Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

«ДОПУЩЕН К ЗАЩИТЕ»
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На тему: Исследование и разработка метода, моделей и алгоритмов приложений для мобильных устройств

Специальность 6М070300 – Информационные системы

Выполнил: Ким Д.В., группа МИСн-17-1 _____
Научный руководитель: PhD, доцент Кожамкулова Ж.Ж._____

Консультанты:

По применению
вычислительной техники: ст. преп. Айткулов Ж.С.
(ученая степень, звание, Ф.И.О.)
_____ «17» 05 201___г.
(подпись)

Нормоконтролер: ст. преп. Алимсеитова Ж.К.
(ученая степень, звание, Ф.И.О.)
_____ «24» 05 201___г.
(подпись)

Рецензент: к.ф.н., Қонысбаев Ә.Т_
(ученая степень, звание, Ф.И.О.)
_____ «____» _____201___г.
(подпись)

Алматы, 2019

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг
Специальность 6М070300 – Информационные системы

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Ким Дмитрию Валерьевичу

Тема диссертации: Исследование и разработка метода, моделей и алгоритмов приложений для мобильных устройств
Утверждена приказом по университету №161 от «23» 10 2017 г.

Срок сдачи законченной диссертации «27» 05 2019 г.

Исходные данные к диссертации (требуемые параметры результатов исследования (проектирования) и исходные данные объекта):

- SQL
- Pl SQL.
- Oracle Database 11g Release 1 (11.1.0.6).
- WEB API.
- C#

Перечень вопросов, подлежащих к разработке в диссертации, или краткое содержание диссертации:

- Анализ мобильных приложений и тенденции развития
- Компоненты, методы и основы дизайна мобильного приложения
- Исследование и создание системы оценки персонала
- Подготовка и реализация API для мобильного приложения использование метода WEBAPI

Основная рекомендуемая литература:

1. Jacobson, Daniel; Woods, Dan; Brail, Greg (November 2011). APIs a strategy guide. Sebastopol, Calif: O'Reilly Media.

2. Эндрю Троелсен. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание = Pro C# 5.0 and the .NET 4.5 Framework, 6th edition. — М.: «Вильямс», 2013. — 1312 с.

3. Джозеф Албахари, Бен Албахари. C# 6.0. Справочник. Полное описание языка = C# 6.0 in a Nutshell: The Definitive Reference. — М.: «Вильямс», 2018. — 1040 с.

4. Коньшин С.В., Кондратович А.П. Учебно –методические и учебные работы. Общие требования к построению, изложению, оформлению и содержанию учебно-методических и учебных работ

Консультации по диссертации с указанием относящихся к ним разделов диссертации

Раздел	Консультант	Сроки	Подпись
Компоненты, методы и основы дизайна мобильного приложения	Кожамкулова Ж.Ж.	15.10.2018	
Исследование и создание системы оценки персонала	Кожамкулова Ж.Ж.	07.01.2019	
Подготовка и реализация API для мобильного приложения использование метода WEBAPI	Кожамкулова Ж.Ж.	20.02.2019	
Нормоконтролер	Алимсеитова Ж.К.	24.05.19	
Разработка ПО	Айткулов Ж.С	17 05 19	

График подготовки диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководству	Примечание
Компоненты, методы и основы дизайна мобильного приложения	15.10.2018	выполнено
Исследование и создание системы оценки персонала	07.01.2019	выполнено
Подготовка и реализация API для мобильного приложения использование метода WEBAPI	20.02.2019	выполнено

Дата выдачи задания «4» 09 2017 г.

Заведующий кафедрой _____ Картбаев Т.С.

Научный руководитель диссертации _____ Кожамкулова Ж.Ж..

Задание принял к исполнению магистрант _____ Ким Д.В.

Аңдатпа

Диссертациялық жұмыста мобильді қосымшалардың әдістерін және құралдарын зерттеп, зерттеді. Мобильді қосымшаларды талдау, нарықтың ағымдағы жағдайы және олардың даму тенденциялары. Мобильдік қосымшаларды әзірлеуге арналған әртүрлі әдістер мен алгоритмдер, оның ішінде ана, гибридтік және веб-сайттар қарастырылған.. Архитектура базасы және бизнес-процесті таныстыру үдерісінде персоналдың бағалау жүйесі. Веб-API-ны таңдауға арналған әдістер мен кеңестерді әзірлеуге арналған PL SQL, C # сияқты бағдарламалау тілдерін қолдану.

Аннотация

В диссертационной работе рассмотрены и проанализированы методы и средства создания мобильных приложений. Проведен анализ мобильных приложений, текущее состояние рынка и тенденции их развития. Рассмотрены различные методы и алгоритмы разработки мобильных приложений, в том числе нативный, гибридный и Web. Разрботан API для системы оценки персонала, создана архитектура базы и прописан бизнес процесс. В качестве метода и среды разработки выбран Web API с использованием таких языков программирования, как PL SQL, C#.

