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

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 «Математическое обеспечение и администрирование информационных систем»
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для управления процессом оценки сотрудников»

Обучающийся Д.Р Шарафутдинов
(И.О. Фамилия) (личная подпись)

Руководитель к.э.н., доцент, Т.А. Раченко
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент С.А. Гудкова

Тольятти 2024

Аннотация

Тема бакалаврской работы «Разработка мобильного приложения для управления процессом оценки сотрудников».

Цель бакалаврской работы – разработка мобильного приложения для оценки сотрудников. В рамках ВКР будет создан эффективный инструмент, который позволит повысить объективность и прозрачность процесса оценки персонала, автоматизировать сбор и анализ данных, предоставить руководителям и HR-специалистам удобные средства для управления оценкой и развития сотрудников, а также обеспечить своевременную обратную связь, способствующую профессиональному росту персонала.

Объектом исследования является процесс оценивания сотрудников на предприятии, и соответствующие методики для этого.

Предметом исследования будет являться разработка мобильного приложения для автоматизации процесса оценки эффективности и профессионального развития сотрудников организации.

Структура работы представлена введением, 3 главами, заключением, списком литературы и приложением.

В первой главе представлены описание предметной области, цель и задачи разработки мобильного приложения.

Во второй главе идет проектирование мобильного приложения.

В третьей главе идет реализация и тестирование мобильного приложения.

Работа состоит из 71 страниц, 56 рисунков, 2 таблицы, 18 листингов, 2 приложений и списка из 24 источников.

Abstract

Bachelor Thesis Theme: «Development of a mobile application for managing the process of employee evaluation»

The aim of the bachelor thesis is to develop a mobile application for employee assessment. Within the framework of the Bachelor's work an effective tool will be created, which will allow to increase objectivity and transparency of the personnel evaluation process, automate data collection and analysis, provide managers and HR-specialists with convenient means to manage the evaluation and development of employees, as well as provide timely feedback, contributing to the professional growth of personnel.

The object of the study is the process of employee evaluation at the enterprise, and the relevant techniques for this.

The subject of the research is a convenient mobile application using Android Studio environment and SQLite database, which is being developed to simplify and improve the process of employee competence assessment at the enterprise.

The structure of the work includes an introduction, three chapters, a conclusion, a list of references, and an appendix.

The first chapter presents a description of the subject area, the goal, and the objectives of developing the mobile application.

The second chapter covers the design of the mobile application.

The third chapter deals with the implementation and testing of the mobile application.

A mobile application and server have been developed, with the ability to interact mutually for accounting energy resources and calculating forecasted charges.

The work consists of 71 pages of text, 56 figures, 2 tables, 2 appendices, and a list of 24 references.

Оглавление

Введение

Большинство компаний сталкивается с необходимостью оценивания персонала. Можно задать определённые KPI и отслеживать их выполнение. Так задача сводится к простому сбору и анализу цифр. А что, если работа специалиста не позволяет получить чёткие цифровые метрики? Или — и это бывает ещё чаще — формально специалист выполняет все требования, а вот обедает всё время один, и все остальные сотрудники его сторонятся? Здесь уже важны не показатели, а коммуникативные навыки. Оценить их могут специализированные системы оценивания персонала.

В настоящее время широкое распространение получили различные методики оценивания сотрудников на предприятии. Подобные опросы позволяют сотрудникам проходить оценку и самооценку по различным критериям, а руководителям – просматривать и агрегировать результаты тестирования.

Обычно подобные опросы проводятся стандартным бумажным способом, но на рынке доступно несколько решений по проведению оценивания онлайн. В то же время на рынке представлено достаточно мало функциональных и удобных мобильных приложений для проведения подобных опросов.

Тема бакалаврской работы «Разработка мобильного приложения для управления процессом оценки сотрудников».

Цель бакалаврской работы – разработка мобильного приложения для оценки сотрудников. В рамках ВКР будет создан эффективный инструмент, который повысит объективность и прозрачность процесса оценки персонала, автоматизирует сбор и анализ данных, предоставит руководителям и HR-специалистам удобные средства для управления оценкой и развитием сотрудников, а также обеспечит своевременную обратную связь, способствующую профессиональному росту персонала.

Объектом исследования является процесс оценивания сотрудников на предприятии, и соответствующие методики для этого.

Предметом исследования будет являться разработка мобильного приложения для автоматизации процесса оценки эффективности и профессионального развития сотрудников организации.

Задачи, необходимые для достижения цели бакалаврской работы – следующие:

- провести описание предметной области и предприятия, для которого разрабатывается система;
- дать общую характеристику процесса оценивания сотрудников и выбрать подходящую для реализации методику;
- выполнить постановку задачи на разработку системы, провести идентификацию требований пользователей и определение основных функциональных и нефункциональных требований;
- провести проектирование архитектуры мобильного приложения, определив основные модули и взаимосвязи между ними, а также взаимодействие с базой данных через соответствующий API;
- провести разработку схемы данных приложения;
- реализовать пользовательский интерфейс и основные модули мобильного приложения;
- провести тестирование разработанного решения.

В первой главе представлено описание предметной области, дается общая характеристика предприятия, проводится анализ требований к информационной системе и сбор данных.

Во второй главе проводится проектирование архитектуры мобильного приложения, определение основных компонентов и модулей, разработка базы данных и непосредственно функциональности приложения.

В третьей главе проводится интеграция и тестирование разрабатываемого приложения.

Глава 1 Аналитическая часть

1.1 Описание предметной области

ООО “Квартплата 24” – ИТ – компания, которая занимается разработкой сервисов, предоставляемых управляющим компаниям в сфере жилищно-коммунальных услуг. Компания была основана в 1996 году и на данный момент обслуживает более 1 миллиона лицевых счетов по всей России, сотрудничая с более чем 1000 ТСЖ (товариществ собственников жилья), управляющими организациями (УО) и ресурсоснабжающими организациями (РСО) в 70 субъектах РФ.

«Компания решает такие задачи, как расчет и учет платы за жилищно-коммунальные услуги, прием и распределение платежей, а также взыскание долгов в соответствии с законодательством для товариществ собственников жилья, управляющих и ресурсоснабжающих организаций, информационно-расчетных центров по всей России» [1]. Основная миссия организации «ООО Квартплата 24» заключается в создании удобной и надежной платформы, которая обеспечивает эффективность и удобство управления жилищно-коммунальными услугами через онлайн-сервисы.

Далее рассмотрим организационную структуру предприятия, представленную на рисунке 1.

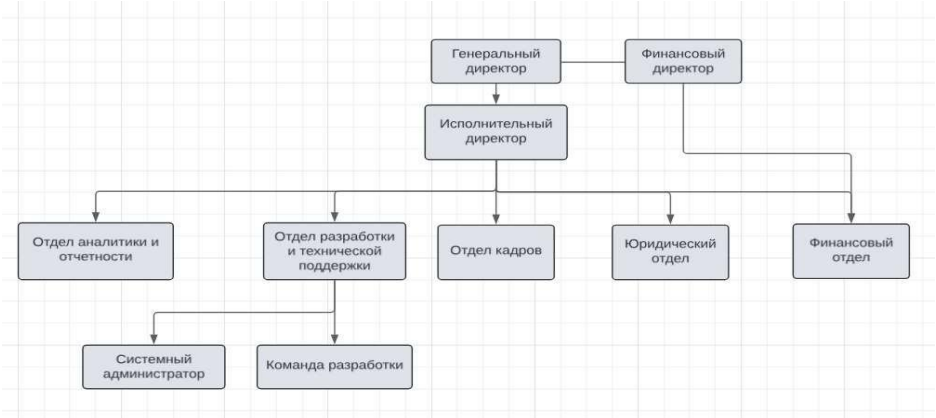

Рисунок 1- Организационная структура компании «Квартплата 24»

Структура состоит из четырех уровней управления, с помощью которого обеспечивается высокий уровень предоставляемых услуг. Первый уровень состоит из генерального директора, который ответственен за стратегическое руководство, принимает ключевые решения и определяет общую стратегию развития компании.

На следующем уровне находится исполнительный директор, который координирует повседневные операции компании и следит за выполнением стратегических целей и задач, поставленных генеральным директором.

На третьем уровне структуры компании расположены ключевые отделы: разработка и техническая поддержка, аналитика и отчетность, юридический отдел и отдел кадров. Отдел разработки и технической поддержки занимается созданием и поддержкой программного обеспечения, а также оказывает техническую помощь клиентам. Отдел аналитики и отчетности анализирует данные, готовит отчеты для руководства и клиентов, выявляет тенденции и предлагает улучшения.

На четвертом уровне работают специалисты и сотрудники, которые выполняют свои задачи в зависимости от того, к какому отделу они относятся.

Офис компании оснащен всем необходимым для обеспечения эффективной работы и предоставления высококачественных услуг:

- компьютеры с установленным и настроенным программным обеспечением для работы с платежными системами, базами данных и расчетами;
- сетевое оборудование;
- принтеры;
- рабочие телефоны.

Таким образом, успешная деятельность компании достигается за счет использования передового оборудования, четко налаженной организационной структуры и профессионализма квалифицированных сотрудников. Это позволяет компании эффективно выполнять свои задачи и предоставлять клиентам высококачественные услуги в области жилищно-коммунального хозяйства.

1.2 Характеристика процесса оценки сотрудников

«В данном подразделе описывается методика оценивания сотрудников «360 градусов», которая будет использоваться в реализуемом приложении, ее основные особенности и этапы разработки» [2].

Различные методики оценивания сотрудников на предприятии могут использоваться для решения задач управления мотивацией и другими качествами сотрудников.

Кроме «360 градусов», широко используются такие методы, как прямая оценка руководителем, ассессмент-центры и анализ результатов работы.

- прямая оценка руководителем - традиционный и наиболее распространенный подход, при котором руководитель напрямую оценивает выполнение сотрудником его рабочих обязанностей. Основное преимущество - простота и оперативность. Однако этот метод может быть субъективным и не всегда отражает полную картину взаимодействия сотрудника с коллегами и клиентами;
- ассессмент-центры - комплексное мероприятие, включающее ряд заданий, имитирующих различные рабочие ситуации. Преимущества включают объективность и всестороннюю оценку компетенций сотрудника. ;едостатки - высокая стоимость и организационная сложность.
- анализ результатов работы - метод, основанный на количественных показателях результативности сотрудника. Это обеспечивает ясность и измеримость, но не всегда показывает полезность мягких навыков, таких как коммуникативные способности или умение работать в команде.

Каждый из этих методов имеет свои место и может быть эффективен в определенных условиях, но метод «360 градусов» предлагает более сбалансированный и комплексный подход к оценке, что делает его предпочтительным выбором для многих современных компаний.

Алгоритм оценки сотрудников по методике:

Определение критериев оценки:

- администратор системы определяет перечень критериев для оценки сотрудников;
- критерии могут включать профессиональные навыки, личностные качества, результативность и т.д.;
- для каждого критерия задаются показатели и шкала оценки;

Выставление оценок по критериям:

- руководитель или назначенный оценщик выставляет оценки сотруднику по каждому критерию
- оценки выставляются по заданной шкале (например, от 0 до 4 баллов)
- оценщик может оставить комментарии по каждому критерию

Расчет итоговой оценки:

- система суммирует баллы, полученные сотрудником по всем критериям
- рассчитывается средний балл как общая сумма баллов, деленная на количество критериев
- итоговая оценка может быть представлена в виде числового значения или с использованием качественных градаций (низкий, средний, высокий уровень и т.д.).

Анализ результатов оценки:

- руководитель или HR-специалист анализирует полученные оценки сотрудников;
- выявляются сильные и слабые стороны каждого сотрудника;
- определяются области для развития и повышения эффективности работы.

Обратная связь и развитие сотрудников:

- результаты оценки доводятся до сведения каждого сотрудника
- обсуждаются сильные стороны и области для улучшения
- разрабатываются индивидуальные планы развития сотрудников

Сам алгоритм оценки сотрудников представлен на рисунке 2.

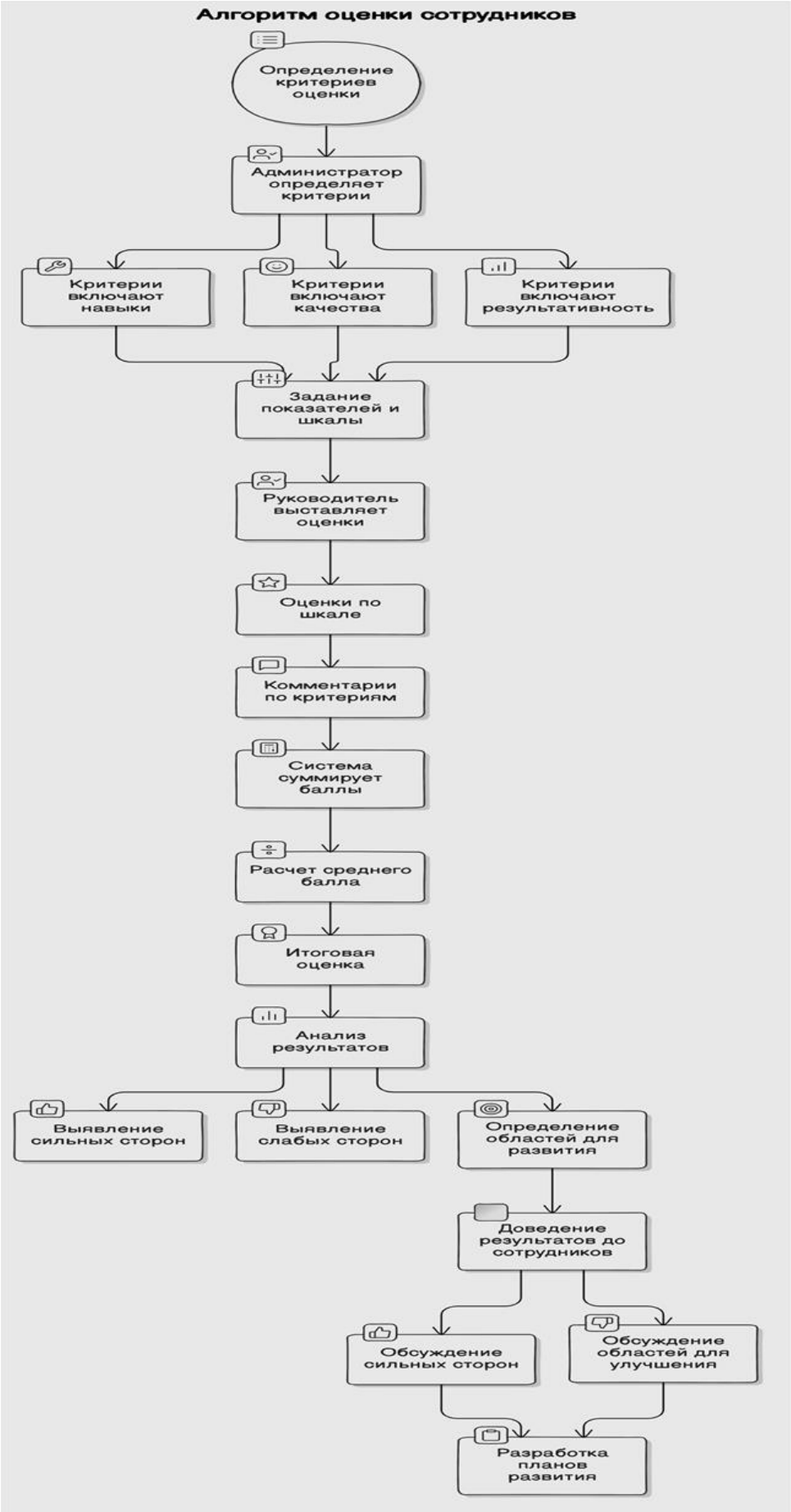

Рисунок 2 – Алгоритм оценки сотрудников

Рассмотрим основные задачи, которые могут решатся с использованием методики «360 градусов».

1. Например, если нужно найти тех, кто готовы к повышению, особенно в таких кампаниях, которые огромные, и где много различных филиалов, где у HR-отдела нет времени и возможности оценивать каждого сотрудника отдельно. Анкетирование по методу «360 градусов» может поспособствовать в кратчайшие сроки найти тех, кто заслуживает повышения.

2. Или же если нужно спланировать обучение, когда есть понимание, что сотрудникам не хватает знаний, но неизвестно кому, и каких конкретно — оценка «360 градусов» сможет помочь в этом.

3. Если задача состоит в оценке управленческих навыков руководителей с участием подчинённых, обеспечение анонимности отзывов становится ключевым аспектом для получения искренних мнений о начальствах. Это позволяет создать объективную картину восприятия руководства среди сотрудников.

