а не о количественных метриках	этап разработки программного продукта в соответствии с философией интерфейсоянного управления качеством
Экспертная оценка	Данный метод во многом полагается на опыт, компетентность экспертов	исследование, насколько анализируемый интерфейс соответствует известным правилам.

Продолжение таблицы 2

Методы оценки пользовательских интерфейсов	Недостатки	Достоинства
Эвристическая оценка	вместо десятков и сотен конкретных требований интерфейс проверяется на соответствие всего нескольким общим принципам	Сокращение продолжительности проведения проверки по контрольному списку
Анкетирование пользователей	Затратный способ, сложности с выявлением фокус-группы для анкетирования	«Этот метод позволяет и лучше понять пользователей – выявить волнующие их проблемы и пожелания» [1]

Количественные оценки, базирующиеся на экспериментальных данных	Интерфейс – это также вопрос качества, которые в эксперименте не учитываются	«основан на следующих характеристиках: Количество ошибок Средний уровень выполнения задач Объединенняяметрика (Single Usability Metric, SUM)» [3]
Формальные методы оценки пользовательских интерфейсов	Рассмотрение только типовых задач пользователя	«моделирование выполнения той или иной задачи пользователем и на основе такой модели оценить качество интерфейса»[1]

Как видно из этой главы, методов оценки интерфейсов много, но одни из этих методов дороги (анкетирование пользователей), другие не дают ответы о качестве интерфейсов с точки зрения ответов на вопрос удобства интерфейса.

Поэтому предлагается объединить два подхода (количественного и качественного анализа) к оценки пользовательских интерфейсов и разработать концепцию программы для реализации.

Вывод по 2 главе

В результате рассмотрения вопроса «Анализ существующих методов и подходов к решению сформулированной проблемы исследования» во второй главе был сделан обзор существующих методик анализа пользовательских интерфейсов, проведен анализ достоинств и недостатков существующих методов оценки пользовательских интерфейсов.

Глава 3. Представление авторского решения интерфейсавленной в исследовании проблемы

3.1. Методика оценки пользовательских интерфейсов на основе качественного и количественного подхода

В рамках данной главы описывается методика получения качественных оценок пользовательских интерфейсов для оценки интерфейсов самими разработчиками. Проблему анализа и повышения качества, разрабатываемых интерфейсов предлагается решать с помощью интерфейсоянного анализа разработанного кода.

Вначале работы определим методику оценки интерфейса. В основу методики взят индивидуальный экспертный опрос, суть которого предполагает индивидуальный опрос участников команды в форме ответов на анкету и выставления экспертных оценок каждому вопросу.

«Метод оценки пользовательских интерфейсов на основе качественного и количественного подхода предполагает следующие шаги» [2]:

– определение всех лиц, так или иначе заинтересованных в оценке качества пользовательских интерфейсов программных продуктов,

– определение критериев, формирующих представление о качестве пользовательского интерфейса программного продукта для каждого из участников,

– приоритезацию критериев качества пользовательского интерфейса программного продукта, с учетом важности конкретного участника для компании, выполняющей проект, и важности каждого из критериев для данного участника,

– составление анкеты на основе выбранных критериев,

– проведение опроса всех участников команды, работающих над проектом.

– анализ, полученных результатов опроса» [2].

Алгоритм оценки представлен на рисунке 6.

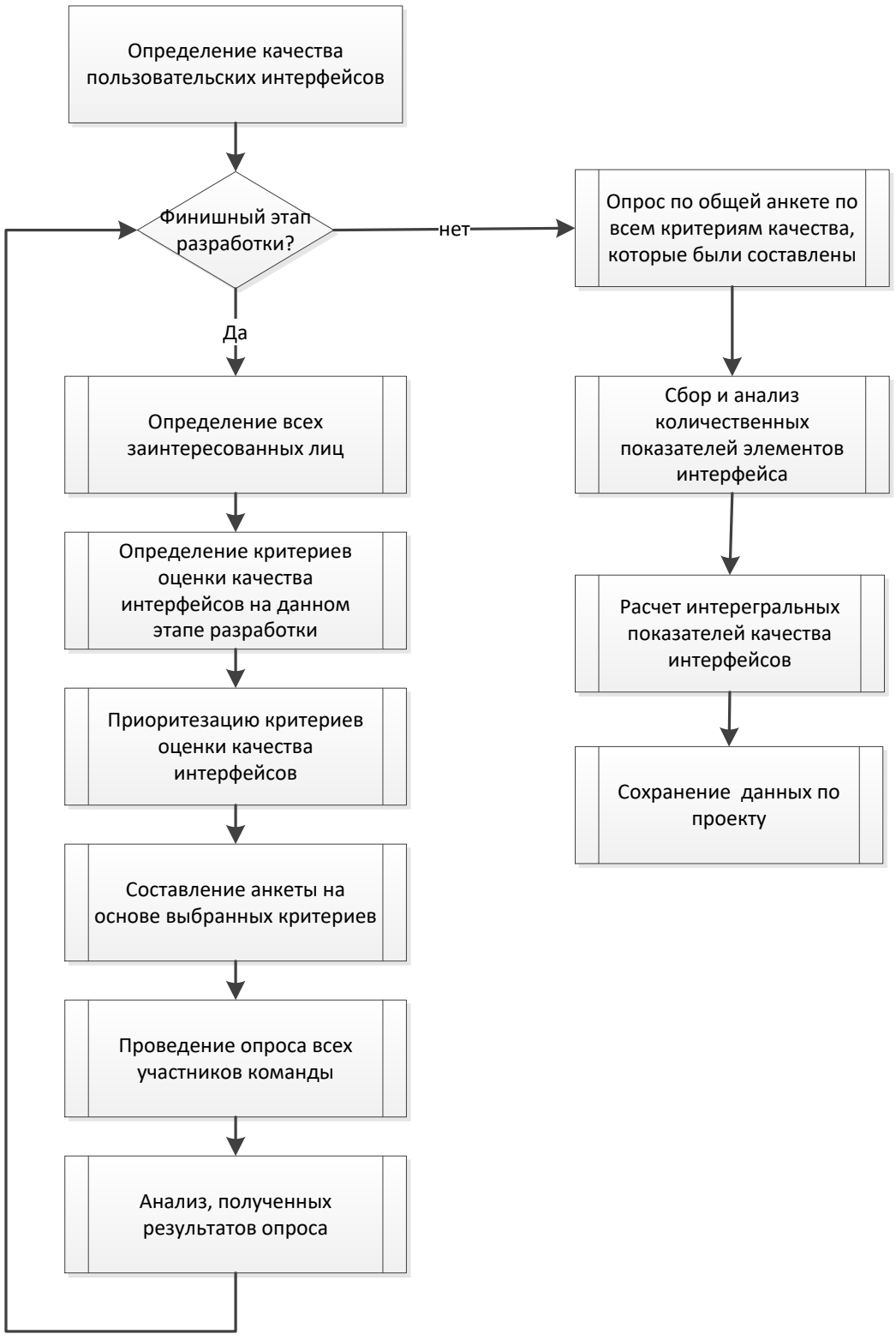

Рисунок 6 – Алгоритм оценки пользовательских интерфейсов

«Данная методика подходит как для совместной разработки, когда партнеры в команде равны – студенческие проекты, а также где в работе выделяют разные роли в проекте – научные стартапы.

Работая над проектом в программе для совместной разработки на определенных этапах завершения работ, одной из задач ставится прохождения анкетирования по вопросам качества получившихся интерфейсов. Для упрощения работы качество каждого критерия будет определяться как среднее значение всех ответивших, и чем больше балл получит критерий, тем значит критерий качества интерфейса максимально удовлетворяет всех разработчиков. Особое внимание следует уделить критериям, которые получат наименьшее количество баллов»[2].

«Согласованность мнения членов команды разработки можно оценивать по величине коэффициента конкордации (1)» [2]:

$$W = \frac{12S}{n^2(m^3-m)} \quad (1)$$

«где S - сумма квадратов отклонений всех оценок рангов каждого объекта экспертизы от среднего значения;

n - число экспертов;

m - число объектов критериев качества интерфесов.

Коэффициент конкордации изменяется в диапазоне 0<W<1, причем 0 – полная несогласованность, 1 – полное единодушие» [2].

«Данные этапы рекомендуется повторять до финишного этапа. На финишном этапе необходимо:

– провести анкетирование и опрос по всем критериям качества интерфейсов, которые возникали на любом из этапов;

– собрать количественные показатели интерфейсов: количество элементов по видам, надписи на элементах и т.д.;

– рассчитать показатель, в котором определить есть ли зависимость между качественными и количественными показателями. На данном этапе предлагается использовать простую линейную корреляцию» [2];

– сохранить данные по проекту.

Пользователь – человек, который пользуется или планирует пользоваться, программным продуктом, интерфейс которого оценивается

Разработчик – специалист, который разрабатывает программный продукт.

Эксперт – человек, который оценивает программный продукт и обладает навыками в использовании программных продуктов определенного класса.

В ходе опроса о качественных характеристик интерфейса, в роли экспертов на начальных стадиях разработки выступают члены команды разработки

Результаты работы алгоритма представляются в виде таблиц (таблица 3) экспертных оценок и графиков (рисунок 7), полученных результатов.

Таблица 3 – Таблица экспертных оценок

	Критерий 1	Критерий 2	Критерий 3	Критерий 4	Критерий ….
Пользователь 1					
Пользователь 2					
Пользователь 3					
Пользователь 4					
Пользователь 5					
Пользователь 6					
……					
Пользователь…..					

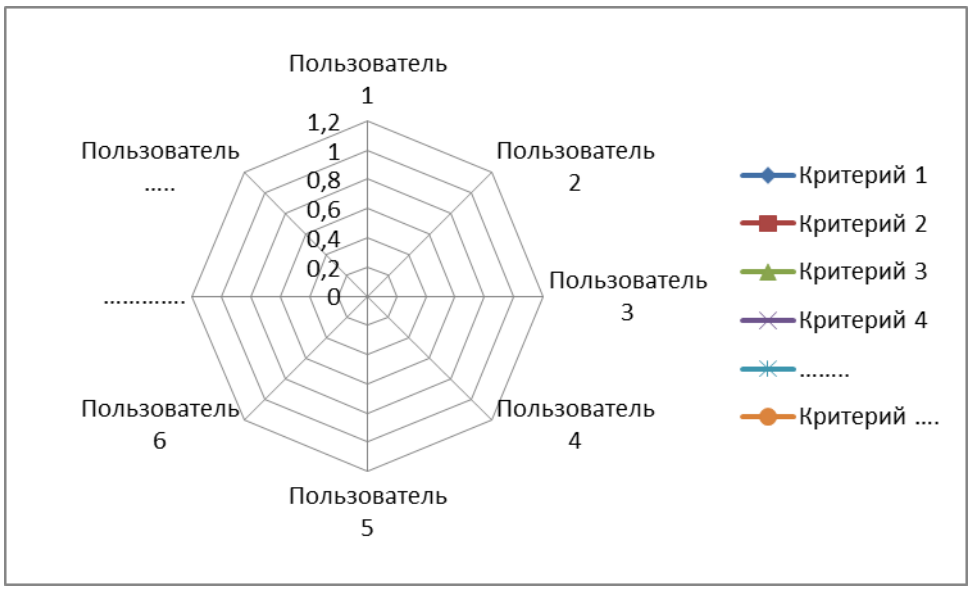

Рисунок 7 – Шаблон графика по экспертным оценкам

«Качественные оценки интерфейса предполагается оценивать вместе с количественными характеристиками, собранными в ходе разработки проекта и на основе полученной оценки проводить анализ качества разрабатываемых интерфейсов программных продуктов, тестирование которых стандартными средствами невозможно в силу отсутствия большого числа пользователей программных продуктов у программ, которые находятся на начальных стадиях разработки приложения и интерфейса» [2].

И предлагается на этапе сбора и анализа количественных показателей элементов интерфейса - добавить анализ интерфейса по ключевым словам (рисунок 8). Ключевыми словами будут слова кнопок и элементов интерфейса.