Annotation

The dissertation work has been reviewed and analyzed by the methods and tools of mobile applications. The analysis of mobile applications, the current state of the market and their development trends. Various methods and algorithms for developing mobile applications are considered, including native, hybrid, and Web.. The benefits of the API for the system of staff evaluation, the architecture bases and business process. In the method of layout and development of the Web API, use the same programming language as PL SQL, C #.

Содержание

Введение 6
1 Анализ мобильных приложений и тенденции развития 8
1.1 Основы мобильных приложений 8
1.2 Развитие приложений 11
1.3 Сравнение мобильного приложения с веб-приложением 16
1.4 Мобильные операционные системы 19
2 Компоненты, методы и основы дизайна мобильного приложения 26
2.1 Архитектура мобильного приложения 26
2.2 Компоненты мобильного клиента 37
2.3 Основы дизайна мобильного приложения 41
2.4 Жизненный цикл проекта мобильного приложения 43
3 Стандарты безопасности 44
4 Исследование и создание системы оценки персонала 47
5 Построение архитектуры базы системы оценки персонала и создание WEB API 54
Заключение 69
Список литературы 70
Приложение А. Техническое задание 71
Приложение Б. Листинг программы 73
Приложение В. Акт внедрения 78
Приложение Г. Оттиски статей 79

Введение

Диссертационная работа исследует мобильные устройства с программной и аппаратной архитектурой части. Акцент делается на исследование методов и алгоритмов разработки приложений для мобильных устройств различных аппаратных и программных систем. Миллионы приложений доступно в APPSTORE устройств Apple, а именно iPhone и iPad. То же самое в случае с Android и Windows Store. Следовательно, развитие мобильных приложений стало одним из самых востребованных навыков для IT специалистов.

Для разработки любого приложения, необходимо иметь знание базовой архитектуры следовательно, компоненты мобильного приложения проходят процедуры базового моделирования архитектурной модели и построения. Кроме того, любое разработанное приложение должно потреблять меньше ресурсов, чтобы производительность устройства не пострадала и была менее требовательной к аппаратному обеспечению, в том числе потребление оперативной памяти(RAM), чтение и запись с дисковых накопителей и обработку процессов ядер. Следовательно, существуют некоторые основы проектирования приложений для мобильных устройств. Любое развитие приложения будет в конечном итоге основано на системе жизненного цикла приложения (проекта), которое будет также представлено в данной работе. Также нельзя игнорировать базовое оборудование, следовательно, разные типы процессоров, памяти, датчики и устройства ввода / вывода. В последнем блоке работы будут рассмотрены инструменты для разработки приложений, которые являются платформой.

Работа состоит из 5 глав и организована следующим образом:

Глава 1 описывает основы мобильных приложений, рынок и тенденции их развития представляет различные типы и компоненты мобильных приложений, а также основные принципы дизайна мобильных приложений.

Глава 2 рассматривает компоненты, методы и основы дизайна мобильного приложения

Глава 3 охватывает стандарты безопасности при разработке мобильных приложений.

Глава 4 собирает бизнес требования и исследует систему оценки персонала

Глава 5 описывает архитектуру базы данных для системы оценки персонала и создание WEB API

Разработка приложений не является чем-то новым в мире высоких технологий, и приложения сначала разрабатывались для ПК, затем начали использоваться для различных конфигураций, таких как клиент / сервер, и массово для Web. Более поздней ступеней развития явлется последнее явление - разработка приложений для мобильных устройств. Миллионы приложений уже разработаны и развернуты для различных мобильных платформ.

Прежде чем переходить к приложениям для мобильных устройств данная работа знакомит с приложениями для ПК и Интернета. Перед началом разработки приложений для мобильных устройств необходимо изучить множество тем. Следовательно, блок охватывает различные компоненты мобильного приложения, которое включает его архитектуру. Архитектура охватывает компоненты клиентского приложения Mobile, а также инфраструктуру поддержки Mobile, затем приступает к разработке мобильных приложений.

Объект исследования магистерской диссертации: методы и средства разработки мобильных приложений

Цель диссертационной работы: исследование различных методов и инструментов разработки мобильных приложений. Использование метода WEB API, использование базы Oracle Database 11g Release 1 (11.1.0.6) для создания табличной функции с использованием PL SQL Developer

Задачи магистерской диссертации:

1. Исследовать основы разработки мобильных приложений, рынка и тенденции развития
2. Провести анализ различных методов разработки мобильных приложений
3. Собрать бизнес требования и изучить предметную часть системы оценки персонала
4. Разработать концепцию и создать архитектуру строения базы системы оценки персонала, создать API для мобильного приложения, используя метод WEB API

Актуальность магистерской диссертации: В связи с тем, что кадровые ресурсы компании становятся более значимыми, создание системы оценки является эффективным и качественным источником мотивации для сотрудников системы который смог бы регулировать задачи, а также оценивать данные задачи прямым руководителем тема магистерской диссертации является актуальным.