4. Если нужно сформировать обратную связь для сотрудника. Сама процедура может стать основой для формирования полезной обратной связи. Использование мнений коллег и подчинённых, при этом не раскрывая их личности, дает руководителю возможность узнать о разных взглядах на свою работу и корректировать её с учётом этой информации.

5. Когда речь идет об оценке таких качеств, как умение общаться и слушать, инициативность и другие мягкие навыки, важно установить чёткие цели оценки, отражающие корпоративные требования, например, необходимость повышения качества обслуживания клиентов или наличия в штате высококвалифицированных специалистов. Понимание и принятие целей оценки сотрудниками как инструмента для их развития является важным фактором успеха.

«Очень важно обеспечить открытость информации о предстоящей оценке, начиная информационную кампанию за три месяца до её начала» [3]. Это помогает разрешить возможные опасения и вопросы, а также настроить коллектив на конструктивное участие.

Данная цель объясняется сотрудникам, при этом главное, чтобы все

поняли: мероприятие направлено на их развитие, и по итогам оценки никто не будет уволен. Кроме этого, работники должны знать, что получаемые в ходе исследования данные носят конфиденциальный характер, все сведения предоставляются только объекту оценки и его непосредственному руководителю, а фамилии участников опроса не разглашаются. Эти правила следует довести до сведения персонала через руководителей, а также посредством корпоративного сайта, доски объявлений, с помощью информационных рассылок по электронной почте и других материалов (листовок, информационных писем, статей в корпоративной газете и т.д.).

После этого можно разворачивать PR-кампанию предстоящего мероприятия. «Начинать же ее необходимо не позднее, чем за три месяца до начала оценки [4]». Информационная открытость облегчает работу HR-службы, дает возможность уже на стадии подготовки понять отношение сотрудников к событию, выяснить их опасения, проверить ожидания. Важно сообщить обо всех предварительных этапах мероприятия (описание компетенций, составление списков участников). Постепенное вовлечение руководителей и подчиненных в процедуру, понимание ее структуры и последовательности стадий, причастность к подготовке – все это снижает риск сопротивления нововведению [5].

Документы, сопровождающие PR-кампанию: положение об оценке, описание критериев оценки (компетенций), правила расчета итоговых баллов, план мероприятия.

Рассмотрим основные правила предоставления обратной связи после проведения оценки методом «360 градусов»:

- обратная связь должна осуществляться не позднее двух недель после оценки. Только в этот период данные остаются актуальными;
- при подготовке отчета о результатах важно соблюдать принципы прозрачности и конструктивности обратной связи. Рекомендуется начинать с позитивных моментов, а затем переходить к аспектам, требующим улучшения. Дополнительно, отчет должен включать визуальные элементы, такие как диаграммы и таблицы, для лучшего восприятия информации. Особое внимание следует уделить предложениям по развитию компетенций, что может включать

самоанализ и план действий на будущее. Такой подход позволяет не только оценить текущее состояние управленческих качеств, но и предложить пути их развития, что способствует повышению общей эффективности и удовлетворенности работы в компании;

- содержание передаваемой информации должно быть понятно сотруднику.

Поговорим подробнее о том, как оформить письменный отчет.

Для составления письменного отчета обратной связи можно использовать следующий план:

- обращение к участнику оценки,
- сроки оценки,
- цели мероприятия,
- информирование об анонимности оценки,
- критерии оценки. Краткое описание компетенций для конкретной должности;
- описание шкал, по которым производилась оценка. В случае использования числовых значений приводится их качественное описание;
- результаты оценки и их интерпретация. Этот пункт представляет собой основную содержательную часть отчета. Все данные должны быть достаточно подробными, но приводиться в легкой для восприятия форме. В завершение этой части отчета оцениваемому предлагаются 2–3 вопроса, направленных на развитие его компетенций. Например: «Как вы думаете, что послужило причиной того, что по компетенции "своевременное информирование, обратная связь с членами команды» руководитель оценил вас высоко, а коллеги поставили низкий балл?». Подобное анкетирование поможет сотруднику сосредоточить внимание на полученных данных и связать их со своим поведением;
- общие выводы, описание примерных результатов, которых ожидают от работника при проведении следующей оценки.

Для написания отчета можно заранее заготовить шаблон и использовать его для предоставления обратной связи.

При составлении содержательной части отчета важно помнить, что лучше всего воспринимается информация, когда она представлена как в числовом, графическом (таблицы, схемы), так и словесном (текст) виде.

К примеру, диаграммы очень наглядны и легко воспринимаются. Так, на рисунке 3 диаграмма отражает средние оценки. Эти данные позволят сотруднику понять картину в целом: по каким компетенциям он получил высокие баллы, а по каким – низкие.

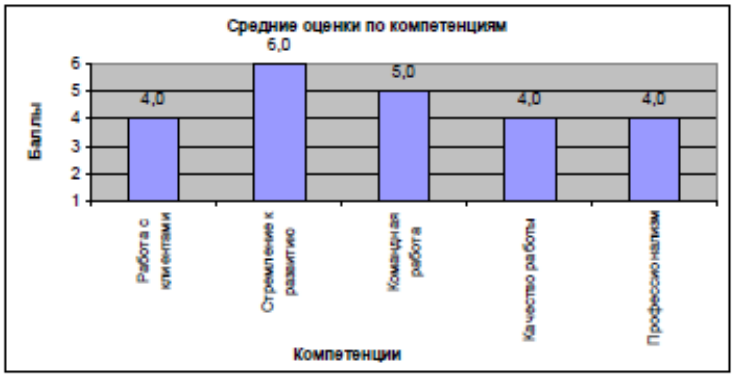

Рисунок 3 – Средние оценки по компетенциям

Чтобы оцениваемый мог разобраться, почему по той или иной компетенции он получил низкий балл, и мог скорректировать свою деятельность, следует описать индикаторы поведения, составляющие данную компетенцию. Например, «качество работы» может включать следующие индикаторы:

- выполняет работу в срок;
- выполняет работу качественно, без ошибок;
- при выполнении работы соблюдает стандарты компании.

Эту информацию можно представить и в виде диаграммы оценок конкретных индикаторов, составляющих компетенцию, которая отображена на рисунке 4.

Рисунок 4 - Пример средних оценок по индикаторам

Благодаря такому представлению данных сотруднику легче разобраться, по каким конкретным поведенческим проявлениям он получил низкую оценку, и скорректировать свою деятельность. В частности, из диаграммы на Рисунке 2 видно, что он соблюдает стандарты компании и без ошибок выполняет свою работу, однако не всегда соблюдает сроки.

При работе с графическими и цифровыми данными при составлении содержания отчета необходимо помнить, что, несмотря на преимущества данной формы изложения материала, не стоит перегружать его таблицами и схемами и придерживаться следующего правила:

- используйте графические элементы только для объяснения компетенций, по которым сотрудник получил низкий балл;
- добавьте текст (желательно), который составляет около 2/3 каждой таблицы или диаграммы.

Стоит обратить внимание на высокие и низкие баллы и прокомментировать, что может вызвать такие оценки, или задать сотруднику вопрос об этом.

Методика «360 градусов» — это комплексный подход, позволяющий получить исчерпывающую информацию о профессиональных качествах и способностях каждого сотрудника. В отличие от традиционных методов, где оценки часто опираются исключительно на мнение непосредственного руководителя, учитывает мнение коллег, подчиненных, руководителей и самого

сотрудника. «Это дает более объективное и всестороннее представление о сотруднике, тем самым способствуя его профессиональному развитию и повышению производительности» [6].

Во-первых, руководство компании определяет цели оценки и основные компетенции, подлежащие оценке. Эти критерии могут включать профессиональные навыки, личные качества, умение работать в команде, лидерство и другие важные аспекты. После этого для каждого сотрудника формируется оценочная группа, в которую входят коллеги, подчиненные, руководители и сам сотрудник. Это даст вам общее представление о его профессиональных качествах.

Основным инструментом сбора данных была анкета, заполненная всеми участниками в процессе оценки. Анкеты могут быть стандартизированы или адаптированы к конкретным потребностям компании. Сотрудники также заполняют анкеты для самооценки, чтобы оценить свои навыки и способности. Это важный элемент, позволяющий сравнить свою самооценку с оценками других участников.

Все анкеты были собраны и систематизированы в мобильной системе. Система автоматически обрабатывает данные, сокращая время анализа и сводя к минимуму вероятность ошибок. Особое внимание уделено анализу различий самооценок сотрудников и других участников. Это помогает определить области развития и потенциальные проблемы.

Каждый сотрудник получает индивидуальный отчет, содержащий результаты оценки по различным компетенциям. Отчеты должны быть четкими и содержать конкретные рекомендации по развитию. Менеджеры встречаются с сотрудниками, чтобы обсудить результаты оценки и определить сильные стороны и области для улучшения.

На основании полученных результатов разрабатывается персональный план развития для каждого сотрудника. Программа может включать обучение, коучинг, участие в проектах и другие мероприятия. Процесс оценки не останавливается на этапе обратной связи, реализацию плана развития также необходимо регулярно контролировать и при необходимости вносить коррективы.

«Информационные системы играют ключевую роль на всех этапах процесса оценки» [7]. Он автоматизирует сбор и обработку данных, значительно упрощая оценки и повышая их точность. Система также предоставляет удобные инструменты анализа данных и отчетности, позволяющие менеджерам быстро получать информацию, необходимую для принятия решений.

Таким образом, в данном подразделе были описаны особенности реализации методики «360 градусов»

1.3 Требования к мобильному приложению по управлению процессом оценки сотрудников

Прежде чем разрабатывать мобильную систему, важно определить цели и задачи, которые она должна решать.

«Большинство компаний сталкиваются с необходимостью оценки сотрудников» [8]. Вы можете установить определенные KPI и контролировать их выполнение. Таким образом, задача сводится к простому сбору и анализу цифр. Что делать, если работа эксперта не позволяет ему получить четкие числовые показатели? Или – и это более распространено – эксперт официально выполняет все требования, но продолжает обедать один, а все остальные сотрудники его избегают? Здесь важны уже не показатели, а коммуникативные навыки. Их можно оценить с помощью специальной системы оценки персонала.

Основной целью при разработке мобильного приложения для оценки сотрудников было внедрение простого и удобного инструмента оценки сотрудников для предприятия «Квартплата 24» методом оценки «360 градусов».

Плюсы метода оценки «360 градусов»:

- позволяет оценить эффективность каждого сотрудника и то, насколько он соответствует своей должности;
- даёт возможность выявить «точки роста» сотрудников и команды;
- можно провести силами HR-департамента. В отличие от ассессмент-центра, который предполагает работу команды внешних экспертов;
- позволяет охватить большое количество сотрудников, если анкетирование проводится онлайн;

– позволяет оценить soft skills — навыки взаимодействия с окружающими.

Минусы метода:

– может привести к неприятным последствиям, если не будет соблюдена анонимность. В частности, к ухудшению отношений между сотрудниками и внутренним конфликтам;
– возможны серьёзные затраты времени на информирование участников.

А после проведения оценки — на обработку данных и анализ результатов. Именно эту проблему решает разрабатываемая мобильная система;

«Метод оценки «360 градусов» позволяет выявить соответствие сотрудника занимаемой должности путём сбора отзывов от его непосредственных руководителей и коллег» [9]. Расширение оценки за счёт включения в процесс клиентов и подрядчиков превращает её в оценку «540 градусов». Такой подход дает возможность создать всестороннее и объективное представление о профессиональных качествах сотрудника и его соответствии определённым компетенциям.

Использование этого метода оправдано, поскольку он хорошо зарекомендовал себя для оценки сотрудников по определенным критериям и позволяет оценивать и самооценивать сотрудников по различным критериям.

Объектом исследования является процесс оценки сотрудников предприятия и соответствующие методы.

Проектируется удобное мобильное приложение, которое разрабатывается для упрощения и улучшения процесса оценки компетенций сотрудников на предприятии.

Задачи, необходимые для достижения цели:

– дать общую характеристику процесса оценивания сотрудников и выбрать подходящую для реализации методику;
– выполнить постановку задачи на разработку системы, провести идентификацию требований пользователей и определение основных функциональных и нефункциональных требований;

- провести проектирование архитектуры мобильного приложения, определив основные модули и взаимосвязи между ними, а также взаимодействие с базой данных через соответствующий API;
- провести разработку схемы данных приложения;
- реализовать пользовательский интерфейс и основные модули мобильного приложения;
- провести тестирование разработанного мобильного приложения для оценивания сотрудников.

В конечном итоге система должна способствовать развитию сотрудников, улучшению их мотивации и вовлеченности, а также поддержке принятия обоснованных управленческих решений.

Требования определяют функции, возможности или задачи мобильного приложения, которые поясняют, как приложение должно вести себя в различных ситуациях. Требования подразделяются на функциональные и нефункциональные.

Анализ требований — часть процесса разработки программного обеспечения, включающая в себя сбор требований к программному обеспечению (ПО), их систематизацию, выявление взаимосвязей, а также документирование. «Является частью общеинженерной дисциплины «инженерия требований» (англ. Requirements Engineering)» [10].

Функциональные требования — это возможности или функции продукта, которые разработчики должны реализовать, чтобы пользователи могли выполнять свои задачи. Поэтому важно сделать их понятными как для команды разработчиков, так и для заинтересованных сторон. Как правило, функциональные требования описывают поведение системы в определенных условиях.

Функциональные требования различаются по функциям, которые они описывают. В соответствии с таким классификационным подходом можно выделить следующие виды функциональных требований.

Аутентификация представляет собой комплекс мероприятий, нацеленных на подтверждение личности пользователя перед предоставлением доступа к

системе. «Это может включать использование имен и паролей пользователей, биометрические методы проверки и многофакторную аутентификацию» [11].

Уровни доступа в системе определяются через механизмы авторизации, которые устанавливают, какой уровень доступа имеет каждый пользователь. Например, администраторы могут обладать неограниченным доступом, в то время как доступ обычных пользователей может быть ограничен только некоторыми функциями.

Обработка данных охватывает всё от ввода и проверки до хранения и извлечения информации в системе.

Разработка пользовательского интерфейса и опыта (UI/UX) касается создания и оптимизации взаимодействия пользователей с системой, чтобы она была интуитивно понятна и удовлетворяла их потребности.

Требования к отчетности задают правила составления отчетов, включая определение источников данных и форматов представления информации.

Интеграция системы с другими приложениями и внешними сервисами описывает, как должно происходить взаимодействие и совместная работа с другими системами.

Требования к обработке транзакций критически важны для систем, занимающихся финансовыми операциями или учетом транзакций.

Обработка ошибок и ведение журналов определяет методы диагностики и регистрации проблем в системе, а также действия по их устранению.

Механизмы резервного копирования и восстановления данных гарантируют сохранность информации и возможность восстановления системы после сбоев или других проблем.

Нефункциональные требования не связаны с функциональностью системы, а скорее определяют, как система должна работать. Они имеют решающее значение для обеспечения удобства использования, надежности и эффективности системы, часто влияя на общий пользовательский опыт. Далее мы подробно опишем основные категории нефункциональных требований.

Процесс анализа требований охватывает несколько ключевых этапов:

– сбор требований — «этот этап включает взаимодействие с клиентами и конечными пользователями для определения их потребностей и

ожиданий» [20]. Также проводится изучение предметной области для глубокого понимания контекста;

- анализ требований — на этом этапе происходит оценка собранных требований на предмет их ясности, полноты, однозначности и взаимосвязей. Цель — выявить и устранить любые несоответствия или противоречия между требованиями;
- документирование требований — требования фиксируются в документальной форме, которая может варьироваться от текстовых описаний до более структурированных форм, таких как сценарии использования, пользовательские истории или спецификации процессов. Это обеспечивает чёткость и доступность информации для всех заинтересованных сторон.

Требования имеют как свои преимущества, в виде того, что помогают разработчикам убедиться, что система отвечает всем потребностям клиента, но также и недостатки, в виде того, что при неправильной постановке их бывает трудно понять и реализовать.

Основываясь на предыдущем описании характеристики процесса оценки сотрудников и определенным целям, и задачам разработки, разделим классификацию требования FRUPS на функциональные и нефункциональные.

Функциональные требования к мобильному приложению, разделив их на требования для каждой роли.

Функциональные требования:

- сотрудник должен иметь возможность регистрации;
- сотрудник должен иметь возможность авторизации;
- сотрудник должен иметь возможность прохождения опросов;
- руководитель должен иметь возможность добавлять новые опросы и критерии к ним;
- руководитель должен иметь возможность посматривать результаты опросов в мобильном приложении или с использованием веб-интерфейса;
- система должна рассчитывать средний балл для каждого сотрудника.

Нефункциональные требования:

Удобство использования:

- приложение должно иметь интуитивно понятный интерфейс для ввода данных;
- надежность;
- приложение должно быть доступно 24/7;
- должна сохраняться точность в расчетах согласно установленным правилам прохождения опросов.