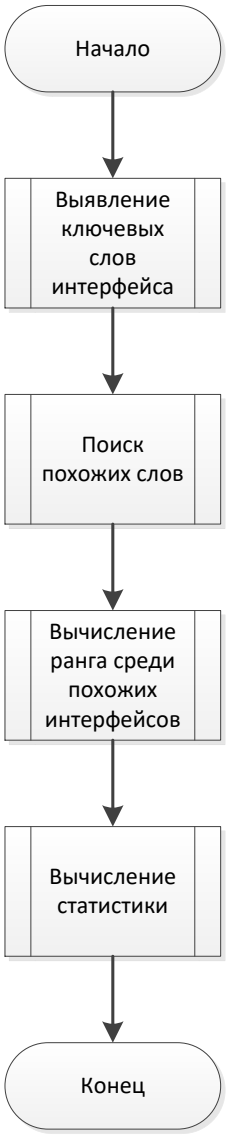

Рисунок 8 – Алгоритм анализа по ключевым словам

Анализ интерфейса по ключевым словам будет проходить в автоматическом режиме, и этот параметр будет относится к количественным характеристикам интерфейса.

«На текущий момент существует достаточно много различных методов разбиения текста на ключевые слова. Можно выделить два основных вида алгоритмов: статистические и гибридные. Рассмотрим каждый метод более подробно.

Статистический метод вычленения ключевых слов основан на статистике, он предполагает интерфейсостроение множества кандидатов

ключевых слов путем ранжирования всех словоформ или лексем документа по частоте.

Гибридный алгоритм комбинирует статистический метод извлечения ключевых слов с различными лингвистическими процедурами» [3].

«Существуют методы извлечения ключевых слов на основе графов. Эти методы используют взвешенные графы, в вершинах у которого находятся лексемы кандидаты в ключевые слова, а вес дуг зависит от того, на сколько кандидаты близки по смыслу. Для выбора кандидатов в ключевые слова используются различные алгоритмы из теории графов. Основным отличием этих методов является применение разных методов определения кандидатов в ключевые слова и определения близости между ними. Гибридные методы дают более точные результаты, чем статистические, но при этом они более сложные» [1].

Отдельным видом гибридных методов являются методы, использующие машинное обучение для вычленения ключевых слов. Эти методы, как правило, используют корпуса текстов, с размеченными ключевыми словами.

«В данной работе будет использоваться алгоритм TF-IDF (от англ. TF — term frequency, IDF — inverse document frequency). Он позволяет вычислить важность слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес слова пропорционален частоте употребления этого слова в исходном тексте и обратно пропорционален частоте его употребления в текстах всей коллекции» [3].

Частота встречи слова в тексте рассчитывается по формуле (2).

$$tf(t,d) = \frac{n_t}{\sum_k n_k}, \quad (2)$$

Где nt – число вхождений слова t в документ, а в знаменателе общее число слов в этом документе.

Частота встречи слова в коллекции документов рассчитывается по формуле (3).

$$idf(t, D) = log \frac{|D|}{|\{d_i \in D | \ t \in d_i\}|} \quad (3)$$

«Где, |D| - число документов в коллекции, $|\{d_i \in D | \ t \in d_i\}|$ - число документов в коллекции D, в которых встречается t (когда $n_t \neq 0$). Выбор основания логарифма в формуле не имеет значения, поскольку изменение основания приводит к изменению веса каждого слова на интерфейсоянный множитель, что не влияет на соотношение весов. Таким образом важность слова определяется по формуле (4).» [1]

$$TF - IDF(t, d, D) = tf(t, d) * idf(t, D) , \quad (4)$$

После того как определены ключевые слова выбранного текста интерфейса, необходимо вычислить ранг названия элементов интерфейса среди ранее разработанных интерфейсов, которые содержат те же ключевые слова. Для этого необходимо с базы ранее проанализированных и оцененных интерфейсов достать необходимые тексты элементов, по ключевым словам, и статистику по ним.

3.2 Проектирование программы для оценки пользовательских интерфейсов

Далее для демонстрации предложенной методики оценки пользовательских интерфейсов будет разработано программное решение.

Разработку программного решения начинают с этапа проектирование. Этап проектирования начнем с разработки диаграммы использования, на

которой можно определить для кого системы и ее основные функции (рисунок 9).

Рисунок 9- Диаграмма вариантов использования системы для оценки пользовательских интерфейсов

«На следующем этапе проектирования программы для реализации метода анализа качества пользовательских интерфейсов дополним диаграмму текстовым сценарием, который будет пояснять диаграмму, раскрывая содержание действий, выполняемых системой и действующими лицами» [2].

Сценарий представим в виде таблицы 3.

Таблица 3 – Описание диаграммы вариантов использования

Варианты использования	Ввод и редактирование оценки интерфейса, Идентификация в программе, Ввод комментариев, Просмотр и печать отчетов, Редактирование данных об интерфейсе, Загрузка количественных показателей интерфейса, Ввод текста интерфейса, Администрирование системы
Основные актеры	Администратор системы, Аналитик, Эксперт
Краткое описание	Для оценки пользовательских интерфейсов, программ, которые предназначены для реализации научных исследований или стартапов необходимо собрать количественные характеристики интерфейсов(возможно автоматический режим загрузки этих параметров или Администратором), также за качественную оценку интерфейсов отвечают Эксперты, которые оценивают интерфейсы с экспертной точки зрения. Качественная оценка может характеризоваться разным набором параметров в зависимости от предметной области программного приложения. Обязанности Администратора – полный доступ ко всем элементам программы для оценки интерфейсов, а также возможность создавать новых пользователей системы. Задача Аналитика, на основе количественных и качественных оценок интерфейса приложений формировать настраиваемые отчеты и принимать решения. Основным отличием программы от ранее существующих подходов в оценки качества интерфейсов предполагается наличие возможности «собирать и анализировать текстовую составляющую интерфейсов и на основе (базы знаний по другим интерфейсам, которые признаны лучшими) делать прогноз и оценку по качеству интерфейса предложенного на оценку» [3].
Цель	Провести оценку различных интерфейсов программ, предназначенных для реализации научных исследований или стартапов или . программ, которые находятся только на начальных стадиях разработки

Далее покажем, из каких классов будет состоять система для оценки пользовательских интерфейсов (рисунок 12).

«Основными классами являются:

– эксперт,

– оценка экспертов,

– аналитик,

– интерфейс,

– количественные параметры интерфейсов,

– тексты интерфейсов,

– отчеты,

– роли,

– администратор.

От класса Роли наследуются три класса:

– администратор,

– эксперт,

– аналитик.

Администратор – имеет доступ ко всем элементам программы.

Эксперт – имеет возможность интерфейсавить оценку, т.е. оценить интерфейс с качественной точки зрения.

Аналитик – получает доступ к результатам работы системы и может работать с отчетами.

Класс Интерфейс включает следующие классы:

– оценка экспертов,

– тексты интерфейса,

– количественные параметры интерфейсов.

В классе Оценка экспертов – хранится оценки полученные интерфейсом у эксперта. Один эксперт может оценить множество интерфейсов, и один интерфейс может быть оценен множеством Экспертов» [3].

Общая структура взаимодействия представлена на диаграмме классов(рисунок 10).

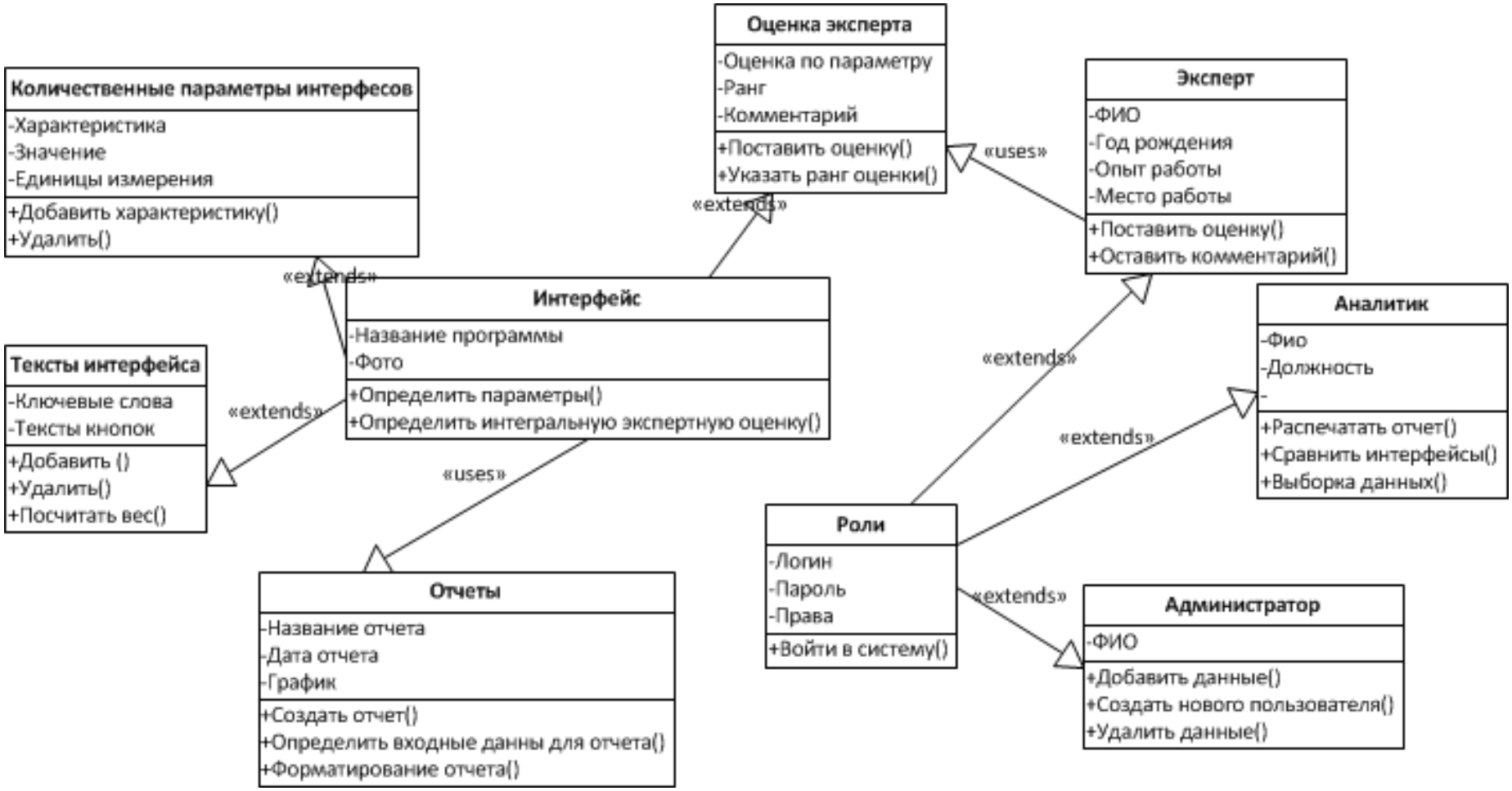

Рисунок 10 – Диаграмма классов системы для оценки пользовательских интерфейсов

«В классе Тексты интерфейса хранятся ключевые слова и тексты кнопок. Наличие данного класса позволит собирать и анализировать текстовую составляющую интерфейсов и на основе (базы знаний по другим интерфейсам, которые признаны лучшими) делать прогноз и оценку по качеству интерфейса предложенного на оценку.

Класс Количественные параметры интерфейсов – позволит собирать информацию о количественных метриках.

«Третий из этих показателей относится к субъективным и качественным, а первый, второй и четвертый могут быть оценены количественно. Скорость работы пользователя представляет собой время, которое пользователь затрачивает на выполнение задания предметной области» [1].

– K = 0.2 с – время, необходимое для нажатия клавиши клавиатуры;

– P = 1.1 с – время, необходимое для перемещения указателя мыши к определенной позиции на мониторе;

– H = 0.4 с – время, необходимое для перемещения руки пользователя с клавиатуры на мышь.

Оценка механического времени заключается в разложении действий пользователя на элементарные составляющие и вычислении среднего времени, которое будет затрачиваться пользователем на выполнение задачи. Несмотря на то, что метод KLM-GOMS был создан давно, его модификациями пользуются до сих пор.