1 Анализ мобильных приложений и тенденции развития

1.1 Основы мобильных приложений

Данная глава ставит следующие цели:

- Понимание ключевых концепции разработки мобильных приложений
- Исследование возможностей и проблем мобильных приложений
- Подробности о приложениях для ПК и о веб-приложениях,
- Изучение контента и ключевых протоколов мобильных приложений.
- Эволюцию мобильных приложений.
- Сравнение мобильных приложений и веб-приложений с предстоящими тенденциями в области мобильности.

Поскольку мы стали свидетелями революции в потребительском пространстве в направлении мобильности, большинство аналитиков определили, что мобильные устройства являются основными каналами в Интернет по сравнению с настольными браузерами. Мобильное устройство заменяет все традиционные каналы доступа к информации. Чтобы соответствовать этой тенденции, предприятия также разрабатывают цифровые приложения для широкого спектра мобильных устройств и платформ.

Разработка мобильных приложений включает в себя процесс разработки приложений для мобильных устройств, таких как Personal Digital Assistants (PDA), планшетов и смартфонов и других мобильных устройств. Нативные мобильные приложения предназначены для работы на определенной мобильной платформе, иногда на конкретной мобильной операционной системе и поддерживаемом оборудовании.

Мобильные приложения являются частью цифровой стратегии для бизнес-потребителей (B2C). Большинство предприятий в настоящее время применяют стратегию «сначала на мобильных», в которой цифровые приложения разрабатываются, и тестируются для мобильных устройств; мобильные пользователи достигают основной цели в цифровой стратегии. Нарушение мобильности оказывает существенное влияние на доходы предприятий. Мобильные приложения формируют пользовательский опыт, предоставляют информацию в режиме реального времени и предлагают пользователям более удобный способ для получения информации или услуги.

Цифровая стратегия, основанная на мобильности, учитывает различные аспекты, такие как взаимодействие с пользователем, производительность, интерактивность, форм-факторы устройства, ограничения устройства, потребности в местоположении и персонализация.

Ниже перечислены ключевые двигатели мобильных приложений:

- Инновации в мобильном пространстве, такие как распространение смартфонов, более высокая пропускная способность, предлагаемая технологиями 3G (третьего поколения) и 4G (четвертого поколения), а в настоящее время и 5G в сочетании с технологиями хранения данных с большей емкостью с более быстродействующими микросхемами.

- Поведение потребителей: пользователи привыкли к мобильным устройствам, и им легко получить доступ к информации на ходу.
- Персонализированная доставка контента: Предприятие может использовать местоположение и датчики, чтобы предлагать более контекстуализированный, релевантный и персонализированный контент, предложения и рекламу.
- Мобильная экосистема. Большой рост магазинов мобильных приложений, таких как Apple store, Google Play store, Windows marketplace, сопровождался доступностью игр, утилит и других приложений.
- Социальные сети. С популярностью технологий Web 2.0 и социальных сетей, таких как Facebook, пользователи Twitter все чаще используют функции определения местоположения в платформах социальных сетей.
- Влияние мобильных приложений на различные домены Мобильные приложения влияют на различные отраслевые вертикали и функциональные домены. Ниже приведены изменения высокого уровня, которые могут быть внесены мобильными приложениями в различных отраслях:
- Отрасль: розничная торговля и потребительские товары (CPG). Мобильные приложения обеспечивают поиск магазинов на основе местоположения, целевые рекламные акции предложения купоны, напоминания об услугах, мобильные ставки, инструменты в магазине, инструменты перекрестной продажи перепродажи и инструменты сравнения. По сути, мобильные приложения играют ключевую роль в стимулировании трафика, повышении продаж и повышении лояльности к бренду. В сфере B2B мобильные приложения имеют новое определение управления потенциальными возможностями, функции CRM, эффективное отслеживание, автоматизацию работы на местах и тому подобное. Мобильные приложения также привели к улучшению управления товарами магазина, цепочки поставок и управления запасами.
- Банковская индустрия. Мобильные приложения предоставляют удобные способы проведения транзакций, таких как баланс счета, оплата, локализованные оповещения, тап-топ, поиск филиала и платежные купоны. Мобильные приложения также обеспечивают мобильный банкинг, мобильный кошелек и предоставляют функции «на ходу».
- Логистика. С помощью мобильных приложений проще отслеживать поставки, получать обновления, управлять складом и парком машин.
- Здравоохранение. Мобильные приложения могут легко подключать пациентов, врачей и поставщиков страховых услуг, а также предоставлять решения для управления здоровьем.
- Помимо вышеупомянутых корпоративных сценариев, мобильные приложения произвели революцию в потребительском пространстве с различными мобильными приложениями, связанными с играми, утилитами, социальными сетями, потоковым видео и многими другими.

Атрибуты мобильных приложений Ниже перечислены ключевые атрибуты мобильных приложений:

- Повсеместность. Мобильные приложения всегда доступны и подключены и позволяют пользователям получать доступ к информации в любое время и в любом месте.