Производительность:

- сервер должен иметь возможность одновременной работы до 25 пользователей одновременно;
- сервер должен иметь возможность обрабатывать большие объемы данных;
- серверная часть должна обрабатывать запросы от мобильного приложения;
- должна быть реализована возможность обработки ошибок.

Поддерживаемость:

- приложение должно поддерживаться на любых мобильных устройствах с версией операционной системы не ниже Android 5.0;
- приложение должно минимально использовать память и процессор устройства;
- приложение должно быть не требовательным к программным и аппаратным характеристикам используемого устройства.

Диаграмма вариантов использования представляет собой графическое изображение возможных взаимодействий пользователя с системой. Диаграмма вариантов использования показывает различные варианты использования и различные типы пользователей, которые есть в системе, и часто сопровождается диаграммами других типов. Варианты использования представлены кружками или эллипсами. Актеров часто изображают в виде фигурок.

Таким образом, на основе разрабатываемых требований можно построить Use Case диаграмму для разрабатываемой системы. «Диаграмма вариантов

использования в UML — диаграмма, отражающая отношения между акторами (пользователями) и прецедентами, являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне» [21]. Эта диаграмма представлена на рисунке.

На диаграмме, представленной на рисунке 5, описаны два актера и их действия в приложении.

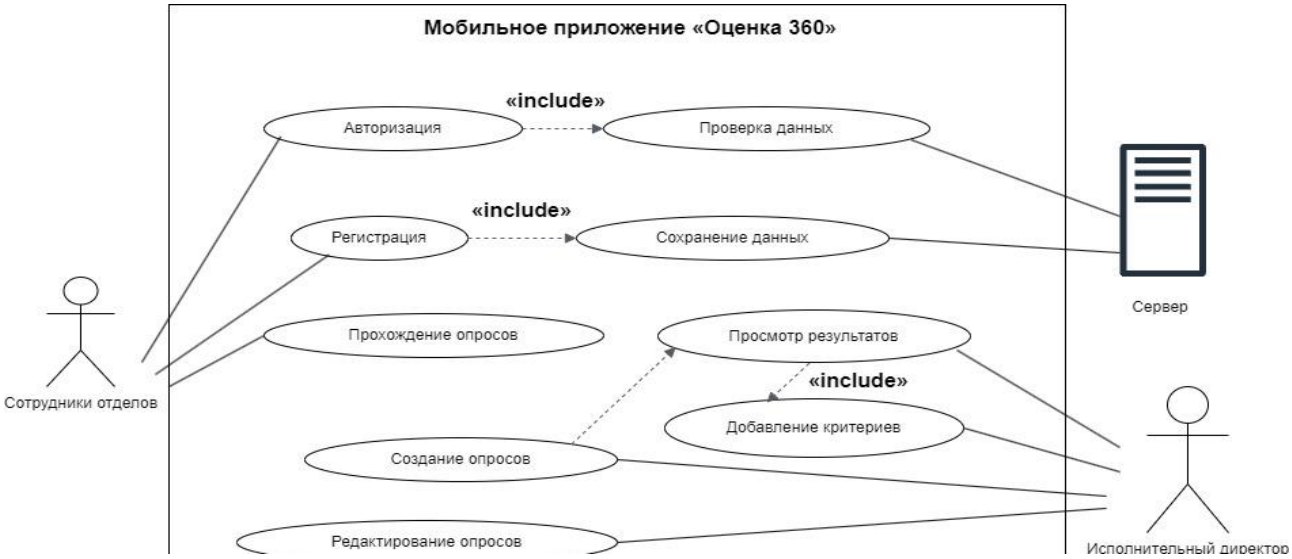

Рисунок 5 – Диаграмма прецендентов использования сотрудниками организации мобильного приложения

Сотрудники могут входить в систему, регистрироваться, участвовать и проходить опросы, при этом данные проверяются и сохраняются на сервере. Руководитель может создавать опросы, просматривать результаты, добавлять критерии и редактировать опросы. Эти данные также хранятся на сервере через API.

В первой главе был проведен детальный анализ текущей системы оценки сотрудников на предприятии. Методика «360 градусов» была выбрана за ее способность предоставлять всестороннюю оценку от различных участников, включая коллег, подчиненных и руководителей. Были определены ключевые особенности и проблемы текущей системы, а также выявлены потребности пользователей. В результате анализа сформулированы как функциональные, так и нефункциональные требования к новому приложению, включая цели повышения объективности, прозрачности и автоматизации процесса оценки. Особое внимание было уделено улучшению управления данными для HR-специалистов и руководителей, а также обеспечению своевременной обратной связи для сотрудников. Проведенный анализ позволил заложить прочную основу для разработки эффективного инструмента оценки, соответствующего современным требованиям. Это обеспечит не только повышение качества оценок, но и поддержку профессионального развития сотрудников.

Глава 2 Проектирование мобильного приложения

2.1. Проектирование архитектуры приложения

Приложение было реализовано с использованием REST.

«REST — (сокращенно от англ. Representation State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределенного приложения в сети по модели клиент-сервер» [12]. Он был разработан в диссертации Роя Филдинга в 2000 году как альтернатива SOAP, когда запрос клиента несет исчерпывающую информацию о любом ответе сервера, и серверу не требуется поддерживать сеанс взаимодействия с клиентом.

Основные особенности REST:

- стандартизированные методы: GET, POST, PUT, DELETE и другие методы HTTP используются для взаимодействия с ресурсами;
- унифицированный интерфейс: Универсальный способ взаимодействия с ресурсами, позволяющий использовать стандартные методы для выполнения операций;
- отсутствие состояния: Каждый запрос от клиента к серверу должен содержать всю необходимую информацию для обработки запроса. Это упрощает масштабирование сервера и управление сессиями;
- кэшируемость: Ответы от сервера могут быть кэшируемыми, что позволяет сократить нагрузку на сервер и улучшить производительность приложения.

На рисунке 6 представлена архитектура программной системы, использующей REST API для взаимодействия между клиентом и сервером. Формат обмена данными между клиентом и сервером был выбран JSON (JavaScript Object Notation) из-за его легкости и простоты интеграции с различными языками программирования.

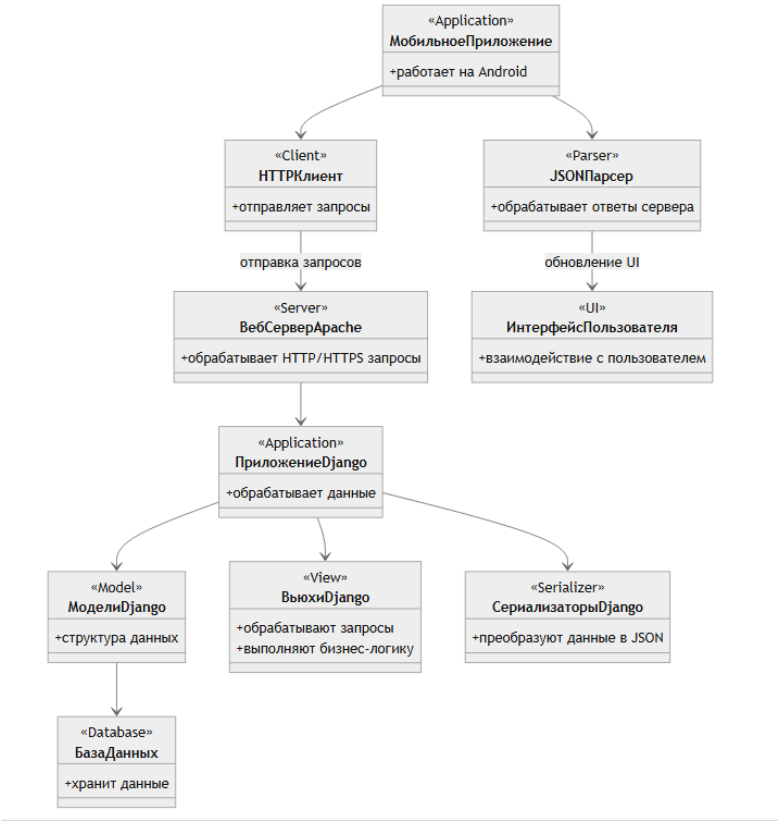

Рисунок 6 – Архитектура программной системы

REST API — это набор удаленных вызовов стандартных методов, которые возвращают данные в определенном формате. В нашем случае использовался формат JSON.

2.2 Определение основных компонентов и модулей приложения

Приложение включает в себя несколько ключевых компонентов и модулей, которые взаимодействуют между собой для обеспечения полной функциональности системы.

Основные компоненты:

– клиентская часть: Это интерфейс, с которым взаимодействует пользователь. Клиентская часть была разработана с использованием Android Studio, что позволяет создавать удобные и интерактивные пользовательские интерфейсы для устройств на базе Android;

- серверная часть: Серверная часть отвечает за обработку запросов от клиента и выполнение бизнес-логики приложения. Для реализации серверной части использовался фреймворк Django, который предоставляет мощный и гибкий набор инструментов для создания веб-приложений;
- база данных: Для хранения данных приложения была выбрана СУБД SQLite. Эта база данных легка в использовании и идеально подходит для мобильных приложений, так как обеспечивает быстрый доступ к данным при низком уровне потребления ресурсов.

Основные модули:

- модуль авторизации и регистрации: отвечает за создание новых пользователей и аутентификацию существующих;
- модуль опросов: управляет созданием, редактированием и проведением опросов среди пользователей;
- модуль результатов: сохраняет и отображает результаты опросов, предоставляя аналитику и отчеты;
- модуль управления ролями: позволяет назначать и управлять ролями пользователей в системе, обеспечивая гибкую настройку уровней доступа.

Схема размещения компонентов системы представлена на рисунке 7, где наглядно отображены различные компоненты и их взаимодействие.

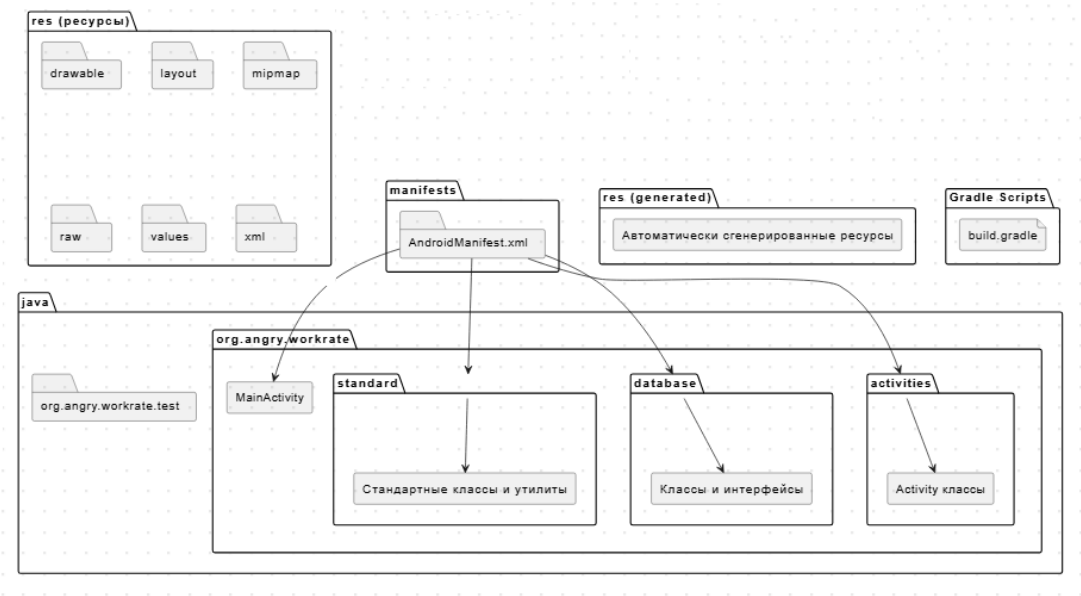

Рисунок 7 – Схема размещения компонентов системы

Смотря на этот рисунок, можно понять как размещены разные компоненты системы в данном приложении.

Определение основных компонентов и модулей приложения является важным этапом разработки. Каждый компонент выполняет свою специфическую функцию, обеспечивая стабильную работу системы.

Основные компоненты включают:

– клиентская часть, которая включает пользовательский интерфейс, разработанный на платформе Android Studio;
– серверная часть, в которую входит обработка запросов и бизнес-логика на базе Django;
– база данных, для хранение данных с использованием SQLite.

Основные модули включают:

– авторизация и регистрация, например управление пользователями;
– опросы, это создание и проведение опросов;
– результаты, сохранение и отображение результатов;
– управление ролями, настройка уровней доступа.

Алгоритм проведения оценки сотрудников в мобильном приложении представлен на рисунке 8.

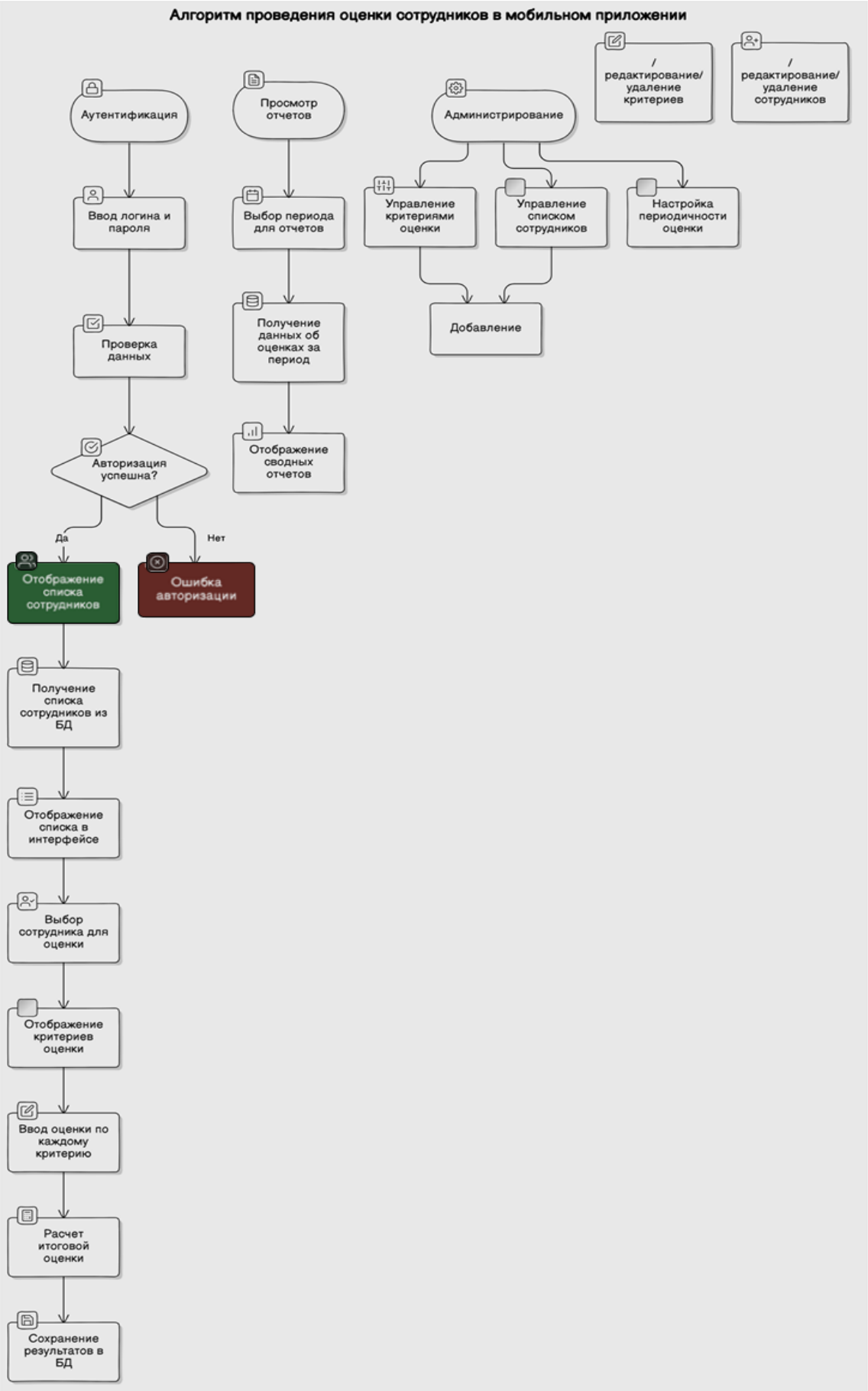

Рисунок 8 – Алгоритм проведения оценки сотрудников мобильном приложении

Аутентификация пользователя:

– пользователь вводит логин и пароль;
– приложение проверяет введенные данные и авторизует пользователя.