Класс Отчет – использует данные класса Интерфейсы и через него получает все количественные и качественные показатели» [3].

3.3. Технология реализации модуля программного приложения для оценки пользовательских интерфейсов

В рамках дальнейшей работы над магистерской диссертацией были определенны следующие ключевые моменты:

– оценка пользовательских интерфейсов предполагает совместную работу над проектом;

– предложенная методика оценки интерфейсом предполагает экспертную оценку интерфейса всех членов команды и преобразование экспертных оценок в количественные оценки;

– для работы алгоритмов необходимо разработать систему совместной разработки над проектом.

«Ключевым параметром в выборе формата системы будет являться требование к ее доступности с более чем одной платформы. Удовлетворение данного требования можно добиться несколькими способами» [1]:

– нативные приложение для каждой платформы,

– кроссплатформенное приложение,

– веб-приложение.

«С учетом специфики целевой аудитории системы, первый и второй пункты не подходят. Кроссплатформенные и нативные приложения требуют большого количества вложение ресурсов и наличия специалистов для поддержания работы и доработки функциональной составляющей, что затрудняет возможность полноценного глобального масштабирования модуля в дальнейшем, потому данные варианты не являются валидными в данном случае» [1].

«Вариант веб-приложения в свою очередь обеспечивает малую стоимость разработки относительно других вариантов. Данный вариант так же имеет простоту поддержки, в связи с распространенностью технологий используемых в веб-приложения и популярности направления в целом, а так же большим количеством прибывающих специалистов данной области. Оставшимся преимуществом предыдущих вариантов является возможность общего доступа к приложению без подключения к сети Интернет, однако необходимость взаимодействия между участниками команды, а следственно синхронизации прогресса задач требует наличия соединения. Возможность дальнейшего

преобразования сайта в прогрессивное веб-приложение дает нам возможность дальнейшего масштабирования при доработке модуля.

В следствии приведенных выше факторов веб-приложение является идеально подходящим решением для удовлетворения пункта доступности системы» [2].

Вывод по 3 главе

В третьей главе был предложена авторская методика оценки качества пользовательских интерфейсов, которая основывается на количественной и качественной оценке интерфейсов. Также было спроектировано программное решение для реализации методики, а также проведен выбор технологии реализации модуля программного продукта для методики оценки качества пользовательских интерфейсов.

Глава 4. Представление экспериментальных и расчетных результатов апробации

4.1. Реализация модуля программного приложения для оценки пользовательских интерфейсов

Для долгосрочного хранения данных не привязанных к конкретному пользователю веб-приложения в общем случае используют внешнюю базу данных.

В следствии выбора серверного языка разработки будет целесообразно использовать веб-интерфейс СУБД phpMyAdmin распространяемый и поддерживаемый на большинстве серверов поддерживающих исполнение PHP и реляционную систему управления MySQL.

База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Взаимодействие с базами данных осуществляется с использованием языка интерфейсроения запросов SQL, его основной задачей является организация способа чтения и записи элементов базы данных.

На основе анализа данных можно выделить необходимые таблицы базы данных и их атрибуты представленные диаграммой на рисунке 11.

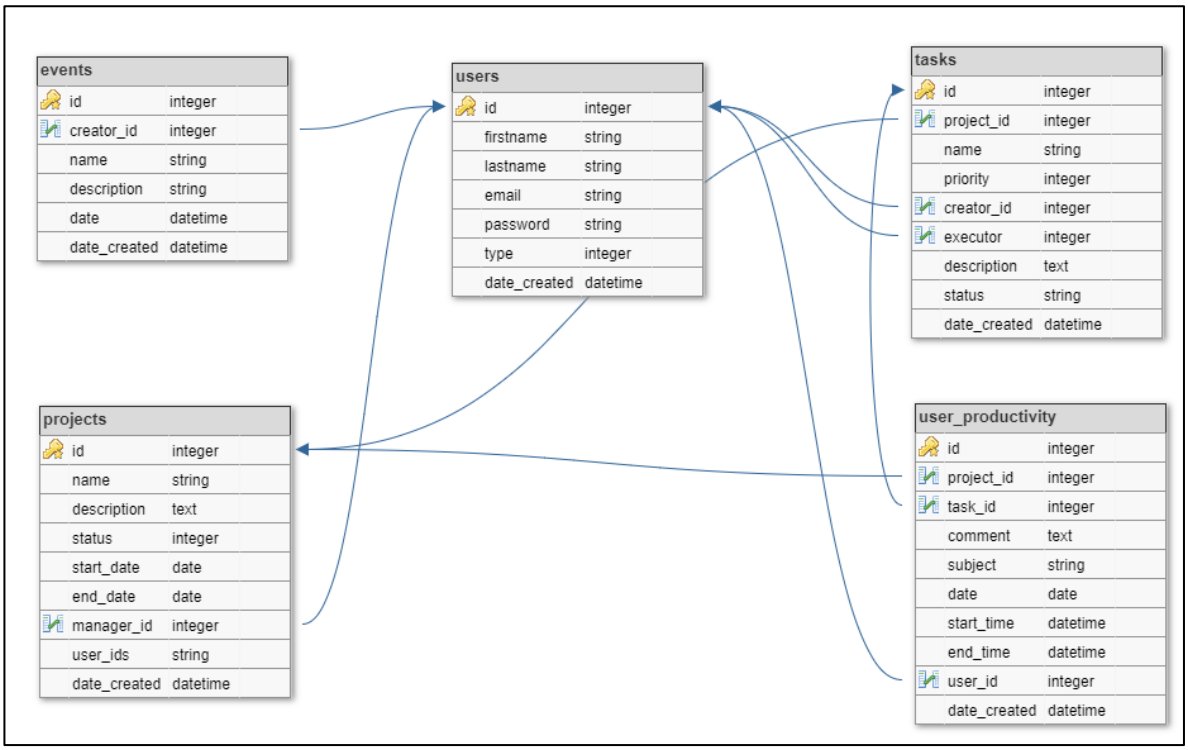

Рисунок 11 – Диаграмма базы данных

Реализация веб-приложения представленного в данной работе состоит из двух этапов: реализации веб-интерфейса и реализация API для взаимодействия с серверов.

Для реализации веб-интерфейса приложения был использован прогрессивный фреймворк Vue версии 2.6. и дизайн-фреймворк Vuetify, позволяющий создавать интерфейсы в соответствии с дизайн системой Material Design, разработанной компанией Google.

Устройство фреймворка Vue позволяет применять компонентный подход к проектированию приложения, реализуя возможность создания однофайловых компонентов. Однофайловый компонент представляет из разметку компонента представленного на языке разметки HTML или с применением JSX, скрипт позволяющий организовывать взаимодействие между частями системы и обрабатывать события происходящие внутри компонента реализованный на языке программирования JavaScript или TypeScript, каскадную таблицу стилей

применяемую глобально или только внутри компонента с использованием CSS или его препроцессоров.

Рассмотрим устройство компонента более подробно на примере реализованного в приложении компонента ошибки приведенного на рисунке 12.

```
<template>
  <v-app id="404">
    <v-container fluid fill-height>
      <v-layout align-center justify-center row>
        <div class="mr-3 hidden-sm-and-down">
          <img src="errorIcon" alt="" />
        </div>
        <div class="text-md-center">
          <h1>500</h1>
          <h2 class="my-3 headline">Ошибка продключения.</h2>
          <div><v-btn color="primary" @click="goHome">Вернуться на главную</v-btn></div>
        </div>
      </v-layout>
    </v-container>
  </v-app>
</template>

<script>
export default {
  data: () => ({
    errorIcon: null,
  }),
  created() {
    this.initialize()
  },
  methods: {
    initialize() {
      this.errorIcon = this.$store.getters.getImgIcon
    },
    goHome() {
      this.$router.push({ path: '/' })
    },
  },
}
</script>

<style lang="sass" scoped>
h1
  font-size: 150px
  line-height: 150px
  font-weight: 700
  color: #252932
  text-shadow: rgba(61, 61, 61, 0.3) 1px 1px, rgba(61, 61, 61, 0.2) 2px 2px, rgba(61, 61, 61, 0.3) 3px 3px
</style>
```

Рисунок 12 – Пример однофайлового компонента Vue

В данном компоненте использованы все необходимые возможности фреймворка. Участок кода ограниченный тегом template отвечает за представление компонента, тег script ограничивает код обработки компонента,

тег style представлен в формате препроцессора SASS и ограничен данным компонентов атрибутом scoped.

Особыми переменными представленными в примере являются переменная $router и переменная $store. Данные компоненты являются глобальными для всего приложения и подключаются к нему при инициализации. Компонент Vue-router представленный переменной $router реализует маршрутизацию приложения позволяя тем самым отображать только необходимые участки кода без перезагрузки страницы. Данный компонент критичен для реализации формата SPA.

Пример использования маршрутизации в приложении приведен на рисунке 13.

```
{
  path: '/auth',
  component: LayoutAuth,
  meta: {
    title: 'Login',
  },
  redirect: '/auth/login',
  hidden: true,
  children: [
    {
      path: 'login',
      name: 'login',
      meta: {
        title: 'Login',
      },
      component: () => import('@/views/auth/Login.vue'),
    },
  ],
},
```

Рисунок 13 – Пример маршрута Vue-router

«Для реализации структуры API, соответствующего спроектированным конечным точка, серверной части был использован избранный язык программирования PHP. Одни из главных принципов интерфейсостроение

выбранной в данной работе архитектуры реализации API является единство интерфейса запроса и ответа. Таким образом ресурс однозначно идентифицирует действие по единому URL-адресу запроса. При этом для разграничения функционала обязательно наличие в запросе спецификации метода (DELETE,PUT,POST,GET). При этом серверная часть не сохраняет состояние рабочих данных и не производит никаких действий над ними, все управление происходит на клиентской стороне. В запросе допускается передача дополнительных данных для дополнительной спецификации запроса. В заголовке ответа в свою очередь необходимо наличие кода-статуса, обозначающего успешность запроса или специфику его ошибки. В контексте данной работы форматом передачи данных является JSON» [2].

Предварительно разработке программного кода создана и заполнена база данных в соответствии со спроектированной UML диаграммой представленной на рисунке 14.