- Удобство для пользователя. Мобильные приложения предоставляют отзывчивый и интерактивный пользовательский интерфейс с необходимой информацией. Камера, датчики, медиа-выход, сенсорный / мультитач / голосовой интерфейс все вышеперечисленные функции используются для предоставления упрощенной полезной информации.

- Информация о местоположении. Мобильные приложения предоставляют информацию о местоположении, используя глобальную систему позиционирования (GPS) и другие датчики.

- Минималистичный: контент и функции в мобильных приложениях минимальны, что важно для функциональности.

Рассмотрения и вызовы для мобильного приложения

Основные аспекты для мобильных приложений приведены на рисунке 1.1:

Полезность мобильного приложения	Типы приложений	Принципы	Пользователи
Продуктивность	Гибридное	Пользовательский опыт	Бизнес-пользователи
Доход	Нативное	Безопасноть	Партнеры
Преобразование	Web	Управление	Сотрудники
Лояльность		Хостинг	Клиенты

Рисунок 1.1 - Особенности мобильного приложения

Основные концепции относительно дизайна мобильного приложения перечислены ниже:

- Взаимодействие с потребителем
- Производительность за счет эффективных потоков
- Получение дополнительных доходов за счет привязанности пользователей
- Привлечение клиентов
- Лояльность пользователей полученная с помощью целевых и персонализированных предложений

Архитектура приложения

- Требования на основе Native или Hybrid в зависимости с веб-интерфейсом
- Требование промежуточного программного обеспечения для централизованной конфигурации
- Автономный или онлайновый режим хранения данных

- Принципы разработки приложений
- Опыт пользователя благодаря более богатым элементам управления и интерактивным компонентам
- Задача совместимости на различных устройствах и платформах
- Безопасность данных
- Инструменты повышения производительности
- Целевые пользователи
- Потребители для приложений B2C
- Приложения для бизнеса для бизнеса (B2B)
- Партнеры для приложений B2B
- Приложения для сотрудников для бизнеса (B2E)
- Тестирование
- Тестирование устройства
- Тестирование производительности
- Различные сценарии тестирования
- Основные проблемы в стратегии мобильных приложений приведены ниже:
- Разнообразие устройств и разнородных технологий. Существуют различные мобильные платформы и устройства. Приложение должно обеспечивать оптимальный опыт работы включающий все сценарии.
- Безопасность: мобильное приложение должно обеспечивать безопасность данных во время передачи и хранения.
- Пользовательский опыт. Мобильное приложение должно обеспечивать оптимальный пользовательский опыт, используя возможности устройства для обеспечения максимально возможного взаимодействия.
- Сеть: мобильное приложение должно быть разработано для работы в регионах с проблемами сети, задержек и слабой пропускной способностью.
- Соответствие различным стандартам ISO, ОС, мобильным платформам и устройствам.

1.2 Развитие приложений

ПК-приложения

Операционная система (ОС) управляет всеми аппаратными и программными ресурсами компьютера. Это работает как средство связи между различными компонентами компьютера.

Ниже приведены некоторые из основных функций операционной системы:

Управление периферийными устройствами

ОС управляет связью между различными устройствами с помощью своего ассоциативных драйверов. Отслеживает каждое подключенное устройство в системе. Устройство, которое отслеживает устройства, называется контроллером ввода-вывода

Управление файлами данных

Для плавной навигации и использования файлов данных; файловая система обычно организована в каталоги папки, которые будут состоять из файлов и подкаталогов. Операционная система ведет журнал информации по каждому файлу и папке, такие как местоположение файла, пользователи файла, дата создания, доступ права и т. д. Эта информация позволит ОС распределять освобождать ресурсы эффективно.

Управление памятью. Управление памятью включает в себя управление первичной памятью, которая является коллекцией данных или байтов, где каждый элемент данных или байт имеет свою собственную память адрес или местоположение. Первичная память обеспечивает быстрый репозиторий, который может иметь прямой доступ к центральному процессору (ЦП).

Приложения, основанные для персонального компьютера (ПК) - это программы, разработанные для работы на конкретных операционных системах и аппаратных платформах. Это были первые приложения, которые использовались в начальных этапах разработки программного обеспечения. Существует в основном два типа приложений для ПК, а именно автономные приложения для ПК и клиент-серверные приложения.

- Автономные приложения для ПК Автономные приложения для ПК - это независимые программы, которые будут работать в определеной операционной системе. Эти приложения обычно не используют сетевые ресурсы и не поддерживают многопользовательский режим. Служебные программы, такие как текстовый процессор, калькуляторы, и медиаплееры попадают в эту категорию. Каждое из приложений имело хороший пользовательский интерфейс для взаимодействия с пользователем ПК.