Отображение списка сотрудников:

– приложение получает список всех сотрудников из базы данных;
– список сотрудников отображается в интерфейсе приложения.

Проведение оценки сотрудника:

– пользователь выбирает сотрудника для оценки;
– отображаются критерии оценки;
– пользователь вводит оценку по каждому критерию;
– приложение рассчитывает итоговую оценку сотрудника;
– результаты оценки сохраняются в базе данных.

Просмотр отчетов по оценке:

– пользователь может выбрать период для просмотра отчетов;
– приложение получает данные об оценках сотрудников за выбранный период;
– отображаются сводные отчеты по результатам оценки.

Администрирование приложения:

– пользователи с соответствующими правами могут добавлять/редактировать/удалять критерии оценки;
– могут управлять списком сотрудников (добавлять, редактировать, удалять);
– настраивать периодичность проведения оценки.

2.3. Разработка схемы базы данных и выбор соответствующих технологий

В этом подразделе рассмотрим средства разработки мобильного приложения и обоснуем выбор конкретной из них.

Программа будет разрабатываться в среде разработки Android Studio. Эта среда от Google создана специально для разработки приложений для

операционной системы Android. «Он содержит макеты для создания пользовательского интерфейса, с которого обычно начинается работа над приложением. Android Studio содержит инструменты для разработки решений для смартфонов и планшетов, а также Android TV, Android Wear, Android Auto, Glass и других контекстов. Новые технологические решения. для модулей» [13].

Android Studio имеет множество преимуществ перед другими средами разработки.

Visual Studio Code — редактор кода Microsoft. Хоть это и обычный редактор, у него есть отличные возможности в виде плагинов и расширений, которые можно использовать для разработки мобильных приложений. «Теоретически разработчики практически на любом современном языке могут использовать VS Code. Но на практике часто он используется там, где полноценные возможности IDE не нужны». [15]. Интерфейс данной программы отображен на рисунке 9.

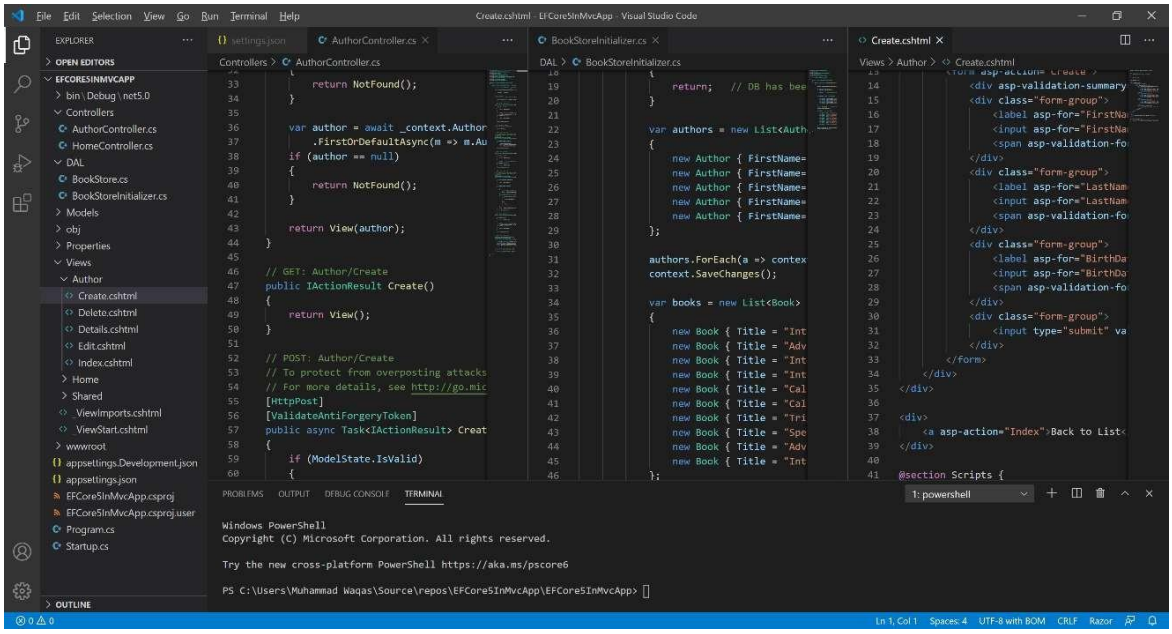

Рисунок 9 - Интерфейс Visual Studio Code

Разработка базы данных включает в себя следующие этапы.

Процесс разработки базы данных для анкетирования пользователей начинается с определения основных сущностей и связей между ними.

Пользователи системы проходят опросы в системе. Каждый опрос включает критерии тестирования, и содержит список возможных вариантов ответа.

Результаты тестирования сохраняются в таблицу результатов.

Пользователи относятся к определенным ролям.

Таким образом, сущности, которые должны быть представлены в базе данных – следующие:

- пользователи;
- опросы;
- критерии;
- шкала оценивания;
- результаты оценивания.

Концептуальная модель разрабатываемой базы данных представлена на рисунке 10.

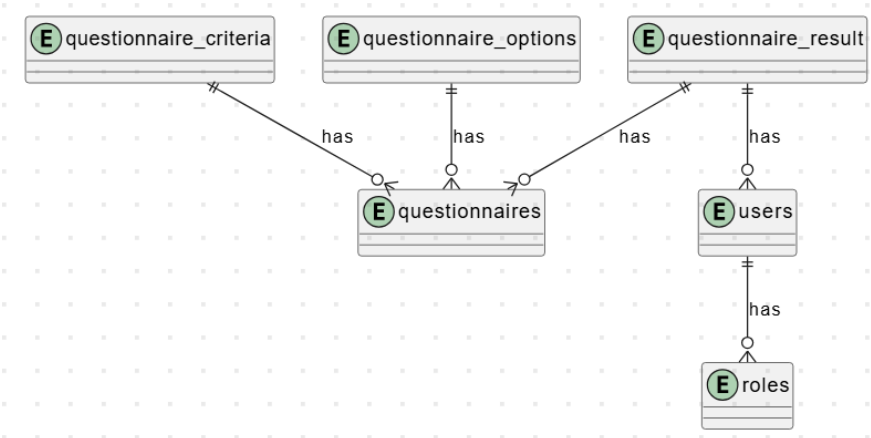

Рисунок 10 – Диаграмма классов БД

Далее была получена логическая модель данных, представленная на рисунке ниже.

На основе данного описания была получена диаграмма классов, которую необходимо реализовать в приложении.

На рисунке 11 отображена физическая модель данных.

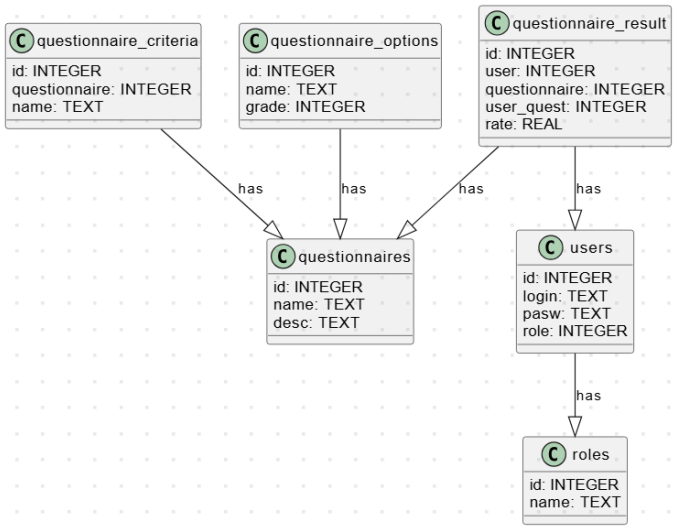

Рисунок 11 – Физическая модель данных

«Наиболее приемлемым выбором для реализации проекта является связка Python + Django + СУБД SQLite, поскольку SQLite является по умолчанию используемой с Python СУБД» [22].

Django — это фреймворк для создания веб-приложений на Python, который следует принципу DRY (Don't Repeat Yourself), предполагающему минимизацию повторения кода. Веб-проекты на Django состоят из одного или нескольких модульных приложений, которые рекомендуется разрабатывать так, чтобы их можно было легко переиспользовать или интегрировать в другие проекты. Это отличает архитектуру Django от некоторых других фреймворков, например, Ruby on Rails. В отличие от многих фреймворков, в Django маршрутизация URL задается явно с использованием регулярных выражений, а не генерируется автоматически на основе контроллеров.

Изначально Django разрабатывался для использования с сервером Apache с модулем mod_python и базой данных PostgreSQL. Сегодня Django поддерживает работу с различными системами управления базами данных, включая MySQL (MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle. Django использует собственную ORM систему, в которой структура данных определяется через классы Python, на основе которых затем формируется схема базы данных.

«Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (Templates). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV)».[14]

Проанализируем возможные средства разработки мобильного приложения и сведём всё к таблице 1.

Таблица 1 – Анализ средств разработки

Критерий	Android Studio	IntelliJ IDEA	Visual studio
Удобство использования	Интуитивный интерфейс, обилие шаблонов и мастеров, отличная документация	Удобный и настраиваемый интерфейс, мощные рефакторинговые возможности	Сложный интерфейс, высокая настраиваемость, интеграция с Windows
Надежность и безопасность	Высокая стабильность, регулярные обновления и исправления, поддержка Google	Высокая надежность, регулярные обновления от JetBrains	Высокая стабильность, надежная поддержка от Microsoft
Производительность	Высокая производительность, особенно на мощных машинах, оптимизированная сборка и отладка	Высокая производительность, адаптивность к различным проектам	Высокая производительность, хорошая поддержка многозадачности
Поддерживаемость	Отличная документация, большое сообщество, интеграция с Google Services, поддержка через форумы и ресурсы Google	Широкая документация, активное сообщество, поддержка JetBrains	Обширная документация, поддержка от Microsoft, интеграция с Azure
Функциональность	Полная поддержка Android, встроенный эмулятор, инструменты для тестирования	Поддержка множества языков, расширяемость через плагины, поддержка Android	Поддержка множества языков, интеграция с Azure

Анализ показывает, что Android Studio является лучшим выбором, так как имеет следующие возможности:

– понятный интерфейс;
– мощный редактор кода, который обладает всеми нужными;
– функциями для повышения производительности при разработке;
– (автозаполнение, подсветка синтаксиса, быстрая и понятная навигация);
– поддержка языка программирования Java;
– богатые возможности отладки;
– интеграция с другими инструментами Google;
– инструменты для тестирования;
– обширная документация.

Разработка базы данных включает в себя следующие этапы.

– определение требований: на данном этапе формулируются цели создания базы данных, определяется перечень информации, которая будет храниться в базе данных, и способы ее использования.;
– проектирование концептуальной модели: на этом этапе часто используют ER-диаграммы (Entity-Relationship Diagram);
– проектирование логической модели: концептуальная модель преобразуется в логическую, которая уже учитывает особенности конкретной СУБД. Определяются типы данных для каждого атрибута, первичные и внешние ключи, индексы и другие ограничения целостности данных;
– физическое проектирование: на этом этапе определяется способ хранения данных на физическом носителе (жесткий диск, SSD). Выбираются типы файлов для хранения таблиц, методы индексирования, параметры разбиения таблиц на разделы и т.д.
– создание базы данных: используя язык SQL, создаются таблицы, индексы, ограничения целостности и другие объекты базы данных;

– заполнение данными и тестирование: база данных наполняется тестовыми данными, проверяется корректность работы запросов, производительность системы и другие аспекты.

Каждый из перечисленных этапов является важным шагом в обеспечении успешной разработки базы данных. Тщательно выполненное определение требований и проектирование концептуальной модели позволяют избежать ошибок на ранних стадиях. Логическое и физическое проектирование обеспечивают соответствие базы данных техническим требованиям и характеристикам оборудования. Наконец, создание базы данных, её наполнение и тестирование гарантируют, что система будет работать корректно и эффективно.

Таким образом, комплексный подход к разработке базы данных, включающий все перечисленные этапы, является ключом к созданию надежной и производительной мобильной системы, способной удовлетворить потребности бизнеса и пользователей.

Во второй главе был описан процесс проектирования архитектуры мобильного приложения. Рассматриваются основные компоненты и модули системы, включая интерфейс пользователя, базу данных и взаимодействие с сервером через REST API. Были выбраны технологии для реализации проекта: Android Studio для разработки клиентской части и SQLite для базы данных. В главе подробно описаны концептуальная и логическая модели данных, которые легли в основу физической модели. Особое внимание уделено нефункциональным требованиям, таким как удобство использования, надежность, производительность и поддерживаемость приложения.

Глава 3 Разработка мобильного приложения

3.1 Разработка функциональности приложения

Для реализации API приложения был разработан интерфейс на языке Python с использованием Фреймворка Django.

API реализован с использованием Django-View (функций представления), отвечающих за обработку HTTP-запросов для получения информации о пользователе по его идентификатору.

Код функции авторизации отображен на рисунке 12.

Вначале определяются декораторы

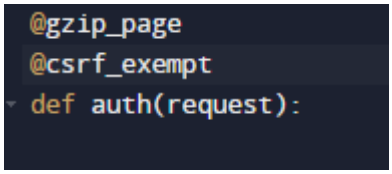

Рисунок 12 – Код авторизации

@gzip_page: «Данный декаратор с использованием алгоритма gzip, что уменьшает размер данных, передаваемых клиенту, и ускоряет загрузку страницы» [24].

@csrf_exempt: Этот декоратор отключает проверку CSRF (Cross-Site Request Forgery) для данного представления.

Функция users_info представлена на рисунке 13.

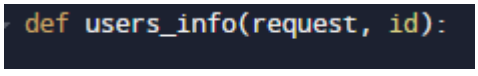

Рисунок 13 – Информация о пользователе

Функция принимает на входе два параметра: request: Объект запроса, который содержит основную информацию из запроса, id - идентификатор пользователя, информацию о котором нужно получить.

Из базы данных извлекается объект пользователя по идентификатору. Если такой пользователь существует – выводится исключения, как отображено на рисунке 14.

```
r = QuestionnaireResult.objects.filter(user_quest=u)
```

Рисунок 14 – Извлечение результатов

Данная строка извлекает все результаты анкетирования, связанные с данным пользователем.

После этого формируется ответ сервера, представленный на рисунке 15.

```
obj = {
'role': u.role.name,
'login': u.login,
'rate': 0 if len(r) < 1 else r.aggregate(Avg('rate'))['rate__avg'],
}
```

Рисунок 15 – Ответ сервера

Здесь создается словарь obj, содержащий имя роли пользователя, его логин, и среднюю оценку, которая высчитывается на основе оценок остальных сотрудников.

Если у пользователя нет результатов анкетирования, рейтинг устанавливается в 0. «В противном случае используется агрегатная функция Avg для вычисления среднего значения рейтинга» [23].

Далее сервер возвращает ответ с использованием функции, со статусом 200 (успешно), как отображено на рисунке 16.

```
def users_info(request, id):
```

Рисунок 16 – Получение объекта пользователя

Функция users_info получает идентификатор запрос, и идентификатор пользователя, после чего получает из базы данных объект данного пользователя.

На рисунке 17 отображено, как данные переменные передаются в словарь, после чего функция возвращает данный объект.

```
def users_info(request, id):
u = Users.objects.get(id=id)
r = QuestionnaireResult.objects.filter(user_quest=u)
obj = {
'role': u.role.name,
'login': u.login,
'rate': 0 if len(r) < 1 else r.aggregate(Avg('rate'))['rate__avg'],
}
return HttpResponse(json.dumps(obj), 200)
```

Рисунок 17 – Возвращение объекта

Функция questionnaries считывает все данные из таблицы опросов, после чего передает их в список и возвращает данный объект через json, как представлено на рисунке 18.

```
def questionnaires(request):
list = []
for i in Questionnaires.objects.all():
list.append({
'id': i.id,
'name': i.name,
'desc': i.desc,
})
return HttpResponse(json.dumps(list), 200)
```

Рисунок 18 – Возврат списка анкет

Функция questionnaries_info получает объект опроса из БД по его id в соответствии с запросом, после чего возвращает данный объект с использованием json.

Этот код представляет собой функцию представления, отвечающую за обработку HTTP запросов клиента, для получения информации из БД об анкете по ее идентификатору.

Сначала извлекается объект анкеты из БД, с использованием передаваемого в функцию идентификатора, как отображено на рисунке 19, которая отвечает за обработку HTTP-запросов для получения информации об анкете по её идентификатору, после чего формируется словарь.

```
q = Questionnaires.objects.get(id=id)
obj = {
'id': q.id,
'name': q.name,
'desc': q.desc
}
```

Рисунок 19 – Извлечение объекта анкеты по id

После этого собираются ответы из модели, и добавляются в словарь, как описано на рисунке 20.

```
    options = []
for o in QuestionnaireOptions.objects.all():
options.append({
'id': o.id,
'name': o.name,
'grade': o.grade,
})
obj['options'] = options
```

Рисунок 20 – Добавление вариантов ответов к объекту анкеты

После этого извлекаются все критерии для данной анкеты, из модели QuestionnaireCriteria, и также добавляются в словарь, это отображено на рисунке 21.