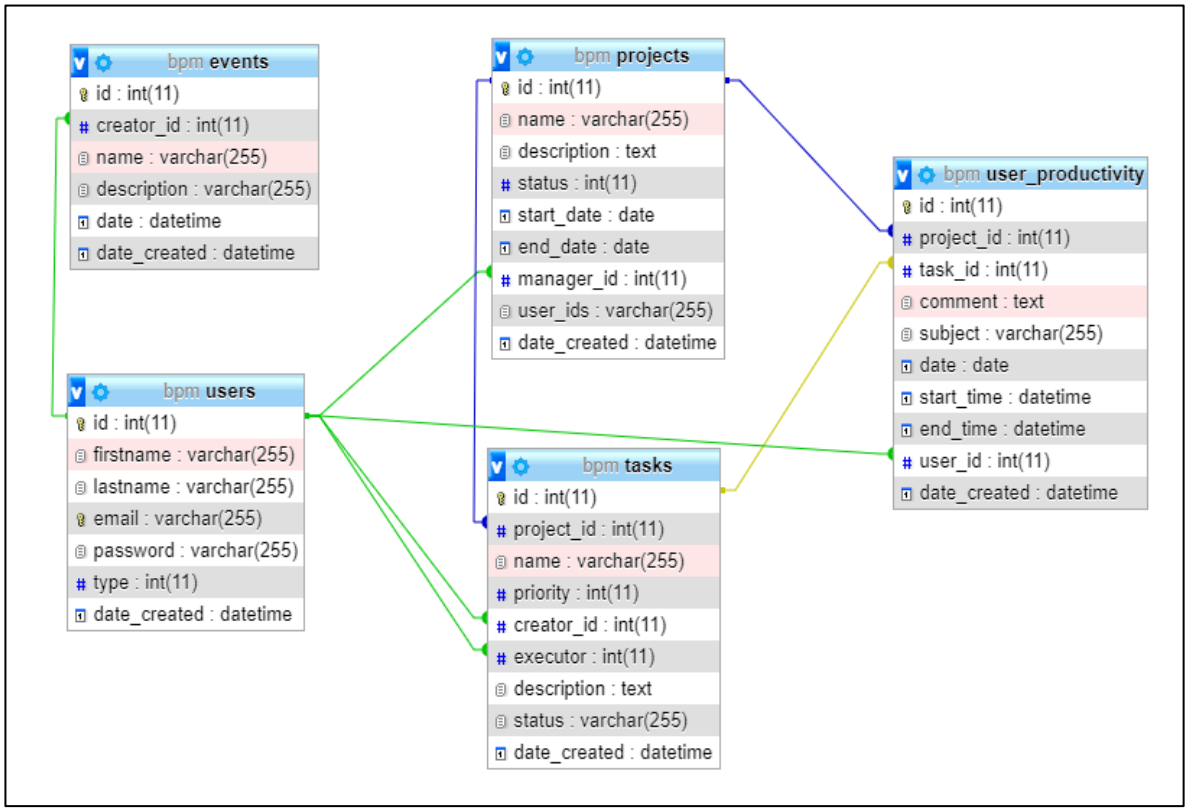

Рисунок 14 – Представление базы данных в веб-интерфейса phpMyAdmin

Для работы с базой данных был создан класс Database отвечающий за установление соединения с базой данных и хранящий данный для подключения (рисунок 15).

```php
<?php
class Database {

    private $host = "localhost";
    private $db_name = "bpm_db";
    private $username = "admin";
    private $password = "";
    public $conn;

    public function getConnection(){

        $this->conn = null;

        try {
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name,
            $this->username, $this->password);
            $this->conn->exec("set names utf8");
        } catch(PDOException $exception){
            echo "Connection error: " . $exception->getMessage();
        }

        return $this->conn;
    }
}
?>
```

Рисунок 15 – Пример реализации класса базы данных

4.2 Тестирование программного продукта

Тестирование программного обеспечения — процесс оценки функциональности программного обеспечения с целью выяснения соответствия работы функционала представленным требованиям и выявление недостатков для оценки и дальнейшего повышения качества конечного продукта.

На этапе разработки модуля ручное тестирование предоставит достаточное покрытие функционала системы, помимо этого позволим продемонстрировать систему в контексте данной работы.

Зная маршруты страниц системы, или при переходе из закладки пользователь может попасть на страницу в обход страницы авторизации. Для того чтобы этого не допустить реализована проверка токена авторизации пользователя, при его отсутствии, попытка доступа к любой странице должна привести к перенаправлению на страницу авторизации (рисунок 16, 17).

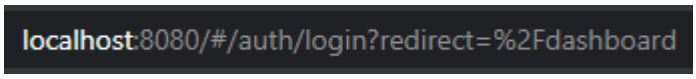

Рисунок 16 –.Результирующая ссылка перенаправления попытки доступа к странице dashboard

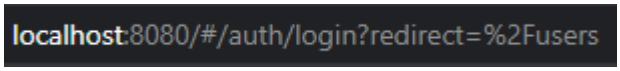

Рисунок 17 – Результирующая ссылка перенаправления попытки доступа к странице users

Исходя из результатов, представленных на рисунке 18 и рисунке 19 может сделать вывод о работе данной функциональной части приложения на представленных тестах.

При проверке форм приложения стоит обращать внимание на наличие обязательных полей, правильность формата ввода данных, правильность самих данных в случае возможности проверки (рисунок 19, 20).

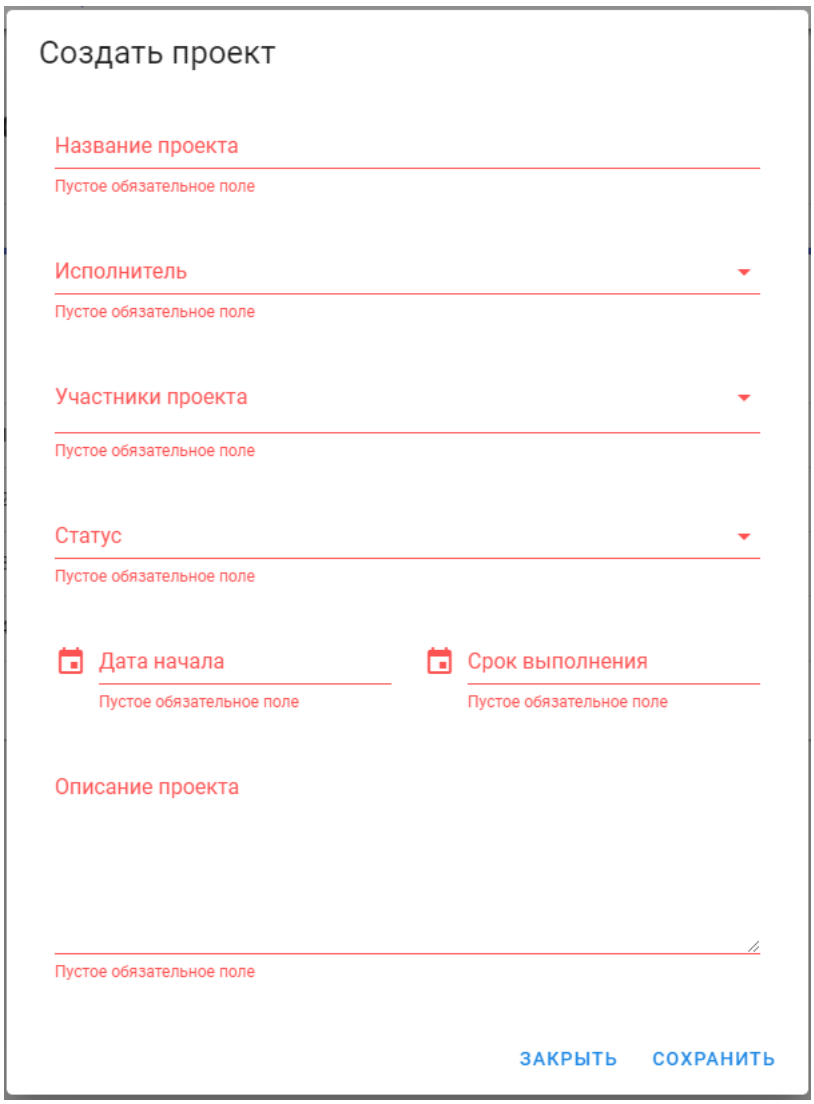

Рисунок 18 – Результат попытки создания проекта с пустым обязательными полями

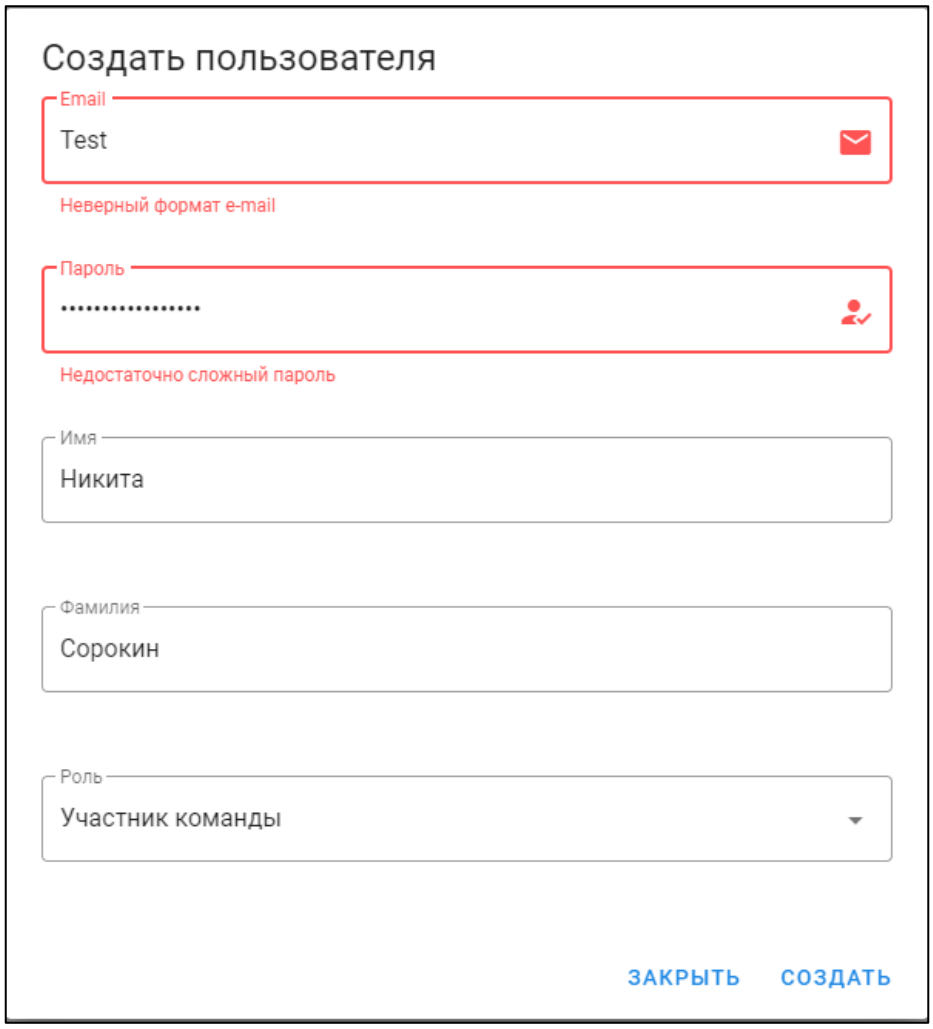

Рисунок 19 – Попытка отправки неправильного формата данных

В случаях редактированиях данных сущностей используется одинаковый компонент формы, следовательно, и правила проверки, в связи с этим результаты тестирования редактирования сущностей опущены. В большей части форм приложения используются поля выборки (рисунок 20) из существующих данных, это решение позволяет минимизировать количество ошибок ввода оставляя пользователя только поля, к которым ограничения содержания не применяется

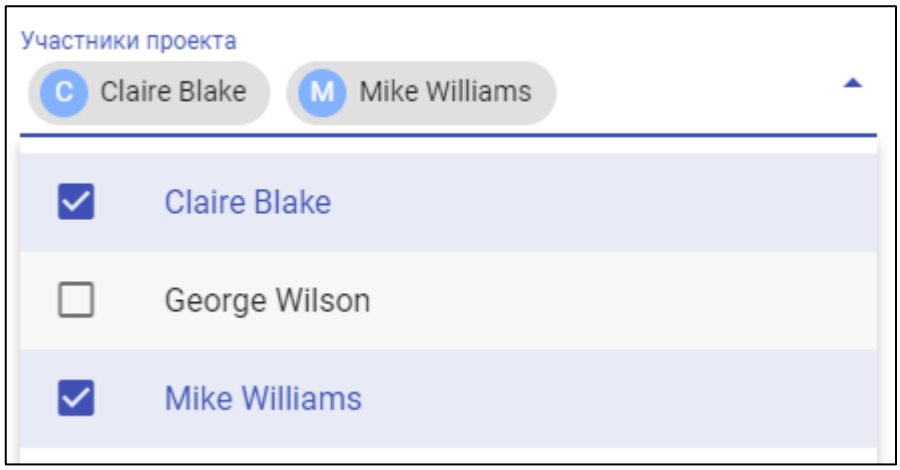

Рисунок 20 – Пример поля выбора подготовленных данных

Исходя из результатов тестирования можем сделать вывод о корректной работе протестированных форм представленных в приложении.

4.3 Результаты работы программного приложения для оценки пользовательских интерфейсов

«При анализе результатов работы приложения рассмотрим работу приложения в корректном отображении всех доступных страниц приложена, а так же реакцию компонентов на изменения данных» [3]. На рисунках 21 и 22 изображены внешний вид главной страницы и страницы списка проектов.