- Клиент-серверные приложения. В клиент-серверных приложениях на каждом терминальном ПК было установлено клиентское программное обеспечение, которое подключено к централизованному серверному программному обеспечению. Клиентская программа получает данные от конечного пользователя и передает подробности серверному программному обеспечению через специальный сеанс, установленный через сеть. Эти приложения также назывались «Thick clients». Программное обеспечение базы данных, сетевые игры, банковское программное обеспечение, сетевая файловая система - вот некоторые примеры приложений этой категории.

Веб-приложения. Интернет-приложения, которые в основном отображаются в настольных браузерах(таких как Google Chrome, Mozilla, Opera), классифицируются как веб-приложения. Большинство современных веб-приложений используют многоуровневую архитектуру Model-View-Controller (MVC), которая поддерживает слабую связь и гибкие модульные компоненты.

Обыное веб-приложение на основе MVC изображено на рисунке 1.2.

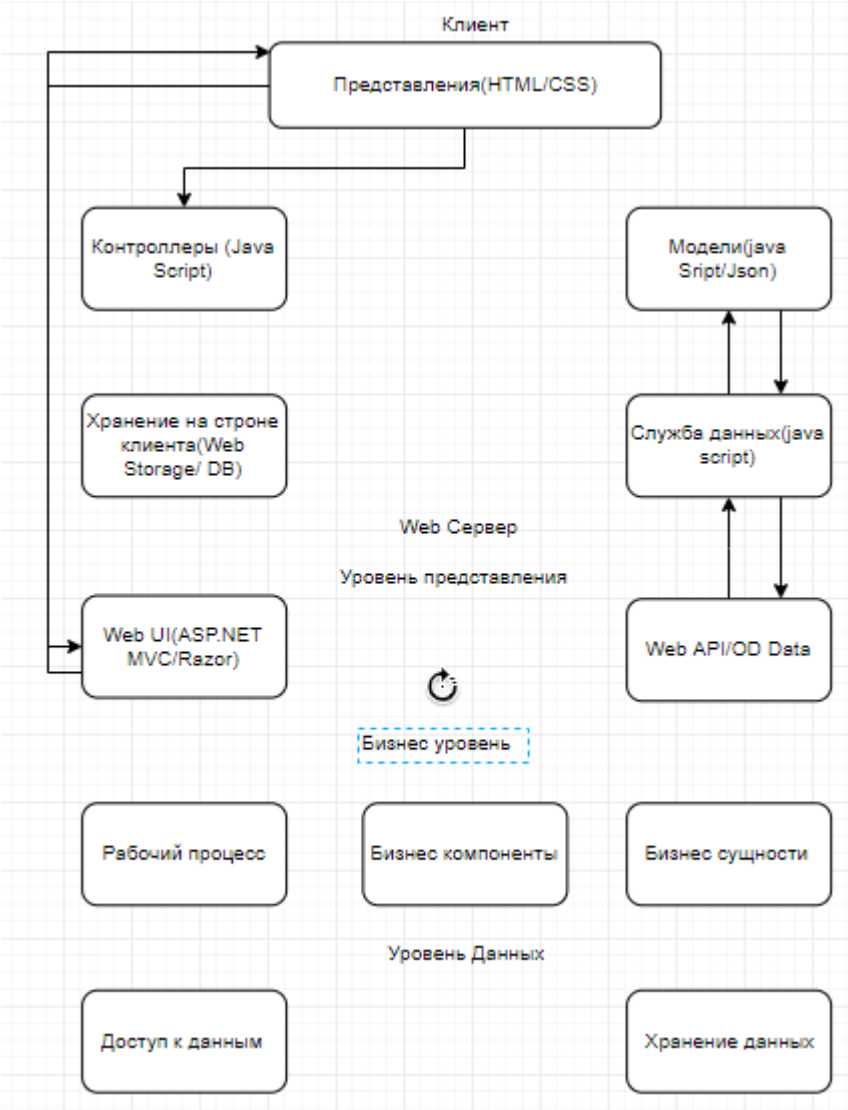

Рисунок 1.2 - Веб-приложение MVC

Ключевые уровни инфраструктуры MVC показаны ниже:

- Уровень представления (уровень представления): на этом уровне присутствуют такие компоненты пользовательского интерфейса, как портлеты, виджеты, страницы, модули пользовательского интерфейса (UI), кнопки и формы. Современные веб-приложения обычно используют компоненты JavaScript для создания модулей пользовательского интерфейса. Модули пользовательского интерфейса будут в основном отображать часть представления приложения. Просмотр компонентов взаимодействует с серверной частью через сервисы. В современных веб-приложениях используются облегченные сервисы, основанные на репрезентативной передаче состояния (REST).

- Бизнес-уровень. Этот уровень состоит из бизнес-компонентов, которые реализуют бизнес-логику и бизнес-правила. Уровень в основном состоит из механизма правил, поиска, бизнес-объектов, рабочих процессов, управления бизнес-процессами (BPM), структур кэширования и других объектов. Все бизнес-модули предоставляют услуги на уровне представления.