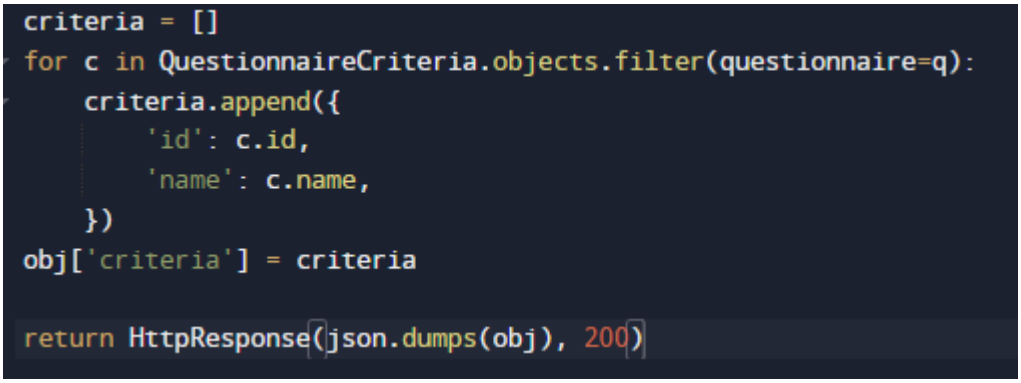

Рисунок 21 – Добавление критериев к объекту анкеты и возвращение словаря

После этого функция возвращает полученный словарь.

Функция сохранения анкет Этот код представляет собой Django представление (view), которое обрабатывает HTTP-запрос для сохранения результатов опроса.

В функцию передается запрос от приложения, после чего на основе данных из таблицы после чего происходит декодирование тела запроса в байтовую строку UTF-8, после чего производится парсинг объекта JSON для получения словаря.

После этого запись сохраняется в базе данных путем создания новой записи в модели QuestionnareResult, получая объект пользователя и его ID, получение объекта опросника и его ID, а также получения объекта пользователя, который заполняет опрос. Также передается оценка.

После этого, как представлено на рисунке 22, возвращается HTTP-ответ с сообщением «Результаты сохранены!» и статусом 200 (OK).

Полный код реализации API представлен в Приложении Б

```
def questionnaires_save(request):
obj = json.loads(str(request.body.decode('utf8')))
q = QuestionnaireResult.objects.create(
user=Users.objects.get(id=obj['user']),
questionnaire=Questionnaires.objects.get(id=obj['questionnaire']),
user_quest=Users.objects.get(id=obj['user_quest']),
rate=obj['rate'],
)
return HttpResponse("Результаты сохранены!", 200)
@gzip_page
@csrf_exempt
def questionnaires_create(request):
obj = json.loads(str(request.body.decode('utf8')))
q = Questionnaires.objects.create(
name=obj['name'],
desc=obj['desc']
)
for i in obj['items']:
c = QuestionnaireCriteria.objects.create(
questionnaire=q,
name=i,
)
return HttpResponse("Анкета успешно создана!", 200)
```

Рисунок 22 – Сохранение результатов и создание анкет

Функция questionnaire_options используется для получения списка оценок.

В начале работы функции создается пустой список.

После этого последовательно перебираются все строки из модели QuestionareOptions и передаются в список.

Функция questionnaires_save сохраняет результаты анкетирования, декодируя данные запроса и создавая соответствующий объект QuestionnaireResult

Функция questionnaires_create создает новую анкету и связанные с ней критерии, декодируя данные запроса и создавая объекты Questionnaires и QuestionnaireCriteria. Она также использует декораторы для сжатия и отключения проверки CSRF.

То, как функция возвращает HTTP-ответ с JSON-данными отображено на рисунке 23.

```
@gzip_page
@csrf_exempt
def questionnaire_options(request):
list = []
for i in QuestionnaireOptions.objects.all():
list.append({
'id': i.id,
'name': i.name,
'grade': i.grade,
})
return HttpResponse(json.dumps(list), 200)
```

Рисунок 23 – Получение и возврат вариантов ответов

Функция def questionnaire_criteria используется для получения списка критериев из БД, работает аналогичным образом, как представлено на рисунке 24.

```
@gzip_page
@csrf_exempt
def questionnaire_criteria(request, id):
list = []
for i in QuestionnaireCriteria.objects.get(id=id):
list.append({
'id': i.id,
'name': i.name,
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def questionnaire_result(request, id):
list = []
for i in QuestionnaireResult.objects.get(id=id):
list.append({
'id': i.id,
'user': i.u,
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def users(request):
list = []
for i in Users.objects.all():
list.append({
'id': i.id,
'login': i.login,
'role': i.role.name,
})
return HttpResponse(json.dumps(list), 200)
```

Рисунок 24 – Возврат данных и пользователей

Декораторы gzip_page и csrf_exempt, которые описаны на рисунке 25, получают и возвращает критерии, результаты и список пользователей анкеты по её `id` в формате JSON.

```
@gzip_page
@csrf_exempt
def users_create(request):
obj = json.loads(str(request.body.decode('utf8')))
l = Users.objects.filter(login=obj['login'])
if len(l) > 0:
return HttpResponse('Данный логин уже используется', status=404)
n = Users.objects.create(
login=obj['login'],
pasw=obj['pasw'],
role=Roles.objects.get(id=obj['role'])
)
return HttpResponse("Пользователь успешно создан!", 200)
```

Рисунок 25 - Создание нового пользователя с проверкой уникальности логина

Функция users_create предназначена для обработки запроса на создание нового пользователя в системе. «Она декодирует данные запроса, проверяет уникальность логина и создает новую запись в базе данных, если логин не занят» [16].

Если создание пользователя прошло успешно, возвращается HTTP-ответ с сообщением "Пользователь успешно создан!" и статусом 200

3.2 Реализация пользовательского интерфейса для сотрудников и руководителей

С увеличением числа мобильных устройств и приложений пользовательский интерфейс (UI) и пользовательский опыт (UX) выступают в роли основных драйверов успеха мобильных приложений. Качественный и интуитивно понятный UI/UX существенно влияют на удовлетворенность пользователей, их лояльность к приложению и его популярность на рынке. В данной статье мы обсудим, почему так важен пользовательский интерфейс и

опыт взаимодействия в мобильных приложениях и почему разработчикам стоит уделять этим аспектам особое внимание.

Эффективно разработанный пользовательский интерфейс облегчает использование приложения. Простота и интуитивность взаимодействий позволяют пользователям быстро адаптироваться и эффективно выполнять необходимые действия, что способствует удержанию клиентов и снижению оттока.

«Через хороший UI/UX разработчики могут повысить эффективность использования приложения. Оптимизация процессов, минимализм и простота взаимодействия помогают пользователям быстро достигать своих целей и выполнять задачи. Меньше времени, затраченного на ориентацию и поиск нужных функций, означает больше времени, проведенного в приложении и большую вероятность завершения конверсии» [17].

Список основных экранов, представленных в приложении отображен на рисунке 26.

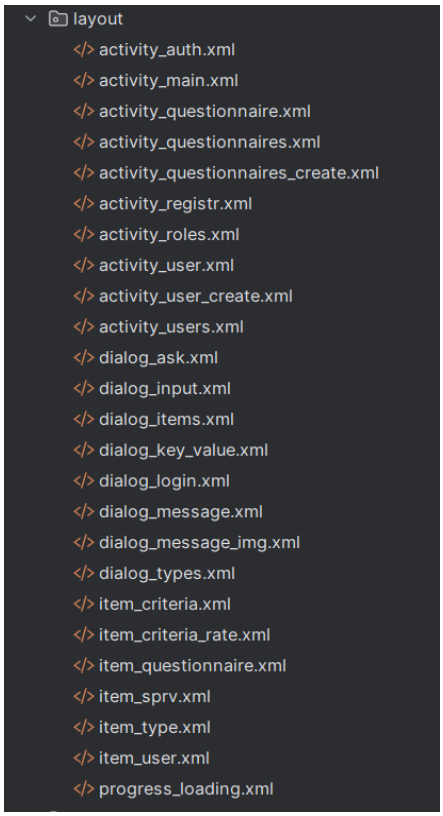

Рисунок 26 – Список экранов приложения

Назначение каждого из них следующее:

- activity_auth.xml – экран авторизации в системе;

- activity_main.xml – экран базовой активности, отвечает за главный экран приложения;

- activity_questionnaire.xml – экран вывода списка опросов в системе;

- activity_questionnaires.xml – экран анкеты пользователя;

- activity_questionnaires_create.xml – экран создания опроса;

- activity_registr.xml – экран активности регистрации в системе;

- activity_roles.xml – экран активности вывода ролей;

- activity_user.xml – экран активности вывода данных пользователя;

- activity_user_create.xml – экран активности создания пользователя;

- activity_users.xml – экран активности списка пользователей;

- dialog_ask.xml – элемент диалога вывода вопроса;

- dialog_input.xml – элемент диалога ввода данных;

- dialog_items.xml – элемент диалога вывода списка активностей;

- dialog_login.xml – элемент модуля авторизации в системе;

- dialog_message.xml – элемент вывода всплывающего сообщения;

- dialog_message_img.xml – сообщение с выводом картинки;

- item_criteria.xml – модуль вывода критерия;

- item_criteria_rate.xml – модуль оценки по критерию;

- item_questionnaire.xml – модуль элементов списка опросов;

- progress_loading.xml – модуль экрана загрузки приложения

Описание элементов пользовательского интерфейса реализован так:

Поля ввода данных (логин, пароль) на экране авторизации, всё это представлено на рисунке 27 и 28.

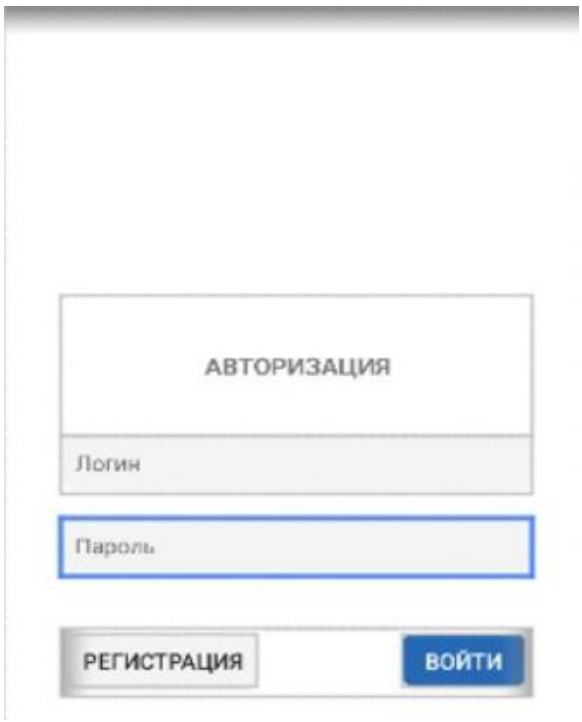

Рисунок 27 – Экран авторизации

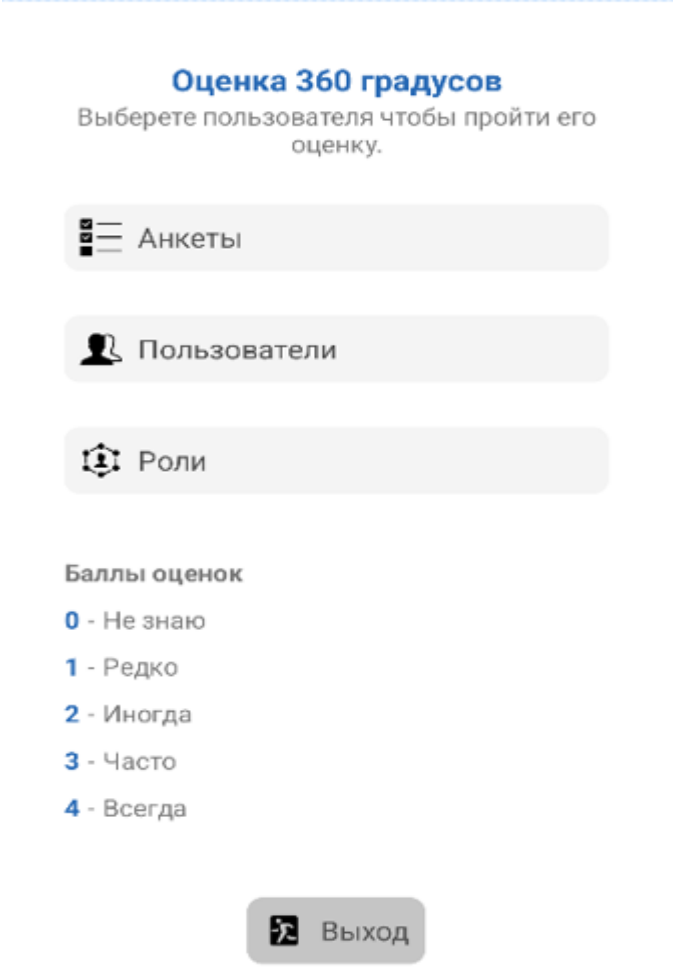

Рисунок 28 - Кнопки навигации и взаимодействия на главном экране

Важно учитывать обратную связь от пользователей для постоянного улучшения UI/UX. Проведение регулярных тестирований и опросов позволяет выявить слабые места в дизайне и функциональности приложения, что помогает вносить необходимые изменения и улучшения.

3.3 Разработка основных модулей приложения

При входе в приложении пользователя встречает первый экран авторизации, на котором ему необходимо авторизацию. Чтобы ее пройти ему нужно ввести данные в соответствующие поля.

Макет разработанного интерфейс представлен на рисунке 29.

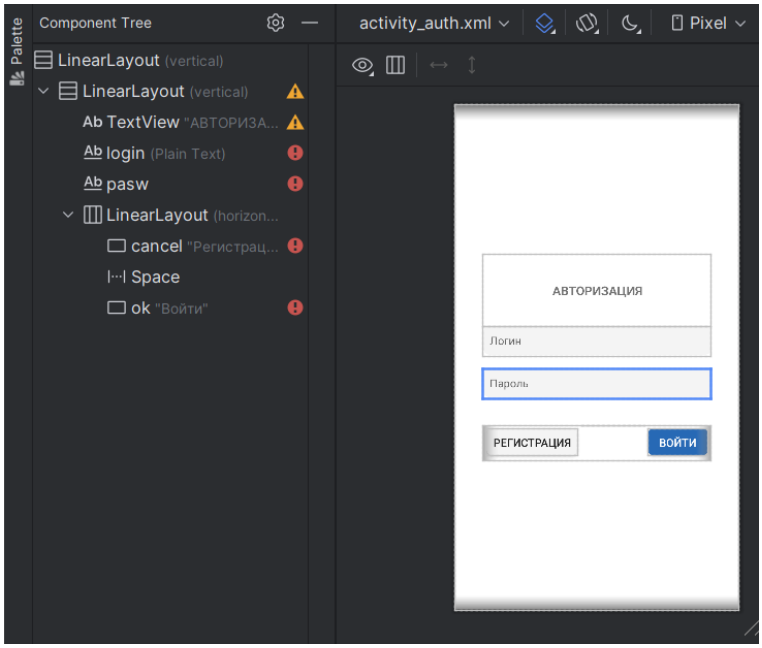

Рисунок 29 – Макет интерфейса «Вход в систему»

Для данного интерфейса была разработана активность Auth, которая принимает данные и производит проверку их на сервере.

В начале определяются базовые методы и свойства класса.

Процесс инициализации переменных отображен на рисунке 30.

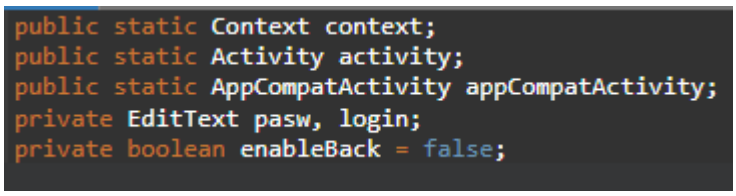

Рисунок 30 – Инициализация переменных

Затем определяются базовые методы класса, метод Oncreate переопределяет метод родительского класса, после чего в него передается сохраненное состояние активности. «После этого управление передается методу родительского класса, и устанавливается макет для текущей активности, показанный выше» [18].

После этого текущая активность присваивается соответствующим переменным в приложении., как представлено на рисунке 31.