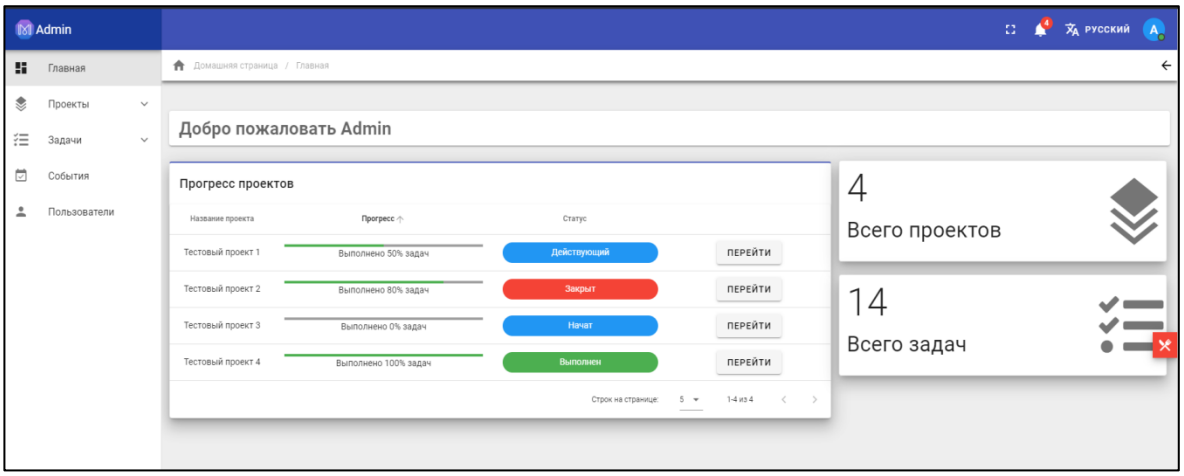

Рисунок 21 – Внешний вид главной страницы.

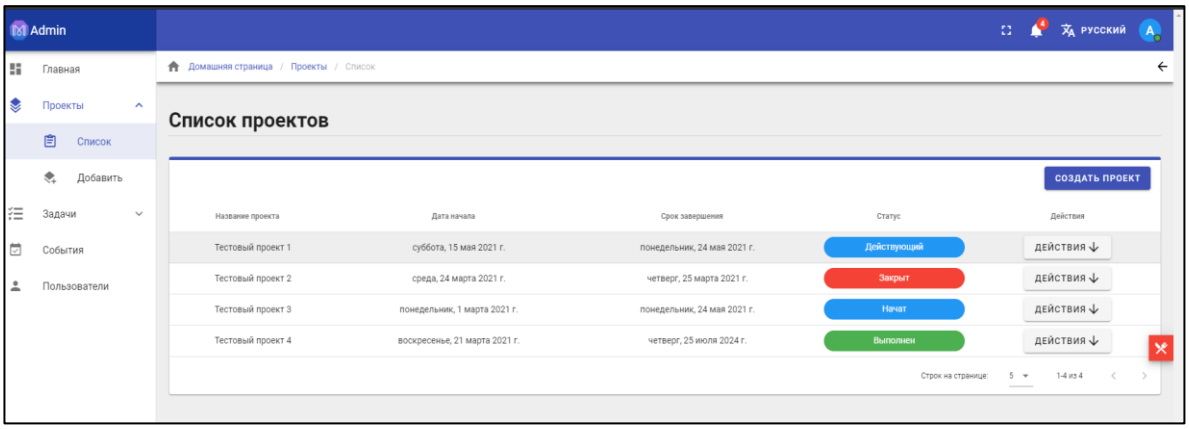

Рисунок 22 – Внешний вид страницы списка проектов

Во избежание случайного удаления данных в приложении реализовано подтверждение запроса на удаление данных (рисунок 23).

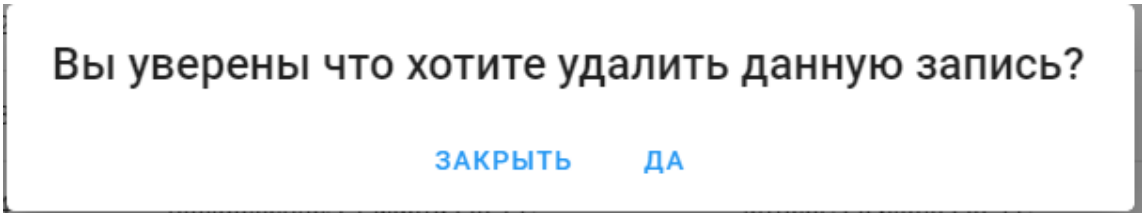

Рисунок 23 – Внешний вид страницы списка проектов

В приложении также реализован модуль подробной информации о проекте, который содержит в себе подробные данные о различных проектах для ознакомления с ними (рисунок 24).

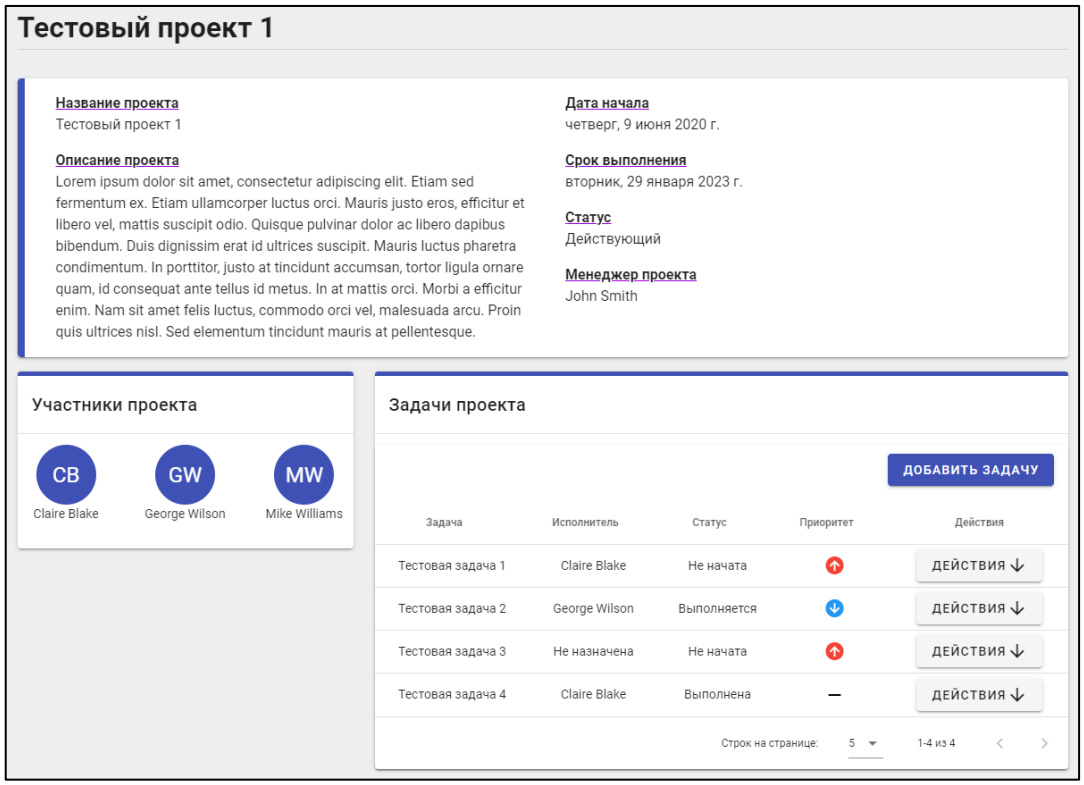

Рисунок 24 – Страница подробной информации о проекте

В приложении также разработан функционал, выполняющий роль канбан-доски (рисунок 25).

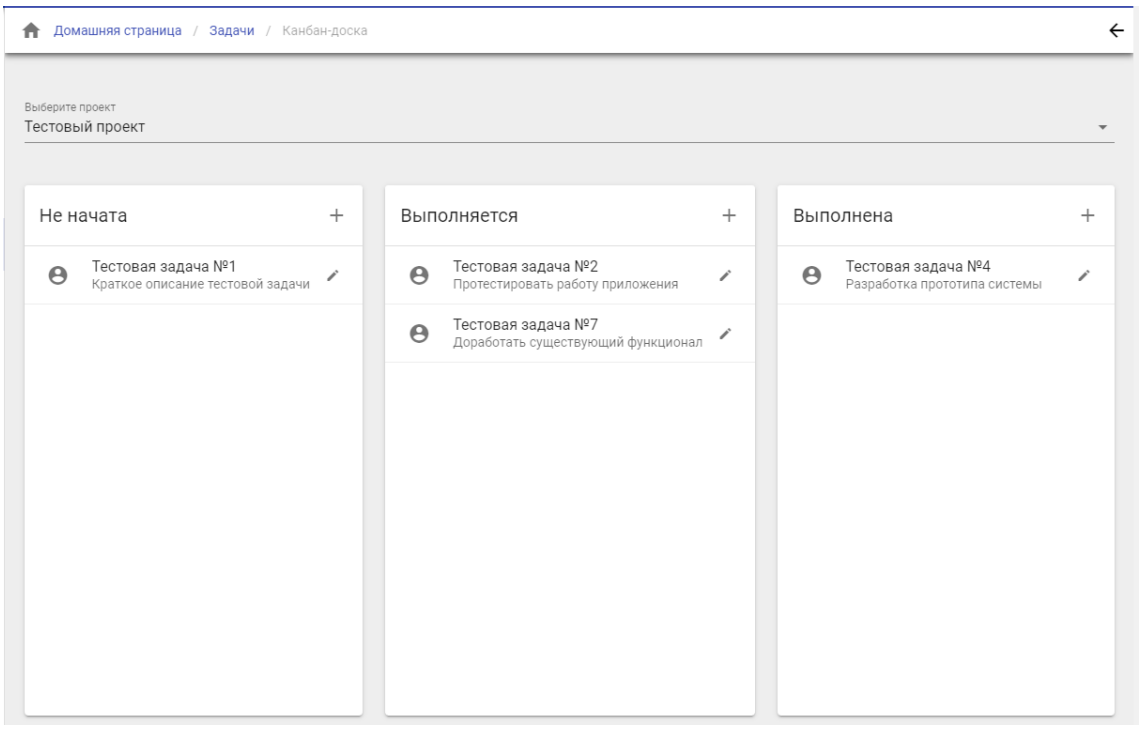

Рисунок 25 – Страница отображения задач в виде канбан-доски

В приведенных выше результатах задействованы все уникальные элементы интерфейса, отображения на других страницах используют те же компоненты. В следствии чего можно сделать вывод о корректном отображении каждого из реализованных компонентов

После реализации каждой задачи, разработчик оценивает свой проект. Результаты работы алгоритма представляются в виде таблиц экспертных оценок и графиков, полученных результатов (рисунок 26, 27).

	G	E	A	H	C	L
User 1	60	52	59	**69**	47	32
User 2	57	48	53	62	41	61
User 3	25	19	46	35	22	33
User 4	17	14	28	11	26	23
User 5	61	63	55	44	**60**	64
User 6	24	23	23	36	22	14
User 7	53	62	44	..	..	..
User ..	..	..	..	..	..	..

Рисунок 26 – Таблица экспертной оценки

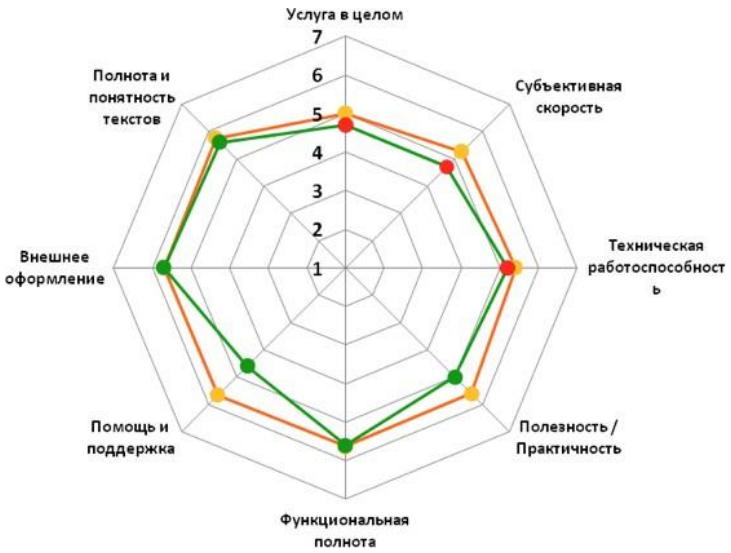

Рисунок 27 – Оценка характеристик в виде графика

Результаты работы по анализу интерфейсов в виде таблицы и графика могут отображаться на экране или быть сохранены в файл (рисунок 28).