- Уровень данных (уровень модели). Этот уровень в основном состоит из компонентов обработки постоянства, таких как компоненты доступа к базе данных, объекты доступа к данным (DAO), компоненты запросов, платформы Object Rational Mapping (ORM) и другие. Помимо этих уровней также имеются компоненты безопасности (отвечающие за аутентификацию и авторизацию) на уровне безопасности и компоненты утилит для решения сквозных задач. Сложное n-уровневое корпоративное веб-приложение изображено на рисунке 1.3.

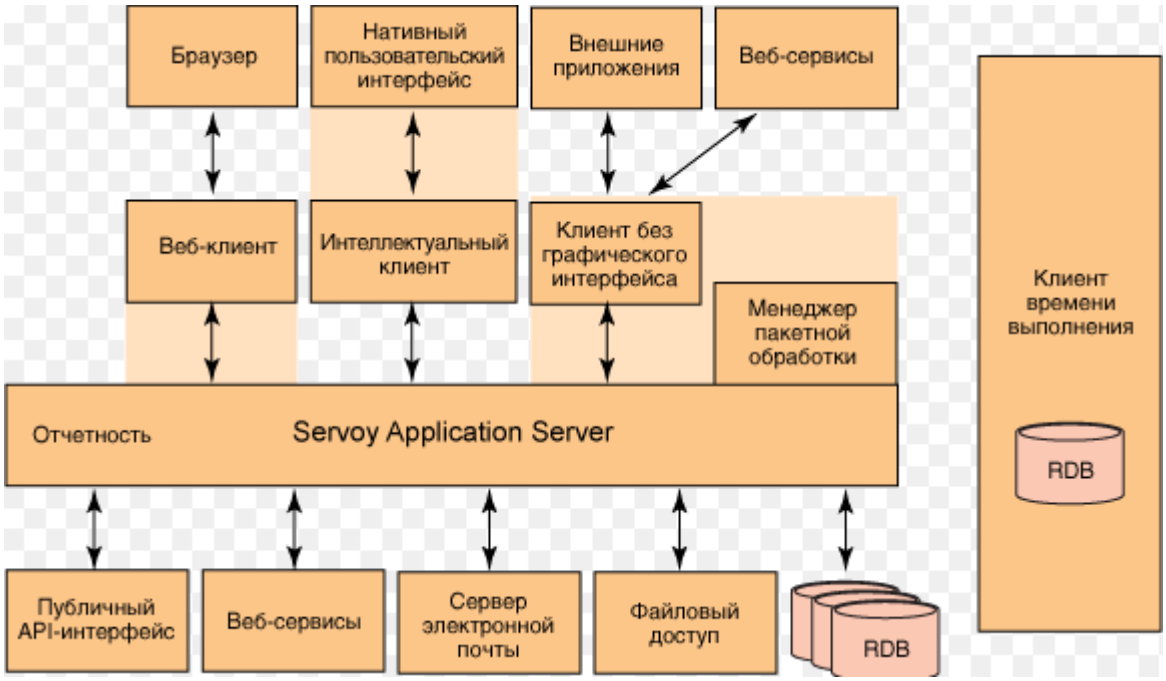

Рисунок 1.3 – N-уровневое корпоративное веб-приложение

Многоуровневая архитектура в основном основана на архитектуре MVC:

Уровень представления. В данный уровень входят компоненты представления, такие как портлеты и другие компоненты взаимодействия с пользователем. Основные компоненты описаны ниже:

- Персонализация: доступ на основе ролей и другой детальный контроль доступа для обеспечения персонализированного взаимодействия с пользователем.

- Виджеты. Везде, где требуется, будут разрабатываться клиентские виджеты на основе jQuery для обеспечения функциональности на стороне клиента. Это поможет улучшить общее восприятие пользователя и повысить производительность страницы. Отображение / обновление данных отчета в режиме реального времени, разбиение на страницы, функции поиска являются типичными сценариями, в которых может использоваться эта функция на основе AJAX.

- Поддержка нескольких устройств. Адаптивный дизайн и функции распознавания устройств будут использоваться для обслуживания различных мобильных устройств.

- Макеты страниц: Гибкий макет страницы для обслуживания различных веб-страниц.

- Информационная архитектура и модели навигации. Состоит из контекстных меню, хлебной крошки, левой навигации, карты сайта, иерархии сайта для веб-сайта.

- Уровень управления веб-контентом

- Уровень управления веб-контентом состоит в основном из следующих модулей:

- Создание контента с использованием шаблонов авторинга.
- Теги контента с соответствующими метаданными и тегами.
- Публикация контента для различных целей и в различных форматах.
- Управление активами различных цифровых активов, документов и мультимедийных файлов.
- Рабочий процесс для управления процессами утверждения, публикации и обновления контента.

Уровень электронной коммерции. Модули электронной коммерции обычно состоят из модулей, связанных с управлением каталогами, управлением заказами, модулями для совершения покупок, продвижения, перекрестных и дополнительных продаж и управления информацией о продуктах.