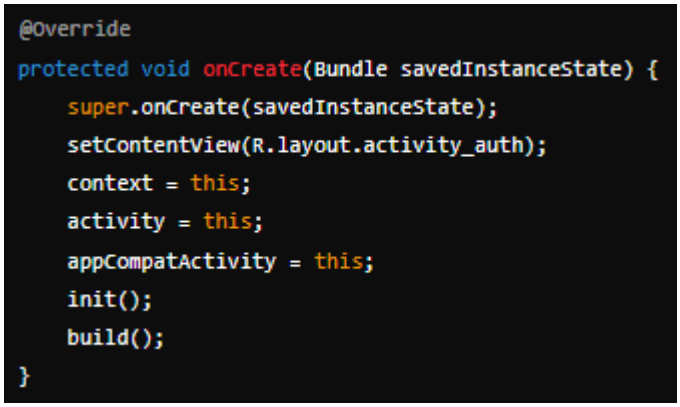

Рисунок 31 – Реализация метода onCreate

Метод onCreate вызывается при создании активности и выполняет инициализацию различных компонентов.

Метод OnClickAuth срабатывает после нажатия на кнопку входа в приложение.

Сначала в методе происходит приведение логина пользователя к нижнему регистру. После этого проводится проверка, не пустая ли это переменная, и если это так – происходит вывод соответствующего сообщения.

После этого метод аналогичным образом получает строку пароля и проверяет ее, выводя соответствующее сообщение если пароль не указан. После этого выполняется передача переменных в соответствующие методы, всё это описано на рисунке 32.

```
public void clickAuth(View view) {
    String l = login.getText().toString().strip().toLowerCase();
    if (l.isEmpty()) {
        Dialogs.msg(context, "Укажите логин!");
        return;
    }
    String p = pasw.getText().toString();
    if (p.isEmpty()) {
        Dialogs.msg(context, "Укажите пароль!");
        return;
    }
    LinkedTreeMap o = new LinkedTreeMap();
    o.put("login", l);
    o.put("pasw", p);
    new runAuth(o).execute();
}
```

Рисунок 32 - Обработка нажатия кнопки авторизации

Этот код реализует метод clickRegistr, который обрабатывает нажатие кнопки регистрации. Метод извлекает введенные пользователем данные, проверяет их на наличие ошибок и запускает процесс регистрации.

public void clickRegistr(View view)**:** Метод, который вызывается при нажатии кнопки регистрации. Параметр view представляет собой виджет, который был нажат.

LinkedTreeMap o = new LinkedTreeMap();**:** Создает новый объект LinkedTreeMap для хранения данных регистрации.

На форме входа в приложение также имеется кнопка «Регистрация», которая открывает форму регистрации и активность для нее.

Разработанный макет активности представлен на рисунке 33.

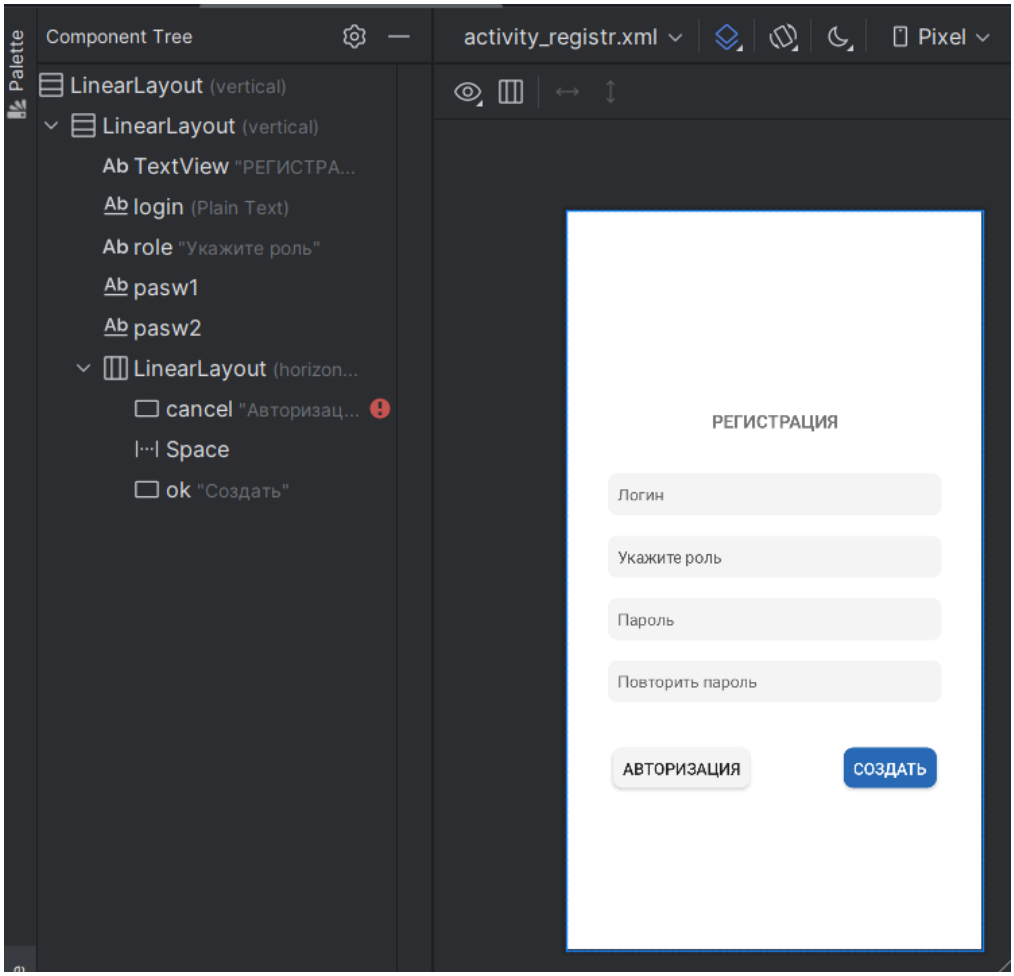

Рисунок 33 – Макет интерфейса регистрации

Для данного экрана разработана активность Registr.

Вначале происходит передача введенных пользователем данных в соответствующие переменные, рисунок 34.

```
private void init() {
    pasw1 = findViewById(R.id.pasw1);
    pasw2 = findViewById(R.id.pasw2);
    login = findViewById(R.id.login);
    role = findViewById(R.id.role);
}
```

Рисунок 34 – Инициализация UI компонентов

Затем производится проверка данных, введенных в окно регистрации, как представлено на рисунке 35, после чего они передаются в соответствующие методы

```
public void clickRegistr(View view) {
    String l = login.getText().toString().strip().toLowerCase();
    if (l.isEmpty()) {
        Dialogs.msg(context, "Укажите логин!");
        return;
    }
    if (selectedRole < 1) {
        Dialogs.msg(context, "Укажите роль!");
        return;
    }
    String p1 = pasw1.getText().toString();
    if (p1.isEmpty()) {
        Dialogs.msg(context, "Укажите пароль!");
        return;
    }
    String p2 = pasw2.getText().toString();
    if (p2.isEmpty()) {
        Dialogs.msg(context, "Повторите пароль!");
        return;
    }
    if (!Objects.equals(p1, p2)) {
        Dialogs.msg(context, "Пароли не совпадают!");
        return;
    }
    LinkedTreeMap o = new LinkedTreeMap();
    o.put("login", l);
    o.put("pasw", p1);
    o.put("role", selectedRole);
    new runRegistr(o).execute();
}
```

Рисунок 35 - Обработка нажатия кнопки регистрации

Этот код реализует метод clickRegistr, который обрабатывает нажатие кнопки регистрации. Метод извлекает введенные пользователем данные, проверяет их на наличие ошибок и запускает процесс регистрации.

Основная активность в системе activity_main содержит главную форму, отображаемую после авторизации в системе.

Макет данной активности выглядит как отображено на рисунке 36.

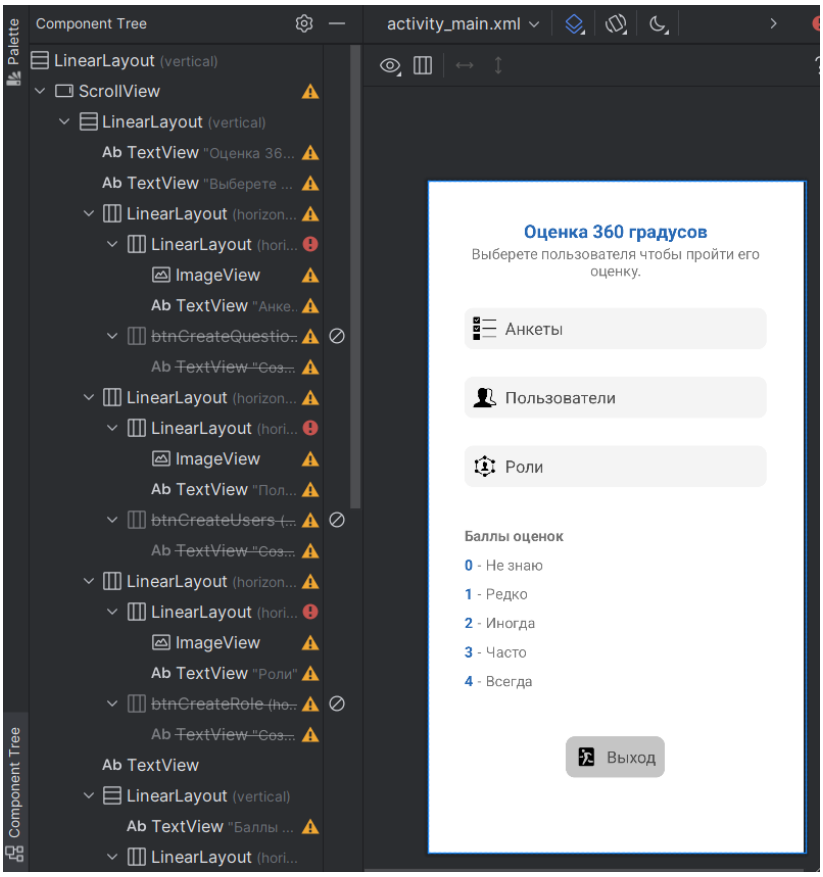

Рисунок 36 – Макет интерфейса главного окна

Для обработки данной активности разработан класс MainActivity, который описан на рисунке 37, в начале его работы определяются базовые методы и свойства.

```
public class MainActivity extends AppCompatActivity {
    public static Context context;
    public static Activity activity;
    public static AppCompatActivity appCompatActivity;
    public static SQLWork sqlWork;
    public static LinkedTreeMap user;
    public static boolean isAdmin = false;
}
```

Рисунок 37 - Инициализация статических переменных

Он определяет класс MainActivity, который наследует AppCompatActivity, и инициализирует несколько статических переменных.

При создании активности выполняется метод onCreate, который представлен на рисунке 38, в котором сначала происходит инициализация приложения и проверяются разрешения текущего пользователя