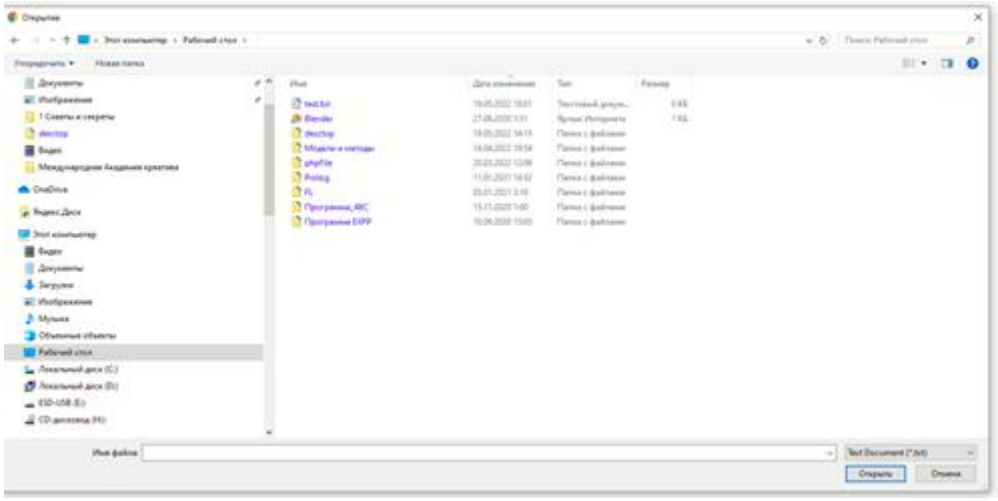

Рисунок 28 – Сохранение файла результатов

Таким образом разработанная система имеет простой интерфейс, в котором невозможно запутаться, и разработчики легко может получить статистику по своему интерефейсу.

В системе предусмотрена возможность работы и оценки каждого проекта и каждой задачи, над которой работает команда. Сбор качественных данных об

юзабилити интерфейса каждого участника проекта позволяет собрать статистику и вовремя реагировать руководителю проекта на оценки сотрудников.

В рамках проведения эксперимента было проанализировано создание двух одинаковых программных продуктов четырьмя командами разработчиков. Две команды использовали предложенное программное обеспечение, две команды работало без дополнительных программных средств для разработки (рисунок 29, 30, 31).

Рисунок 29 – Оценка эффективности методики – по времени

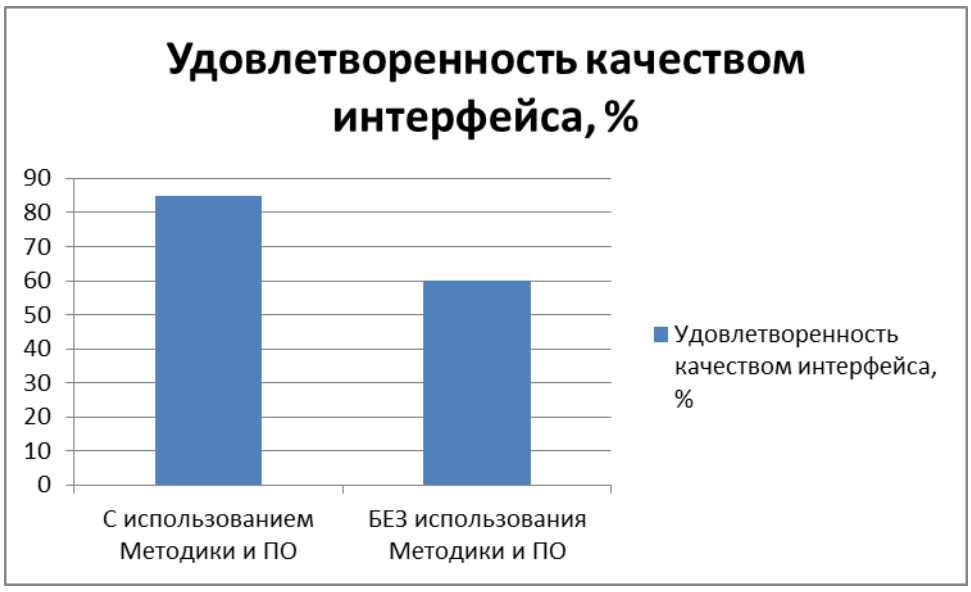

Рисунок 30 – Оценка эффективности методики – удовлетворенность качеством

Рисунок 31 – Оценка эффективности методики – время исправления недостатков

Как видно из приведенных графиков, хоть время разработки с использование предложенной методики увеличивается, но за счет уменьшения времени на доработку и высокими показателями качества, использование предложенной методики можно считать эффективным.

Главное отличие предложенной методики оценки пользовательских интерфейсов от существующих:

– методика предполагает совместную работу над проектом и качеством интерфейсов;

– методика предполагает экспертную оценку интерфейса всех членов команды и преобразование экспертных оценок в количественные оценки;

– методика включает анализ интерфейса по ключевым словам, который проходит в автоматическом режиме.

Вывод по 4 главе

В четвертой главе описана реализация программного приложения для анализа интерфейсов (Приложение А), а также показаны результаты по оценки пользовательских интерфейсов.

Заключение

В ходе выполнения выпускной квалификационной работы выполнена систематизация знаний по разработке пользовательских интерфейсов, обобщении материала по методике оценки качества пользовательских интерфейсов, рассмотрены существующие публикации по теме исследования.

Сделан обзор существующих методик анализа пользовательский интерфейсов, проведен сравнительный анализ существующих методов оценки пользовательских интерфейсов.

Приведена методика оценки пользовательского интерфейса для программ на начальных стадиях разработки, которая основывается на количественной и качественной оценке интерфейсов.

Теоретическая основа диссертационного исследования состоит в систематизации знаний по разработке пользовательских интерфейсов, обобщении материала по методике оценки качества пользовательских интерфейсов. В методологическую основу исследования положен метод экспертной оценки.

Разработан модуль программы для предложенной методики оценки качества пользовательских интерфейсов(Приложение А).

Получены экспериментальные данные по работе методики оценки пользовательского интерфейса для программ на начальных стадиях разработки, которые позволили сказать об ускорении процесса разработки пользовательских интерфейсов и улучшении их качества за счет непрерывной оценки и контроля качества интерфейсов.

Таким образом, данные, полученные в ходе выполнения выпускной квалификационной работы, подтвердили эффективность методики оценки, которая позволяет проанализировать удобство пользовательского интерфейса и произвести качественную и количественную оценку.

Результаты работы докладывались на научно-практических конференциях, публиковались в научных изданиях.

Список используемой литературы

1. Мамаев А.Н. Методы вычленения ключевых слов при оценке пользовательских интерфейсов программ // Студенческий: электрон. научн. журн. 2022. № 8(178). URL: https://sibac.info/journal/student/178/242594 (дата обращения: 21.04.2022).
2. Мамаев А.Н. Разработка web-приложения для оценки пользовательских интерфейсов программ «Студенческий» номер №19(189) 24.05.2022 г. ООО «СибАК»
3. Мамаев А.Н. Методика оценки пользовательских интерфейсов студенческих работ и научных стартапов «Студенческий» номер №20(190) 10.06.2022 г. ООО «СибАК»
4. Tom Tullis, Bill Albert “Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics”, Дата обращения 20.01.2022.
5. Twisted из первых рук : практическое пособие / М. Задка, М. Уильямс, К. Бенфилд [и др.] ; пер. с англ. А. Н. Киселева. - Москва : ДМК Пресс, 2020. - 338 с. - ISBN 978-5-97060-795-4.
6. Vitaly Friedman, 30 Usability Issues To Be Aware Of // URL: http://www.smashingmagazine.com/. Дата обращения 20.01.2022.
7. Баканов, А. С. Эргономический подход к проектированию пользовательского интерфейса распределенной информационной системы / А. С. Баканов // Научный сервис в сети Интернет : труды XVIII Всероссийской научной конференции, Новороссийск, 19–24 сентября 2016 года / ИПМ им. М.В. Келдыша РАН. – Новороссийск: Институт прикладной математики им. М.В. Келдыша РАН, 2016. – С. 72-75.
8. Боброва, И.И. Информационные технологии в образовании : учебно-практический курс / И.И. Боброва, Е.Г. Трофимов. — 3-е изд., стер. — Москва : ФЛИНТА, 2019. - 195 с. - ISBN 978-5-9765-2085-1..

9. Влад Головач «Дизайн пользовательского интерфейса: искусство мыть слона», http://designculture.exmachina.ru/assets/design_culture.pdf Дата обращения 20.01.2022.

10. Гаврилова, И.В. Разработка приложений : учеб. пособие / И.В. Гаврилова. — 3-е изд., стер. — Москва : ФЛИНТА, 2017. — 242 с. - ISBN 978-5-9765-1482-9

11. Гарретт Дж. Веб□дизайн: книга Джесса Гарретта. Элементы опыта взаимодействия». – Пер. с англ. – СПб.: Символ□Плюс, 2008. – 192 с.: ил.

12. Гилязова, Л. М. Пользовательские интерфейсы информационных систем: программная и пользовательская модели интерфейсов / Л. М. Гилязова, Я. Ю. Горбунов, И. М. Яхонтова // Цифровизация экономики: направления, методы, инструменты : Сборник материалов III всероссийской научно-практической конференции, Краснодар, 18–23 января 2021 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2021. – С. 274-276.

13. Горишний Е.Г. Рекомендации по оформлению пользовательского интерфейса для мобильных приложений / Е.Г. Горишний, Е.А. Иванова // сб. ст.: Информационное общество: современное состояние и перспективы развития. Материалы X международного форума. – Краснодар: КубГАУ, 2018. – С. 108-111.

14. Дженнифер Тидвелл. Разработка пользовательских интерфейсов. 4-е изд., доп. —. Питер, 2019. ISBN 978-5-91180-073-4.

15. Jef Raskin «The Humane Interface: New Directions for Designing Interactive Systems (Paperback)», 2000. ISBN 9780201379372

16. Зубкова, Т. М. Модульирование адаптивных пользовательских интерфейсов прикладных программ с использованием методов искусственного интеллекта / Т. М. Зубкова, Л. Ф. Тагирова, В. К. Тагиров // Научно-технический вестник информационных технологий, механики и оптики. – 2019. – Т. 19. – № 4. – С. 680-688.

17. Иванова Е. А. Разработка бизнес-приложений : учеб. пособие / Е. А. Иванова, Н. В. Ефанова, Т. А. Крамаренко. – Краснодар : КубГАУ, 2019. – 118 с.

18. Кирсанов Д. Веб_дизайн: книга Дмитрия Кирсанова. – СПб: Символ_Плюс, 2018. – 368 с.: цв. ил.

19. Козловский, А. В. Экспертные методы в технологии разработки пользовательского интерфейса / А. В. Козловский, С. С. Обрубский, Е. А. Борисова // Инновационные технологии и дидактика в обучении : Сборник статей международной научно-практической конференции, Таганрог, 26–28 июня 2017 года. – Таганрог: Южный федеральный университет, 2017. – С. 129-133.

20. Круг С. Веб-дизайн: книга Стива Круга или «не заставляйте меня думать!» - Пер. с англ. - СПб: Символ-Плюс, 2019. - 200 с: цв. ил.

21. Ликсин, С. С. Сравнительный анализ шаблонов проектирования приложений с пользовательским интерфейсом / С. С. Ликсин, П. А. Лукошкин, С. В. Шибанов // Вестник Пензенского государственного университета. – 2021. – № 4(36). – С. 79-85.

22. Магазанник В.Д. Человеко-компьютерное взаимодействие: учебное пособие. 2-е издание., доп. – М.: Университетская книга, 2020. – 408 с., ил.