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    context = this;
    activity = this;
    appCompatActivity = this;
    Permissions.req(context, appCompatActivity);
    initID();
    buildView();
    String json = Settings.getStr(context, "user", null);
    if (json != null) {
        user = (LinkedTreeMap) Decoder.fromJson(json, null);
    }
    if (user == null) {
        startActivity(new Intent(context, Auth.class));
    }
}
```

Рисунок 38 - Инициализация главной активности

Он устанавливает макет активности, инициализирует контекст и активность, запрашивает разрешения, инициализирует идентификаторы, строит представление и проверяет, авторизован ли пользователь

protected void onCreate(Bundle savedInstanceState)**:** Метод, вызываемый при создании активности. Параметр savedInstanceState содержит сохраненное состояние активности, если оно было.

«Модуль User содержит форму с оценкой выбранного пользователя и кнопкой перехода к опросу» [19].

На рисунке 39 представлено окно разработки, которое отображает структуру компонента пользовательского интерфейса в файле activity_user.xml. Это окно отображает иерархию компонентов, используемых в данном макете, а также визуальное представление интерфейса приложения

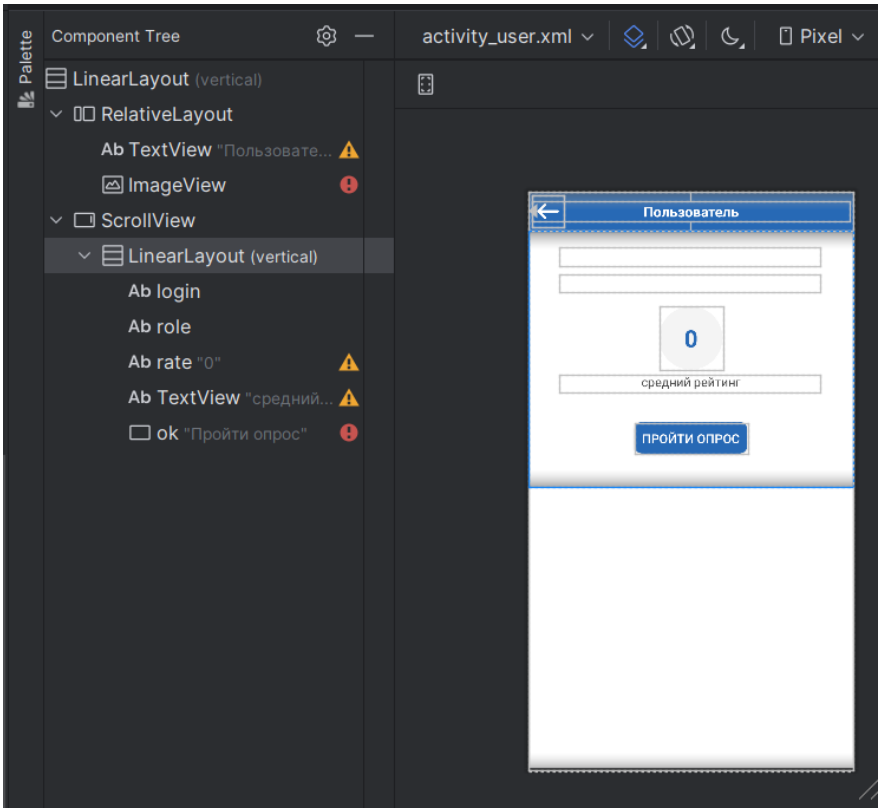

Рисунок 39 – Макет интерфейса вывода оценок пользователей

В левой части окна находится "Component Tree" (дерево компонентов), показывающее иерархию всех элементов интерфейса

В правой части окна отображается предварительный просмотр интерфейса.

Модуль ActivityQuestionarries, который отображен на рисунке 40, содержит список опросов в системе.

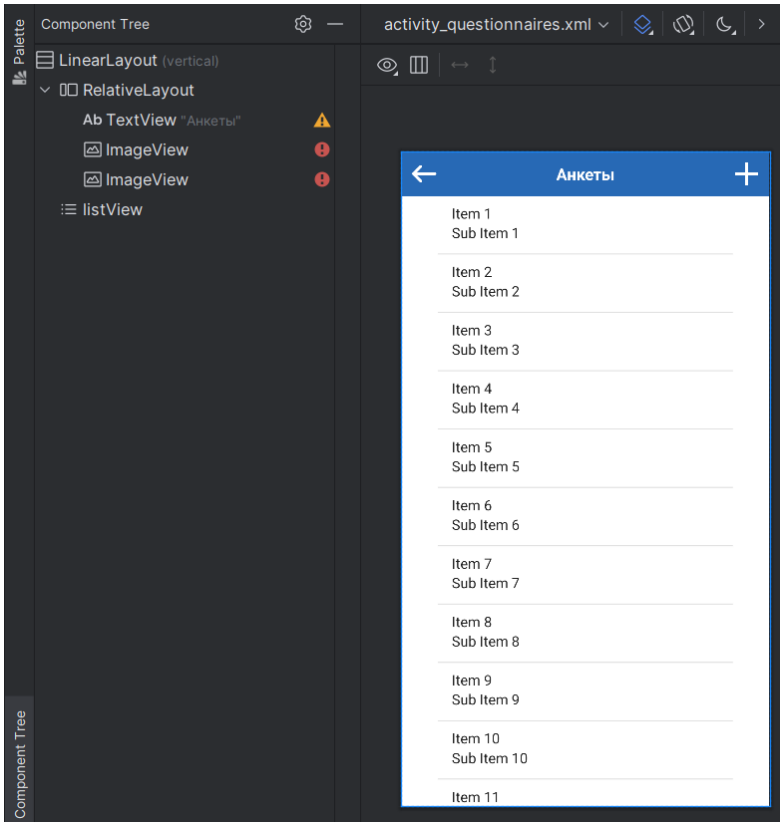

Рисунок 40 – Макет интерфейса списка анкет

На данном рисунке отображен список анкет, где их можно просматривать, создавать и редактировать.

3.4 Тестирование мобильного приложения

Заключительным этапом в разработке мобильного приложения является тестирование. Тестирование помогает оценить качество созданного решения и убедится, что оно удовлетворяет всем требованиям, которые были выявлены в подразделе 1.3.

Целью тестирования является проверка корректной работы всех процессов, которые предоставляет приложение, проверка корректности ввода данных в систему, и обработки исключительных ситуаций.

Объектами тестирования являются:

– вход в систему;
– авторизация;
– добавление опросов и критериев для них;

– функционал прохождения опросов;
– разграничение прав доступа для пользователей с различными ролями в системе.

После запуска приложения на экране отображается форма входа в систему, в которой можно пройти регистрацию, а также войти.

Проверим работу процедуры регистрации, которая выглядит так, как представлено на рисунке 41.

Форма регистрации выглядит следующим образом.

В форме не допускается ввод русских символов в поля ввода логина и пароля.

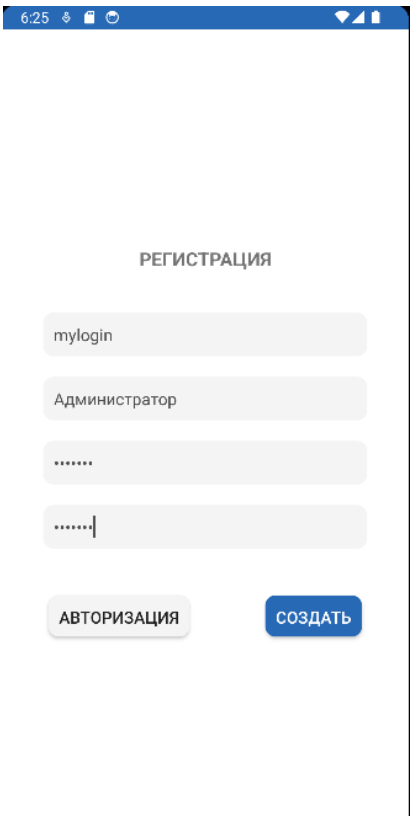

Рисунок 41 – Форма регистрации

При вводе различных паролей в форме регистрации – система выводит сообщение о том, что пароли не совпадают, таким образом обеспечивается проверка правильности ввода пароля, данная ошибка выводится так, как отображено на рисунке 42.

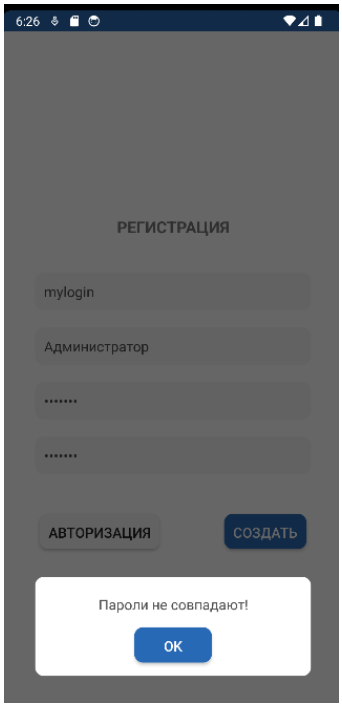

Рисунок 42 – Сообщение системы

После регистрации система выводит сообщение о выводе уведомлений для данного пользователя, как представлено на рисунке 43.

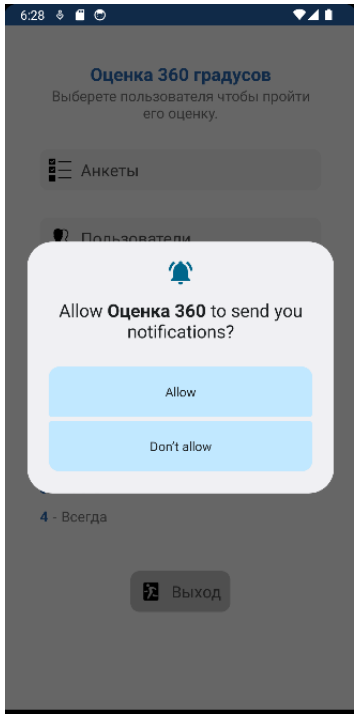

Рисунок 43 – Сообщение о выводе уведомлений

При правильном вводе пароля, на экран выводится интерфейс, который выглядит так, как отражено на рисунке 44.

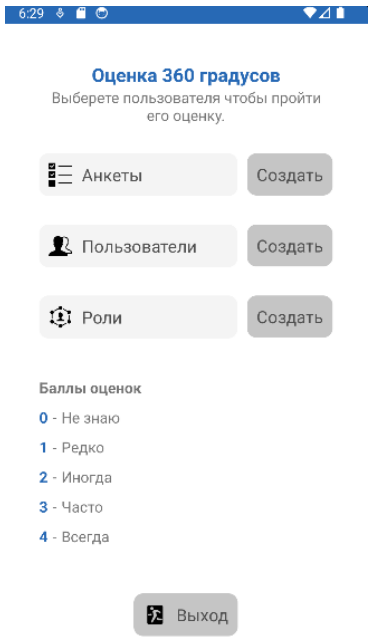

Рисунок 44 – Основной интерфейс приложения

В данном для пользователей с ролями «Администратор» и «Менеджер» доступно создание анкет и редактирование ролей, при необходимости.

Пользователям с ролью «Сотрудник» доступно только прохождение опросов.

Проверим функционал добавления опросов в систему.

Выбираем в главном окне «Анкеты» и нажимаем «Создать». Выводится окно создания опросника, в котором необходимо ввести название и описание опросника, это отражено на рисунке 45.

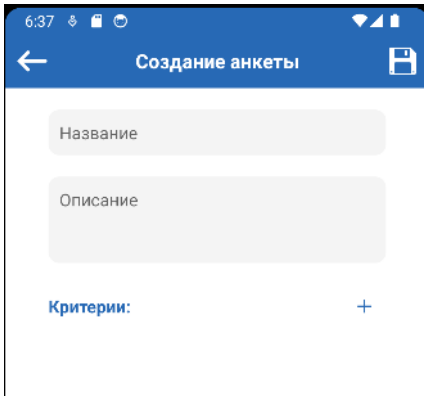

Рисунок 45 – Интерфейс создания анкеты

Если поля оставить пустыми и нажать на кнопку сохранения, то выдается сообщение о необходимости ввода данных, как представлено на рисунке 46.

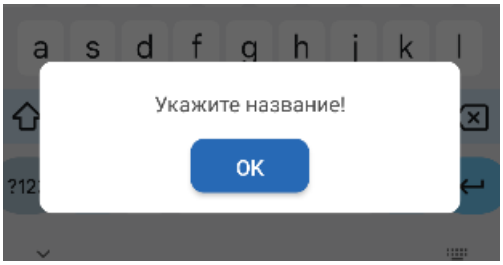

Рисунок 46 – Ошибка ввода данных

Здесь же, как показано на рисунке 47, администратор может добавить критерии для прохождения опроса.

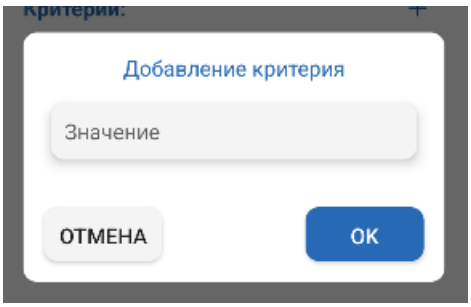

Рисунок 47 – Добавление критериев

После ввода критерия он отображается в окне приложения. Также доступно удаление критериев и добавление новых, это отображено на рисунке 48.

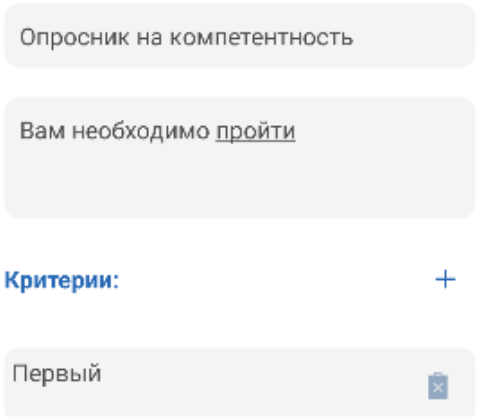

Рисунок 48 – Отображение критериев

После ввода критериев и значений – опросник сохраняется в системе и отображается в списке опросов, рисунок 49.

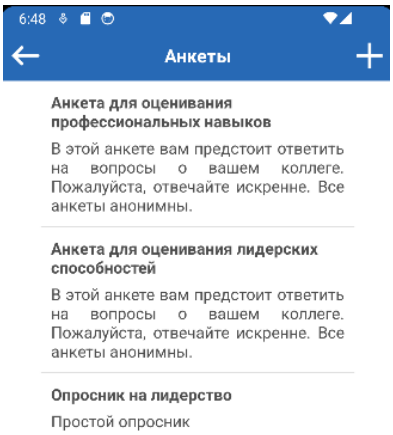

Рисунок 49 – Отображение анкеты-опросника

На рисунке 50 отображено, что пользователям с ролью «Сотрудник» недоступно редактирование анкет, при попытке редактирования выдается сообщение об ошибке.

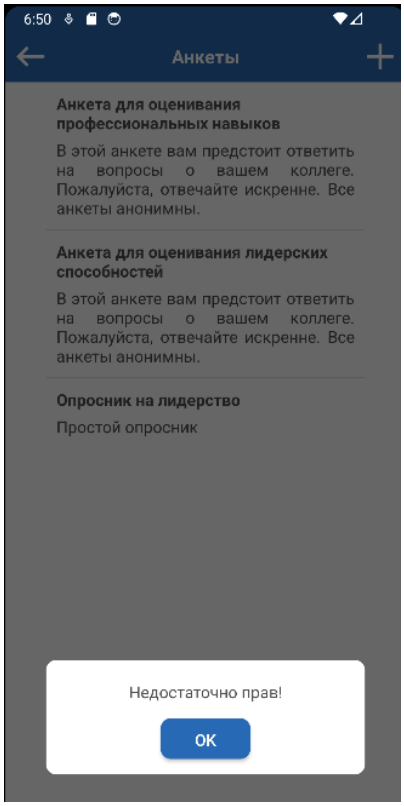

Рисунок 50 – Ошибка об отсутствии прав

При нажатии на кнопку «Пользователи» - можно посмотреть список текущих пользователей в системе, как описано на рисунке 51.

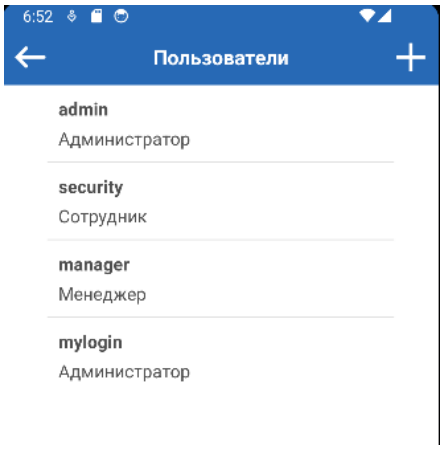

Рисунок 51 – Список пользователей в системе

После нажатия на пункт пользователя – выводится окно с его средним рейтингом, который отражен на рисунке 52.

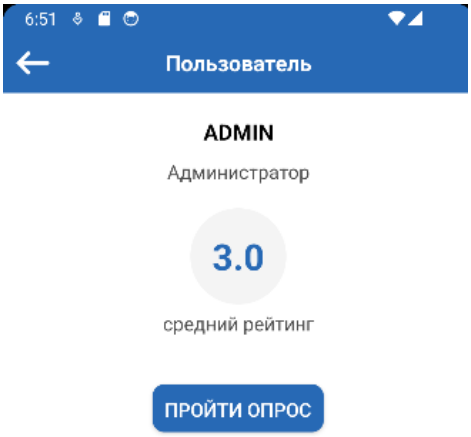

Рисунок 52 – Отображение среднего рейтинга

При нажатии на кнопку «Пройти опрос», который представлен на рисунке 53 - выводится окно выбора анкеты для данного пользователя.

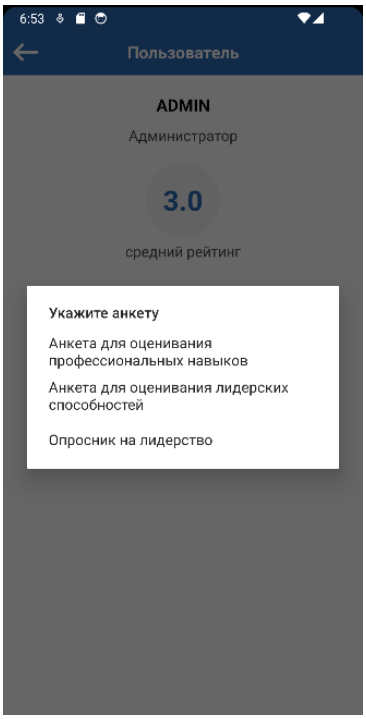

Рисунок 53 – Окно выбора анкеты

После выбора анкеты – выводится окно прохождения опроса, это отображено на рисунке 54.

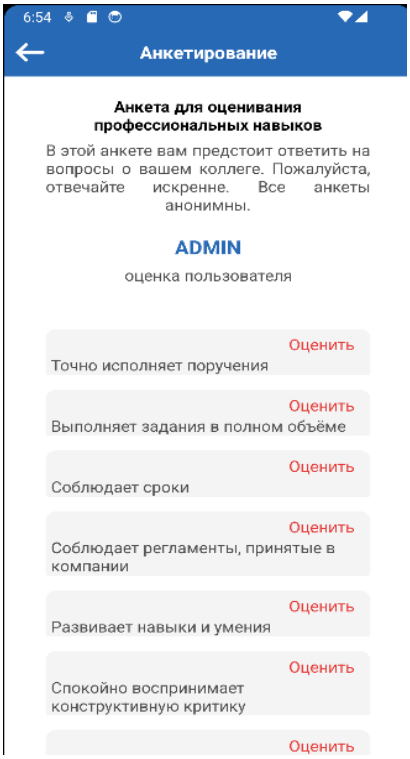

Рисунок 54 – Окно прохождения опроса

На рисунке 55, отображено, что здесь можно оценивать сотрудников

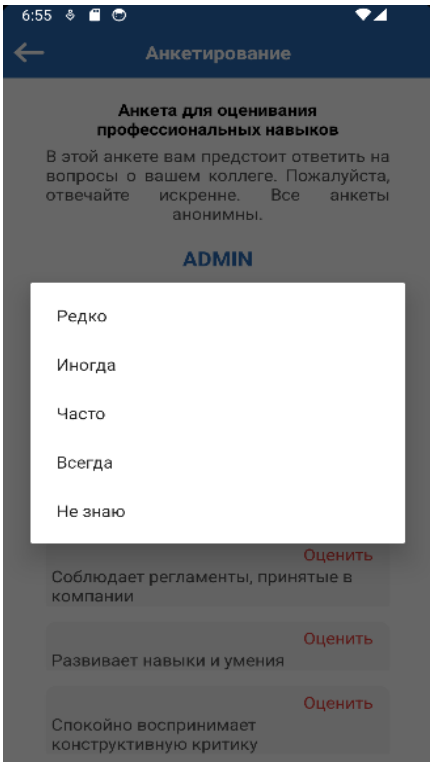

Рисунок 55 - Оценка по критериям

После прохождения опросника выводится сообщение, которое отображено на рисунке 56, о сохранении результатов.

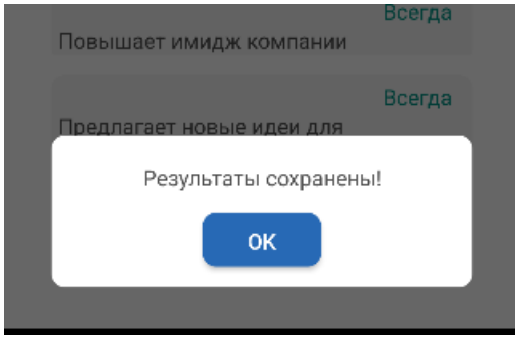

Рисунок 56 – Сообщение о сохранении результатов

По окончании тестирования сведем все проверенные функции в общую таблицу (Таблица 2).

Для анализа проведенного тестирования полученные результаты занесем в таблицу 2.

Таблица 2 – результаты тестирования приложения

Номер теста	Описание	Ожидаемый результат	Полученный результат	Статус
1	Тестирование вариантов использования	Вывод соответствующих сообщений при вводе некорректных данных	При авторизации, регистрации и заполнении тестов – в случае ввода некорректных данных выводится соответствующее сообщение	Пройден
2	Тестирование функционала прохождения опросов	Пользователи имеют возможность проходить опросы в системе и результаты опросов сохраняются в системе	Доступно прохождение опросов, данные сохраняются и вычисляются корректно	Пройден
3	Тестирование функционала добавления опросов в систему	Опросы корректно добавляются в систему и доступны для прохождения	Опросы добавляются в системы, данные сохраняются	Пройден

Продолжение таблицы 2.

Номер теста	Описание	Ожидаемый результат	Полученный результат	Статус
4	Тестирование функционала разграничения прав доступа для пользователей	Для каждого из пользователей доступен только функционал, определенный его ролью	Полный доступ к системе имеет только администратор, менеджер имеет возможность просматривать результаты опросов, пользователь – только проходить анкетирование	Пройден

Полный код реализации методов в AtivityMain представлен в Приложении А

Таким образом, протестировав весь функционал приложения, можно сделать вывод, что разработанное мобильное приложение и серверная часть успешно решают задачу оценки сотрудников компании по методике «360 градусов». Пользователи имеют возможность проходить оценку в приложении, как самооценку, так и оценку других сотрудников, руководитель имеет доступ к полному функционалу, включая просмотр результатов опросов, для менеджера доступен только просмотр результатов.

В третьей главе был описан процесс разработки и тестирования мобильного приложения. Реализованы основные функциональные модули, включая регистрацию и аутентификацию пользователей, создание и проведение опросов, сбор и анализ результатов. Приведены примеры кода, иллюстрирующие работу различных компонентов системы. Проведено тестирование приложения на различных этапах разработки для обеспечения его корректной работы и соответствия требованиям. В ходе тестирования выявлены и устранены ошибки, проведена оптимизация производительности. Таким образом, приложение готово к использованию и может эффективно поддерживать процесс оценки сотрудников по методике «360 градусов».

Заключение

Прохождение опросов в организации по методике «360 градусов» позволяет оценить основные качества сотрудников, их мотивацию на выполнение текущих задач, а также другие профессиональные компетенции с точки зрения самих сотрудников предприятия, а также руководство. Подобная информация может послужить директору и менеджерам предприятия основой для разработки методик повышения мотивации сотрудников, а также улучшения психологического климата в коллективе, а также кадровых решений.

В процессе выполнения работы по разработке мобильного приложения для управления процессом оценивания сотрудников были достигнуты всех поставленных во введении целей и задач.

Был проведен анализ предметной области задачи и организационной структуры предприятия, который позволил выявить слабые стороны текущего процесса оценивания сотрудников на предприятии. В результате характеристики этого процесса для реализации оценивания была выбрана методика «360 градусов».

Были определены основные функциональные и нефункциональные требования к приложению для оценивания сотрудников, выявлены потребности пользователей и определены варианты использования системы.

Была проведена разработка архитектуры приложения, определен состав основных компонентов и модулей.

Далее была разработана схема данных приложения, и определены соответствующие технологии, а именно Android Studio для реализации клиентской части, и СУБД SQLite для реализации базы данных. В качестве средства реализации серверной части была выбрана платформа Node.js

Спроектирована логическая структура, на основе которой было разработано мобильное приложение для платформы Android, которое предоставило пользователям проходить оценку компетенций по методике «360 градусов», а руководителю удобный интерфейс добавления в систему новых опросов и просмотра оценок пользователей.

Проведено всестороннее, чтобы убедиться, что приложение удовлетворяет

всем требованиям и решает поставленные задачи. Тестирование проводилось с учетом разных вариантов использования.

Весь процесс разработки проводился с учетом потребностей клиентов, что позволило создать удобный функционал, который будет приятен в использовании для любого пользователя.

Таким образом, проведенная работа имеет значительное практическое, научное и экономическое значение, а полученные результаты могут быть использованы для проведения оценки сотрудников по методике «360 градусов», что позволит улучшить показатели предприятия.

Список использованной литературы и источников

1. Важность пользовательского интерфейса и опыта взаимодействия в мобильном приложении [Электронный ресурс]. URL: https://vc.ru/u/1096055-ahmedov-behzod/773761-vazhnost-polzovatelskogo-interfeisa-i-opyta-vzaimodeistviya-v-mobilnom-prilozhenii?ysclid=lxpvi2izua952116229 (дата обращения: 21.06.2024).
2. Коновалова В.Г. Организация отбора и оценки персонала. – М.: Экзамен, 2007. – 112 с.
3. Медведев А.А. Разработка мобильных приложений на платформе Android. СПб.: Питер, 2020. – 320 с.
4. Описание бизнес-процессов [Электронный ресурс]. URL: https://trinion.org/blog/opisanie-biznes-processov-kak-est-as-is-i-kak-dolzhno-byt-to-be (дата обращения: 21.06.2024).
5. Пользовательские сценарии [Электронный ресурс]. URL: https://netology.ru/blog/users-scenarios (дата обращения: 21.06.2024).
6. Портной В.В., Козлов В.С. Разработка мобильных приложений на Java для Android: Учебное пособие. – М.: Инфра-М, 2021. – 256 с.
7. Проектирование и разработка мобильного приложения [Электронный ресурс]. URL: https://elib.sfukras.ru/bitstream/handle/2311/34120/diplom_v1.1.pdf?sequence (дата обращения: 21.06.2024).
8. Сошников А., Пеленицын А. Оценка персонала. Психологические и психофизические методы. – М.: Эксмо, 2009. – 97 с.
9. Тарасов С.В. СУБД для программиста.
10. Уорд П. Метод 360 градусов. – М.: Hippo Publishing LTD, 2006. – 64 с.
11. Что такое Android Studio [Электронный ресурс]. URL: http://web.spt42.ru/index.php/chto-takoe-android-studio (дата обращения: 21.06.2024).
12. Что такое Django? Python&Back-End [Электронный ресурс]. URL: https://exceedteam.gitbook.io/python-and-back-end/django/untitled (дата

обращения: 21.06.2024)

13. Что такое цифровизация и зачем она нужна [Электронный ресурс]. URL: https://dis-group.ru/blogs/czifrovizacziya-chto-eto-takoe-prostymi-slovami/ (дата обращения: 21.06.2024).

14. APScheduler library user guide [Электронный ресурс]. URL: https://apscheduler.readthedocs.io/en/stable/userguide.html (дата обращения: 21.06.2024).

15. Gast H. Объектно-ориентированное проектирование: концепции и программный код / Х. Гаст. - М.: Диалектика, 2018. - 1040 с.

16. Goodwill M. Разработка приложений для Android на языке Java для чайников. – М.: Диалектика, 2021. – 384 с.

17. Java + Visual Studio Code [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/visual-studio-code/ (дата обращения: 21.06.2024).

18. Java 3.10.4 documentation [Электронный ресурс] URL: https://docs.java.org/3.10 (дата обращения: 21.06.2024).

19. Nilson M., Smolin L., Enns D. Android Studio 4.0 Development Essentials - Java Edition: Developing Android 10 (Q) Apps Using Android Studio 4.0, Java and Android Jetpack. – Payload Media, 2020. – 820 с.

20. Phillips B., Hardy B., Marsicano K. Android Programming: The Big Nerd Ranch Guide. – 4-е издание. – Addison-Wesley Professional, 2019. – 624 с.

21. Schildt H. Java: Полное руководство, 11-е издание. – М.: Вильямс, 2019. – 1488 с.

22. TOML documentation [Электронный ресурс] URL: https://github.com/toml-lang/toml/blob/main/README.md (дата обращения: 21.06.2024).

23. Visual Studio Code [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/visual-studio-code/ (дата обращения: 21.06.2024).

24. Zed A. Shaw Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code, First Edition, 2017. – 320 p.

Приложение А

Листинг основного модуля ActivityMain

```
package org.angry.workrate;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.LinearLayout;
import androidx.appcompat.app.AppCompatActivity;
import com.google.gson.internal.LinkedTreeMap;
import org.angry.workrate.activitys.Auth;
import org.angry.workrate.activitys.Questionnaires;
import org.angry.workrate.activitys.QuestionnairesCreate;
import org.angry.workrate.activitys.Roles;
import org.angry.workrate.activitys.UserCreate;
import org.angry.workrate.activitys.Users;
import org.angry.workrate.database.SQLWork;
import org.angry.workrate.database.Settings;
import org.angry.workrate.standart.Decoder;
import org.angry.workrate.standart.Dialogs;
import org.angry.workrate.standart.Loading;
import org.angry.workrate.standart.Permissions;
import org.jsoup.Connection;
import org.jsoup.Jsoup;
public class MainActivity extends AppCompatActivity {
public static Context context;
public static Activity activity;
public static AppCompatActivity appCompatActivity;
```

```
public static SQLWork sqlWork;
public static LinkedTreeMap user;
public static boolean isAdmin = false;
private LinearLayout btnCreateRole, btnCreateUsers, btnCreateQuestionnaire;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
context = this;
activity = this;
appCompatActivity = this;
Permissions.req(context, appCompatActivity);
initID();
buildView();
String json = Settings.getStr(context, "user", null);
if(json != null){
user = (LinkedTreeMap) Decoder.fromJson(json, null);
}
if(user == null){
startActivity(new Intent(context, Auth.class));
}
}
@SuppressLint("MissingSuperCall")
@Override
public void onBackPressed() {
moveTaskToBack(true);
}
@Override
protected void onResume() {
super.onResume();
```

```
updateView();
}
public void exit(View view){
Dialogs.ask(context, "Выйти с аккаунта?", v -> {
Settings.setStr(context, "user", null);
isAdmin = false;
user = null;
startActivity(new Intent(context, Auth.class));
}, null);
}
private void updateView(){
if(user == null) return;
isAdmin =
String.valueOf(user.get("role_name")).equalsIgnoreCase("Администратор");
btnCreateRole.setVisibility(isAdmin ? View.VISIBLE : View.GONE);
btnCreateUsers.setVisibility(isAdmin ? View.VISIBLE : View.GONE);
btnCreateQuestionnaire.setVisibility(isAdmin ? View.VISIBLE :
View.GONE);
}
private void initID(){
btnCreateRole = findViewById(R.id.btnCreateRole);
btnCreateUsers = findViewById(R.id.btnCreateUsers);
btnCreateQuestionnaire = findViewById(R.id.btnCreateQuestionnaire);
}
private void buildView(){
sqlWork = new SQLWork(context);
}
public void openRoles(View view){
startActivity(new Intent(context, Roles.class));
}
public void openQuestionnaires(View view){
```

```
startActivity(new Intent(context, Questionnaires.class));
}
public void openUsers(View view){
startActivity(new Intent(context, Users.class));
}
public void createQuestionnaire(View view){
startActivity(new Intent(context, QuestionnairesCreate.class));
}
public void createUser(View view){
startActivity(new Intent(context, UserCreate.class));
}
public void createRole(View view){
Dialogs.InputObj inputObj = Dialogs.input(context, "Добавление роли", "Наименование");
inputObj.cancel.setOnClickListener(v -> inputObj.alertDialog.cancel());
inputObj.ok.setOnClickListener(v -> {
String n = inputObj.text.getText().toString().trim();
if(n.isEmpty()){
Dialogs.msg(context, "Укажите наименование!");
return;
}
new roleCreate(n).execute();
inputObj.alertDialog.cancel();
});
inputObj.alertDialog.show();
}
public class roleCreate extends AsyncTask<String, Integer, Connection.Response> {
private String n;
public roleCreate(String n){
```

```
Loading.start(context);
this.n = n;
}
@Override
protected Connection.Response doInBackground(String... strings) {
try{
return Jsoup.connect(BuildConfig.api + "roles/create/")
.ignoreContentType(true)
.ignoreHttpErrors(true)
.method(Connection.Method.POST)
.requestBody(n)
.execute();
} catch (Exception ex){
return null;
}
}
@Override
protected void onPostExecute(Connection.Response response) {
Loading.destroy();
if(response != null){
if (response.statusCode() == 200){
Dialogs.msg(context, "Роль успешно создана!");
} else {
Dialogs.msg(context, response.body());
}
} else {
Dialogs.msg(context, "Произошла ошибка!");
}
super.onPostExecute(response);
this.cancel(true);
```

Приложение Б

Листинг разработанного API

```
import json
from django.db.models import Avg
from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt
from django.views.decorators.gzip import gzip_page
from datetime import datetime
from ocenka360 import creator, gmethod
from ocenka360.models import Users, Roles, Questionnaires, QuestionnaireOptions, QuestionnaireCriteria, \
QuestionnaireResult
def http_error():
return HttpResponse('Произошла ошибка', content_type='text/html; charset=utf-8', status=404)
def index(request):
return HttpResponse("API")
@gzip_page
@csrf_exempt
def auth(request):
obj = json.loads(str(request.body.decode('utf8')))
try:
o = Users.objects.get(login=obj['login'], pasw=obj['pasw'])
except:
o = None
if not o:
return http_error()
u = {
'id': o.id,
'login': o.login,
'pasw': o.pasw,
```

```
'role': o.role.id,
'role_name': o.role.name,
}
return HttpResponse(json.dumps(u), 200)
@gzip_page
@csrf_exempt
def registr(request):
obj = json.loads(str(request.body.decode('utf8')))
l = Users.objects.filter(login=obj['login'])
if len(l) > 0:
return HttpResponse('Данный логин уже используется', status=404)
n = Users.objects.create(
login=obj['login'],
pasw=obj['pasw'],
role=Roles.objects.get(id=obj['role'])
)
u = {
'id': n.id,
'login': n.login,
'pasw': n.pasw,
'role': n.role.id,
'role_name': n.role.name,
}
return HttpResponse(json.dumps(u), 200)
@gzip_page
@csrf_exempt
def roles(request):
list = []
for i in Roles.objects.all():
list.append({
```

```
'id': i.id,
'name': i.name
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def role_create(request):
r = Roles.objects.create(
name=str(request.body.decode('utf8'))
)
return HttpResponse("Роль успешно создана!", 200)
@gzip_page
@csrf_exempt
def questionnaires(request):
list = []
for i in Questionnaires.objects.all():
list.append({
'id': i.id,
'name': i.name,
'desc': i.desc,
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def questionnaires_info(request, id):
q = Questionnaires.objects.get(id=id)
obj = {
'id': q.id,
'name': q.name,
'desc': q.desc
```

```
options = []
for o in QuestionnaireOptions.objects.all():
options.append({
'id': o.id,
'name': o.name,
'grade': o.grade,
})
obj['options'] = options
criteria = []
for c in QuestionnaireCriteria.objects.filter(questionnaire=q):
criteria.append({
'id': c.id,
'name': c.name,
})
obj['criteria'] = criteria
return HttpResponse(json.dumps(obj), 200)
@gzip_page
@csrf_exempt
def questionnaires_save(request):
obj = json.loads(str(request.body.decode('utf8')))
q = QuestionnaireResult.objects.create(
user=Users.objects.get(id=obj['user']),
questionnaire=Questionnaires.objects.get(id=obj['questionnaire']),
user_quest=Users.objects.get(id=obj['user_quest']),
rate=obj['rate'],
)
return HttpResponse("Результаты сохранены!", 200)
@gzip_page
@csrf_exempt
def questionnaires_create(request):
```

```
obj = json.loads(str(request.body.decode('utf8')))
q = Questionnaires.objects.create(
name=obj['name'],
desc=obj['desc']
)
for i in obj['items']:
c = QuestionnaireCriteria.objects.create(
questionnaire=q,
name=i,
)
return HttpResponse("Анкета успешно создана!", 200)
@gzip_page
@csrf_exempt
def questionnaire_options(request):
list = []
for i in QuestionnaireOptions.objects.all():
list.append({
'id': i.id,
'name': i.name,
'grade': i.grade,
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def questionnaire_criteria(request, id):
list = []
for i in QuestionnaireCriteria.objects.get(id=id):
list.append({
'id': i.id,
'name': i.name,
```

```
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def questionnaire_result(request, id):
list = []
for i in QuestionnaireResult.objects.get(id=id):
list.append({
'id': i.id,
'user': i.u,
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def users(request):
list = []
for i in Users.objects.all():
list.append({
'id': i.id,
'login': i.login,
'role': i.role.name,
})
return HttpResponse(json.dumps(list), 200)
@gzip_page
@csrf_exempt
def users_create(request):
obj = json.loads(str(request.body.decode('utf8')))
l = Users.objects.filter(login=obj['login'])
if len(l) > 0:
return HttpResponse('Данный логин уже используется', status=404)
```

```
n = Users.objects.create(
login=obj['login'],
pasw=obj['pasw'],
role=Roles.objects.get(id=obj['role'])
)
return HttpResponse("Пользователь успешно создан!", 200)
@gzip_page
@csrf_exempt
def users_info(request, id):
u = Users.objects.get(id=id)
r = QuestionnaireResult.objects.filter(user_quest=u)
obj = {
'role': u.role.name,
'login': u.login,
'rate': 0 if len(r) < 1 else r.aggregate(Avg('rate'))['rate__avg'],
}
return HttpResponse(json.dumps(obj), 200)
```