23. Мартынов, В. В. Об исследовании качества пользовательских интерфейсов веб-приложений / В. В. Мартынов, А. М. Кузнецов // Научный сервис в сети Интернет : Труды Всероссийской научной конференции , Новороссийск, 22–27 сентября 2013 года / Московский государственный университет им. М.В.Ломоносова, Ростовский государственный университет, Институт вычислительной математики РАН. – Новороссийск: Московский государственный университет им. М.В. Ломоносова (Издательский Дом (Типография), 2013. – С. 178-179.

24. Нильсен Я. Веб-дизайн: книга Якоба Нильсена - Пер. с англ. - СПб: Символ-Плюс, 2018 -512 с.: цв. ил.

25. Стив Круг. Веб-Дизайн: книга Стива Круга или “не заставляйте меня думать!”. 5-е изд., доп. — М.: Символ-Плюс, 2019. ISBN 978-5-93286-099-1.

26. Тиханычев О.В. — Пользовательские интерфейсы в автоматизированных системах: проблемы разработки // Программные системы и вычислительные методы. – 2019. – № 2. – С. 11 - 22.

27. Susan Weinschenk «100 Things Every Designer Needs To Know About People», 2011. ISBN 978-0-3217-6753-0

28. Якоб Нильсен, Хоа Лоранжер. Web-дизайн. Удобство использования Web-сайтов. Вильямс, 2007. ISBN 978-5-8459-1222-0 На основе большого набора юзабилити-тестов предметно, с многочисленными примерами, рассматриваются типичные ошибки в веб-дизайне.

Приложение А

Листинг программы

```
app = FastAPI()

origins = [
    "http://localhost:3000",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


class Data(BaseModel):
    text: str


@app.post('/get_statistics/')
def get_statistics(data: Data):
    key_words = get_keywords(data.text)
    similar_posts = get_similar_posts(key_words)

    text_analyzer = TextAnalyzer(similar_posts=similar_posts)
    text_analyzer.analyze()

    return {
        'list_of_similar': similar_posts,
        'upvotes': text_analyzer.upvotes,
        'downvotes': text_analyzer.downvotes,
        'views': text_analyzer.count_views,
        'key_words': key_words,
        'count_comments': text_analyzer.count_comments
    }
```

keyword_extractor.py

```
import collections
import math
from rutermextract import TermExtractor


def compute_tf(text: list):
    tf_text = collections.Counter(text)
    for i in tf_text:
        tf_text[i] = tf_text[i] / float(len(text))
    return tf_text


def compute_idf(word, corpus):
    return math.log10(len(corpus) / sum([1.0 for i in corpus if word in i]))
```

```
                    def get_keywords(text):
                    list_text = text.split(' ')
                           key_words = []
                  term_extractor = TermExtractor()
                 for term in term_extractor(text):
                   key_words.append(term.normalized)

               key_words_raw = compute_tf(list_text)
               for keyword in key_words_raw.items():
                           word = keyword[0]
                            tf = keyword[1]
                  idf = compute_idf(word, list_text)
                         if (tf * idf) < 0.03:
                           if word in key_words:
                            key_words.remove(word)

                      if len(key_words) == 0:
                         key_words.append(text)
                          return key_words
```

nn_network.py

```
# Создание модели и определение параметров
model = Sequential([
  Dense(64, activation='sigmoid'),
  Dropout(0.5),
  Dense(64, activation='sigmoid'),
  Dropout(0.5),
  Dense(4, activation='softmax'),
])

model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)

# Тренировка модели
model.fit(
  train_texts,
  to_categorical(train_statistics),
  epochs=100,
)

# Сохранение модели
model.save('model.h5')


predictions = model.predict(train_statistics[:5])

print(np.argmax(predictions, axis=1))
```

reddit_connector.py

```
import praw
import random
```

```
def get_similar_posts(keywords):
    reddit = praw.Reddit(
        client_id="***",
        client_secret="***",
        username="vkr_app",
        user_agent="vkr_app",
    )
    posts = reddit.subreddit(all)
    similar_posts = []
    for post in posts.search(keywords[0], limit=None):
        similar_posts.append({
            'link': post.shortlink,
            'title': post.title,
            'count_comments': post.num_comments,
            'upvotes': post.ups,
            'downvotes': post.downs,
            'score': post.score,
            'view_count': view_count,
            'count_reposts': post.num_reports
        })

    return similar_posts



import uvicorn

uvicorn.run(
    'vkr.app:app',
    reload=True
)

Home.jsx
import { useState, useEffect } from 'react';
import { BallTriangle } from 'react-loader-spinner';
import { getStatistics } from '../api';
import { ChartComments } from '../components/ChartComments';
import { ChartCompareComments } from '../components/ChartCompareComments';
import { ChartCompareUpvotes } from '../components/ChartCompareUpvotes';
import { ChartCompareViews } from '../components/ChartCompareViews';
import { Chart, ChartViews } from '../components/ChartViews';
import { KeyWords } from '../components/KeyWords';
import { Statistics, Votes } from '../components/Statistics';


function Home() {

    const [text, setText] = useState('');
    const [keywords, setKeyWords] = useState([]);
    const [isLoading, setIsloading] = useState(false);
    const [analyzed, setAnalyzed] = useState(false);

    const [upvotes, setUpvotes] = useState(null);
    const [downvotes, setDownvotes] = useState(null);
    const [countComments, setCountComments] = useState(null);
    const [views, setViews] = useState(null);
    const [listOfSimilar, setListOfSimilar] = useState([]);
```

```
    const analyseBtnHandler = () => {
        setIsloading(true);
        getStatistics(text)
            .then(response => {
                setIsloading(false);
                const { upvotes, downvotes, key_words: keywords, count_comments:
countComments, views, list_of_similar: listOfSimilar } = response.data;
                setUpvotes(upvotes);
                setDownvotes(downvotes);
                setKeyWords(keywords);
                setCountComments(countComments);
                setAnalyzed(true);
                setViews(views);
                setListOfSimilar(listOfSimilar);
            })
            .catch(() => {
                setIsloading(false);
                alert('Ошибка');
            });
    }

    const style = { position: "fixed", top: "50%", left: "50%", transform:
"translate(-50%, -50%)" };

    return (
        <>
            {isLoading ?
                <>
                    <h2 className='loadingText'>Анализ текста...</h2>
                    <div style={style}>
                        <BallTriangle color="#009688" width={350} height={350}
/>
                    </div>
                </>

                :
                <>
                    <button className="btn"> <i className="large material-
icons">file_download</i> Загрузить из файла</button>
                    <div className="form-group">
                        <label htmlFor="exampleFormControlTextarea1">Введите
текст интерфейса</label>
                        <textarea
                            className="form-control rounded-0"
                            id="exampleFormControlTextarea1"
                            value={text}
                            onChange={e => { setText(e.target.value); }}
                        >
                        </textarea>
                    </div>
                    <button
                        className="btn btnSend"
                        onClick={analyseBtnHandler}
                    >
                        Анализ текста
                    </button>
                    <KeyWords keywords={keywords} />
                    <Statistics
                        upvotes={upvotes}
                        downvotes={downvotes}
                        countComments={countComments}
```

```
                            views={views}
                        listOfSimilar={listOfSimilar}
                    />
                    {analyzed ?
                        <>
                            <ChartCompareComments
                                similarPosts={listOfSimilar}
                                countComments={countComments}
                                count={10}
                            />
                            <ChartCompareUpvotes
                                similarPosts={listOfSimilar}
                                upvotes={upvotes}
                                count={10}
                            />
                            <ChartCompareViews
                                similarPosts={listOfSimilar}
                                views={views}
                                count={20}
                            />
                            <ChartViews />
                            <ChartComments />
                        </>
                        : ''
                    }
                </>
            }
        </>
    );
}
export { Home };

import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareComments(props) {

    const { similarPosts, countComments, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализируемый интерфейс',
        "Анализируемый интерфейс": countComments
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title.substr(0, 20),
            "Похожие кнопки": post.count_comments
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества комментариев с похожими интерфейсами </h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
```

```
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество комментариев
                        </Label>
                    </YAxis>
                    <Tooltip />
                    {/* <Legend verticalAlign="top" height={36}/> */}
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие кнопки" fill="#3949ab" />
                    <Bar dataKey="Анализируемый интерфейс" fill="red" />
                    {/* <Bar dataKey="original" fill="#3949ab" />
                    <Bar dataKey="1" />
                    <Bar dataKey="2"  />
                    <Bar dataKey="3" />
                    <Bar dataKey="4" />
                    <Bar dataKey="5" /> */}
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}

export { ChartCompareComments };
```

App.jsx

```
import { Footer } from "./components/Footer";
import { Header } from "./components/Header";
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import { Home } from "./pages/Home";
import { NotFound } from "./pages/NotFound";


function App() {
    return (
        <>
            <Router>
                <Header />
                <main className="container content">

                    <Switch>
                        <Route exact path="/" component={Home}></Route>
                        <Route component={NotFound}></Route>
                    </Switch>
                </main>
                <Footer />
            </Router>
        </>
    );
}
```

```
export default App;

Footer.jsx
export function Footer() {
    return (
        <footer className="page-footer cyan darken-4">
            <div className="container">
                © {new Date().getFullYear()}
            </div>
        </footer>
    );
}

Header.jsx

import { Link } from "react-router-dom";

export function Header() {
    return (
        <nav className="header">
            <div className="nav-wrapper">
                <Link to="/" className="brand-logo">Анализ пользовательских
интерфейсов студенческих работ</Link>
                <ul id="nav-mobile" className="right hide-on-med-and-down">
                </ul>
            </div>
        </nav>
    );
}
KeyWord.jsx
function KeyWord(props) {
    const {keyword} = props;
    return (
        <div className="keyword">{keyword}</div>
    )
}

export { KeyWord };

keyWords.jsx

import { KeyWord } from "./KeyWord";

function KeyWords(props) {
    const {keywords} = props;
    return (
        <div className='keywords'>
            {keywords.length !== 0 ?
                keywords.map((word) => {
                    return <KeyWord key={word} keyword={word}/>
                })
                : ''
            }
        </div>
    )
}

export { KeyWords };

ChartCompareComments.jsx
```

```
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareComments(props) {

    const { similarPosts, countComments, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализируемый интерфейс',
        "Анализируемый интерфейс": countComments
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title.substr(0, 20),
            "Похожие кнопки": post.count_comments
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества комментариев с похожими интерфейсами </h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество комментариев
                        </Label>
                    </YAxis>
                    <Tooltip />
                    {/* <Legend verticalAlign="top" height={36}/> */}
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие кнопки" fill="#3949ab" />
                    <Bar dataKey="Анализируемый интерфейс" fill="red" />
                    {/* <Bar dataKey="original" fill="#3949ab" />
                    <Bar dataKey="1" />
                    <Bar dataKey="2"  />
                    <Bar dataKey="3" />
                    <Bar dataKey="4" />
                    <Bar dataKey="5" /> */}
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}
```

```
export { ChartCompareComments };

Statistics.jsx

import { Link } from "react-router-dom";

function Statistics(props) {
    const { upvotes, downvotes, countComments, views, listOfSimilar } = props;
    if (upvotes) {
        return <div className="statistics">
            <h5>Предполагаемое количество <i className="small material-
icons">remove_red_eye</i>: <b>{views}</b></h5>
            <h5>Предполагаемое количество комментариев:
<b>{countComments}</b></h5>
            <h5>Предполагаемое количество <i className="small material-
icons">thumb_up</i>: <b className="up">{upvotes}</b></h5>
            <h5>Предполагаемое количество <i className="small material-
icons">thumb_down</i>: <b className="down">{downvotes}</b></h5>
            {listOfSimilar.length > 0 ?
                <>
                    <br></br>
                    <h5>Похожие кнопки:</h5>
                    <div className="similar-posts">
                        <ol>
                            {listOfSimilar.map((el) => {
                                {/* return <li><Link to={el} className="brand-
logo">{el}</Link></li> */}
                                return <li><a href={el.link}
target="_blank">{el.title}</a></li>
                            })}
                        </ol>
                    </div>
                </> : ''
            }

        </div>
    }
    return '';
}

export { Statistics };

TextPost.jsx

function TextPost(props) {
    // const { strCategory, strCategoryThumb, strCategoryDescription } = props;

    return (
        <>
            <div className="form-group">
                <label htmlFor="exampleFormControlTextarea1">Введите текст
интерфейса</label>
                <textarea className="form-control rounded-0"
id="exampleFormControlTextarea1" rows="5"></textarea>
            </div>
            <button className="btn btnSend">Анализ текста</button>
        </>
    )
}
```

```
export { TextPost };

import { API_URL } from "./config";
import axios from "axios";

const getStatistics = async(text) => {
    const response = await axios.post(API_URL + 'get_statistics/', { "text":
text });
    return response;
}

export { getStatistics };


import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareViews(props) {

    const { similarPosts, views, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализируемый интерфейс',
        "Анализируемый интерфейс": views
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title,
            "Похожие кнопки": post.view_count
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества просмотров с похожими интерфейсами </h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество просмотров
                        </Label>
                    </YAxis>
                    <Tooltip />
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие кнопки" fill="#3949ab" />
```

```
                <Bar dataKey="Анализируемый интерфейс" fill="red" />
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}

export { ChartCompareViews };


import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareUpvotes(props) {

    const { similarPosts, upvotes, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализируемый интерфейс',
        "Анализируемый интерфейс": upvotes
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title.substr(0, 20),
            "Похожие кнопки": post.upvotes
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества использования с похожими интерфейсами </h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество использования
                        </Label>
                    </YAxis>
                    <Tooltip />
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие кнопки" fill="#3949ab" />
                    <Bar dataKey="Анализируемый интерфейс" fill="red" />
                </BarChart>
            </ResponsiveContainer>
        </div>
```

```
    )
}

export { ChartCompareUpvotes };

App.css
.App {
    text-align: center;
}
function HomeRadio() {

    const [text, setText] = useState('');
    const [keywords, setKeyWords] = useState([]);
    const [isLoading, setIsloading] = useState(false);
    const [analyzed, setAnalyzed] = useState(false);

    const [upvotes, setUpvotes] = useState(null);
    const [downvotes, setDownvotes] = useState(null);
    const [countComments, setCountComments] = useState(null);
    const [views, setViews] = useState(null);
    const [listOfSimilar, setListOfSimilar] = useState([]);

    const analyseBtnHandler = () => {
        setIsloading(true);
        getStatistics(text)
            .then(response => {
                setIsloading(false);
                const { upvotes, downvotes, key_words: keywords, count_comments:
countComments, views, list_of_similar: listOfSimilar } = response.data;
                setUpvotes(upvotes);
                setDownvotes(downvotes);
                setKeyWords(keywords);
                setCountComments(countComments);
                setAnalyzed(true);
                setViews(views);
                setListOfSimilar(listOfSimilar);
            })
            .catch(() => {
                setIsloading(false);
                alert('Ошибка анализа данного компонента');
            });
    }

    const style = { position: "fixed", top: "50%", left: "50%", transform:
"translate(-50%, -50%)" };

    return (
        <>
            {isLoading ?
                <>
                    <h2 className='loadingText'>Анализ текста...</h2>
                    <div style={style}>
                        <BallTriangle color="#009688" width={350} height={350}
/>
                    </div>
                </>

                :
                <>
                    <button className="btn"> <i className="large material-
icons">file_download</i> Загрузить из файла</button>
```

```
                    <div className="form-group">
                        <label htmlFor="exampleFormControlTextarea1">Введите
текст интерфейса</label>
                        <textarea
                            className="form-control rounded-0"
                            id="exampleFormControlTextarea1"
                            value={text}
                            onChange={e => { setText(e.target.value); }}
                        >
                        </textarea>
                    </div>
                    <button
                        className="btn btnSend"
                        onClick={analyseBtnHandler}
                    >
                        Анализ текста
                    </button>
                    <KeyWords keywords={keywords} />
                    <Statistics
                        upvotes={upvotes}
                        downvotes={downvotes}
                        countComments={countComments}
                        views={views}
                        listOfSimilar={listOfSimilar}
                    />
                    {analyzed ?
                        <>
                            <ChartCompareComments
                                similarPosts={listOfSimilar}
                                countComments={countComments}
                                count={10}
                            />
                            <ChartCompareUpvotes
                                similarPosts={listOfSimilar}
                                upvotes={upvotes}
                                count={10}
                            />
                            <ChartCompareViews
                                similarPosts={listOfSimilar}
                                views={views}
                                count={20}
                            />
                            <ChartViews />
                            <ChartComments />
                        </>
                        : ''
                    }
                </>
            }
        </>
    );
}
export { Home };

import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareComments(props) {

    const { similarPosts, countComments, count = 5 } = props;
```

```
    let initialData = [];
    initialData.push({
        name: 'Анализ компонента интерфейса - переключатель',
        "Переключатель ": countComments
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title.substr(0, 20),
            "Похожие кнопки": post.count_comments
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества, используемых переключателей</h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Используемые переключатели, в других работах
                        </Label>
                    </YAxis>
                    <Tooltip />
                    {/* <Legend verticalAlign="top" height={36}/> */}
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие переключатели" fill="#3949ab" />
                    <Bar dataKey="'Анализ компонента интерфейса - переключатель
" fill="red" />
                    {/* <Bar dataKey="original" fill="#3949ab" />
                    <Bar dataKey="1" />
                    <Bar dataKey="2"  />
                    <Bar dataKey="3" />
                    <Bar dataKey="4" />
                    <Bar dataKey="5" /> */}
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}

export { ChartCompareComments };
```

App.jsx

```
import { Footer } from "./components/Footer";
import { Header } from "./components/Header";
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
```

```
import { Home } from "./pages/HomeRadioButton";
import { NotFound } from "./pages/NotFound";


function App() {
    return (
        <>
            <Router>
                <Header />
                <main className="container content">

                    <Switch>
                        <Route exact path="/" component={Home}></Route>
                        <Route component={NotFound}></Route>
                    </Switch>
                </main>
                <Footer />
            </Router>
        </>
    );
}

export default App;

Footer.jsx
export function Footer() {
    return (
        <footer className="page-footer cyan darken-4">
            <div className="container">
                © {new Date().getFullYear()}
            </div>
        </footer>
    );
}

Header.jsx

import { Link } from "react-router-dom";

export function Header() {
    return (
        <nav className="header">
            <div className="nav-wrapper">
                <Link to="/" className="brand-logo">Анализ пользовательских
интерфейсов студенческих работ</Link>
                <ul id="nav-mobile" className="right hide-on-med-and-down">
                </ul>
            </div>
        </nav>
    );
}
KeyWord.jsx
function KeyWord(props) {
    const {keyword} = props;
    return (
        <div className="keyword">{keyword}</div>
    )
}

export { KeyWord };
```

keyWords.jsx

```
import { KeyWord } from "./KeyWord";

function KeyWords(props) {
    const {keywords} = props;
    return (
        <div className='keywords'>
            {keywords.length !== 0 ?
                keywords.map((word) => {
                    return <KeyWord key={word} keyword={word}/>
                })
                : ''
            }
        </div>
    )
}

export { KeyWords };
```

ChartCompareComments.jsx

```
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareComments(props) {

    const { similarPosts, countComments, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализ компонента интерфейса - переключатель ',
        "'Анализ компонента интерфейса - переключатель ": countComments
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title.substr(0, 20),
            "Похожие переключатели": post.count_comments
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение интерфейсов, где используют переключатели </h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
```

```
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество переключателей
                        </Label>
                    </YAxis>
                    <Tooltip />
                    {/* <Legend verticalAlign="top" height={36}/> */}
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие интерфесы с переключателями"
fill="#3949ab" />
                    <Bar dataKey="Анализируемый интерфейс" fill="red" />
                    {/* <Bar dataKey="original" fill="#3949ab" />
                    <Bar dataKey="1" />
                    <Bar dataKey="2"  />
                    <Bar dataKey="3" />
                    <Bar dataKey="4" />
                    <Bar dataKey="5" /> */}
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}

export { ChartCompareComments };
```

Statistics.jsx

```
import { Link } from "react-router-dom";

function Statistics(props) {
    const { upvotes, downvotes, countComments, views, listOfSimilar } = props;
    if (upvotes) {
        return <div className="statistics">
            <h5>Предполагаемое количество <i className="small material-
icons">remove_red_eye</i>: <b>{views}</b></h5>
            <h5>Предполагаемое количество переключателей:
<b>{countComments}</b></h5>
            <h5>Предполагаемое количество переключателей <i className="small
material-icons">thumb_up</i>: <b className="up">{upvotes}</b></h5>
            <h5>Предполагаемое количество переключателей <i className="small
material-icons">thumb_down</i>: <b className="down">{downvotes}</b></h5>
            {listOfSimilar.length > 0 ?
                <>
                    <br></br>
                    <h5>Похожие кнопки:</h5>
                    <div className="similar-posts">
                        <ol>
                            {listOfSimilar.map((el) => {
                                {/* return <li><Link to={el} className="brand-
logo">{el}</Link></li> */}
                                return <li><a href={el.link}
target="_blank">{el.title}</a></li>
                            })}
                        </ol>
                    </div>
                </> : ''
            }

        </div>
    }
```

```
    return '';
}

export { Statistics };

TextPost.jsx

function TextPost(props) {
    // const { strCategory, strCategoryThumb, strCategoryDescription } = props;

    return (
        <>
            <div className="form-group">
                <label htmlFor="exampleFormControlTextarea1">Введите
радиокнопку</label>
                <textarea className="form-control rounded-0"
id="exampleFormControlTextarea1" rows="5"></textarea>
            </div>
            <button className="btn btnSend">Анализ радиокнопки</button>
        </>
    )
}

export { TextPost };

import { API_URL } from "./config";
import axios from "axios";

const getStatistics = async(text) => {
    const response = await axios.post(API_URL + 'get_statistics/', { "text":
text });
    return response;
}

export { getStatistics };


import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareViews(props) {

    const { similarPosts, views, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализируемая часть интерфейса - радиокнопка',
        "Анализируемый часть интерфейса - радиокнопка ": views
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title,
            "Похожие радиокнопки": post.view_count
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества интерфейсов с радиокнопками</h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
```

```
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество интерфейсов в базе
                        </Label>
                    </YAxis>
                    <Tooltip />
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие интерфейсы с радиокнопками "
fill="#3949ab" />
                    <Bar dataKey="Анализируемый интерфейс" fill="red" />
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}

export { ChartCompareViews };


import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer, Label } from 'recharts';

function ChartCompareUpvotes(props) {

    const { similarPosts, upvotes, count = 5 } = props;

    let initialData = [];
    initialData.push({
        name: 'Анализируемый интерфейс',
        "Анализируемый интерфейс": upvotes
    });
    similarPosts.slice(0, count).forEach(post => {
        initialData.push({
            name: post.title.substr(0, 20),
            "Похожие кнопки": post.upvotes
        });
    });

    return (
        <div className='charts'>
            <h1>Сравнение количества использования с похожими интерфейсами </h1>
            <ResponsiveContainer width="100%" aspect={3}>
                <BarChart
                    width={500}
                    height={300}
                    data={initialData}
                    margin={{
```

```
                        top: 5,
                        right: 30,
                        left: 20,
                        bottom: 5,
                    }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="name" hide={true}>
                    </XAxis>
                    <YAxis>
                        <Label angle={270} position='left' style={{ textAnchor:
'middle' }}>
                            Количество использования
                        </Label>
                    </YAxis>
                    <Tooltip />
                    <Legend verticalAlign="top" height={36} />
                    <Bar dataKey="Похожие интерфейсы с радиокнопками "
fill="#3949ab" />
                    <Bar dataKey="Анализируемый интерфейс с радиокнопками "
fill="red" />
                </BarChart>
            </ResponsiveContainer>
        </div>
    )
}

export { ChartCompareUpvotes };
```

App.css

```
.App {
    text-align: center;
}
```