Уровень интеграции / уровень услуг. В решениях используется уровень бизнес-услуг для интеграции с внешней системой. Стратегия интеграции основана на архитектуре служебной шины, в которой промежуточное программное обеспечение может действовать как служебная шина, связывающая несколько приложений, которым требуются услуги друг друга через центральный уровень обслуживания. Сервисная шина становится точкой обмена данными и управляет связью с каждым периферийным приложением независимо. Все сервисные вызовы будут выполняться через уровень ESB. Будет разработан компонент JMS для отправки и получения сообщений в пункт назначения сообщений ESB. Прикладные системные службы будут разработаны и представлены на уровне ESB.

Уровень безопасности. Уровень безопасности состоит из модулей, связанных с аутентификацией, авторизацией и единым входом.

Эволюция мобильных приложений. Краткий обзор различных этапов разработки мобильных приложений изображен на рисунке 1. 4.

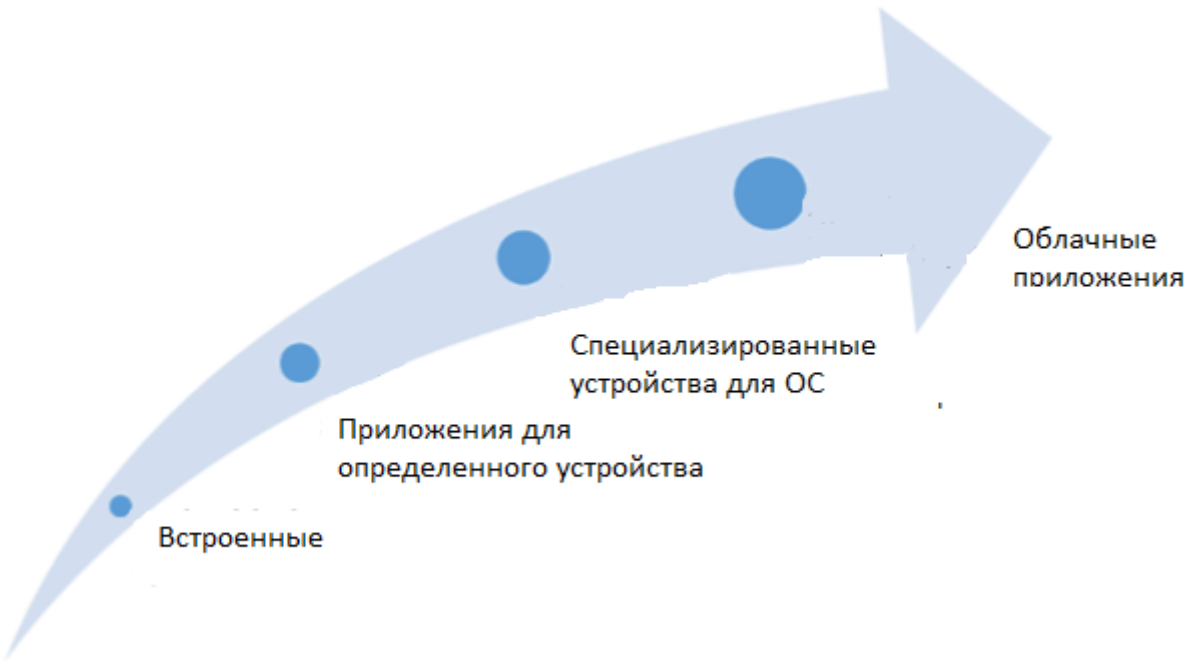

Рисунок 1.4 – Эволюция мобильного приложения

Основные характеристики различных этапов эволюции мобильных приложений подробно описаны ниже:

- Встроенные системы: в этой категории у нас есть встроенные системы, такие как калькуляторы, на различных устройствах. Встроенные системы имели ограниченную функциональность.
- Приложения, специфичные для устройства. Приложения, специфичные для данного устройства, входят в эту категорию. Например, специальные игры для устройств и другие вспомогательные приложения попадают в эту категорию.
- Приложения, специфичные для ОС: в эту категорию попадают приложения, специфичные для операционной системы, такие как игры, медиаплееры.
- Облачные приложения. В этой категории у нас всегда есть облачные приложения.

1.3 Сравнение мобильного приложения с веб-приложением

Сравнение высокого уровня мобильного приложения с веб-приложением приведено в следующей таблице 1.1:

Таблица 1.1 – Мобильное приложение против веб-приложения

Критерий	Web-Приложение	Мобильное приложение
Пользовательский опыт	Обеспечивает хороший опыт, оптимизированный для настольных браузеров.	Мобильные приложения могут использовать все возможности устройства и предлагать пользователям богатый опыт. Мобильные приложения обеспечивают богатый опыт брендинга для пользователей.
Производительность	Веб-приложения обеспечивают хорошую производительность на основе оптимизации производительности.	Собственные мобильные приложения обеспечивают высокую производительность.
Определение геолокации	Веб-приложения предоставляют относительно немного информации о местоположении.	Мобильные приложения предоставляют информацию о местоположении и контекстную информацию.
Стоимость разработки	Можно иметь единую кодовую базу и, следовательно, относительно меньшую стоимость разработки и обслуживания.	Потенциально необходимо иметь несколько баз кода для обслуживания различных мобильных платформ и, следовательно, относительно более высокую стоимость обслуживания.
Механизм доступа	К веб-приложениям в основном обращаются браузеры настольных компьютеров и мобильные браузеры.	Мобильные приложения доступны через мобильные браузеры и мобильные устройства
Интуитивно понятные функции пользователя	Веб-приложения используют ограниченное количество функций устройства.	Мобильные приложения предоставляют интуитивно понятные функции с использованием камеры устройства, датчиков, GPS и т. д. И предоставляют уведомления пользователям.

Продолжение таблицы 1.1

Интерактивность	Веб-приложения предоставляют интерактивные интерфейсы через виджеты.	Мобильные приложения обеспечивают высокий уровень интерактивности благодаря сенсорному интерфейс
Применимость	Веб-приложения обычно используются в качестве платформ отображения информации.	Игры, приложения, специфичные для местоположения, такие как приложения проката автомобилей, приложения для поиска магазинов, приложения для отчетов.
Персонализация	Веб-приложения персонализации предлагают функции персонализации через настройки на стороне сервера	Мобильные приложения обеспечивают высокую степень персонализации с помощью различных контекстных параметров, таких как местоположение, история и т.д.
Общие случаи использования	Новости, блоги.	Игры, социальные сети и услуги, связанные с местоположением

Контент для мобильных приложений разработан с учетом адаптивности. Адаптивный контент использует правила для адаптации в зависимости от контекста. Контекст включает в себя различные параметры, такие как устройство, местоположение, время, форм-фактор, мобильная платформа, разрешение экрана, параметры персонализации, локализации предпочтения и тому подобное. Адаптивный контент, следовательно, обеспечивает превосходный опыт доставки информации для мобильных пользователей.

Для обычной страницы продукта будет подробное описание продукта и краткое описание продукта. Если контент продукта разработан так, чтобы быть адаптивным, подробное описание будет отображаться в браузерах настольных компьютеров и таблицах с более высоким объемом страниц, и только краткое описание будет доставляться на смартфон с меньшим размером экрана. Аналогично, фильтрация контента на основе ролей, фильтрация контента на основе местоположения и времени может выполняться с использованием адаптивных правил.

Адаптивный дизайн в основном обслуживает гибкие макеты страниц и экранов. Комбинация адаптивного дизайна страницы и адаптивного контента была бы идеей для мобильных сценариев. Основные протоколы, используемые в мобильных приложениях:

- Протокол беспроводных приложений (WAP): предоставляет спецификации и правила для устройств беспроводной связи, таких как смартфоны. WAP оптимизирован для мобильных устройств с низкой памятью и низкой пропускной способностью.

- Передача репрезентативного состояния (REST). Отзывчивые мобильные веб-приложения обмениваются данными с сервером с помощью упрощенных вызовов службы REST. Обычно данные JSON используются в сервисе REST для обмена данными.

- Ниже перечислены некоторые из тенденций в области мобильности:

- Адаптивные приложения на основе HTML 5: технологии HTML 5 помогут разработчикам создавать адаптивные веб-приложения, отвечающие требованиям всех мобильных устройств.

- Принесите свое собственное устройство (BYOD): сотрудники могут использовать свои мобильные устройства для работы и упрощения работы.

- Интернет вещей (IoT): предоставляет данные в реальном времени через мобильные устройства и датчики и другие подключенные устройства. Носимые устройства приобретают все большую популярность, и данные, публикуемые с носимых устройств, будут использоваться мобильными устройствами.

- Управление мобильными устройствами (MDM): управление конфигурацией, безопасностью и политиками различных устройств.

- Управление мобильными приложениями (MAM): управление версией, доставкой и предоставлением различных приложений.

- Увеличение разнообразия коммуникаций, таких как касание, голос, распознавание лиц / жестов, видео, сканирование, ближняя связь и т. Д.

- Оповещения и уведомления в режиме реального времени, основанные на местоположении, и другие инновационные функции, такие как оптимизация маршрута.

- Мобильность как услуга (MaaS). MaaS - это ускорение выхода на рынок и снижение сложности.

1.4 Мобильные операционные системы

Рассмотрим две популярные платформы для разработки мобильных приложений: iOS и Android.

Android - это платформа для разработки мобильных приложений с открытым исходным кодом, основанная на Java и поддерживаемая Google. Основными функциями, поддерживаемыми Android, являются легковесное хранилище на основе SQLite, обмен сообщениями SMS и MMS, многоязычная поддержка, мобильный браузер. Другие ключевые функции: поддержка нескольких касаний, многозадачность, голосовые функции, внешнее хранилище и т. д.

Для разработки Android требуется Android SDK, библиотеки, эмулятор и Eclipse IDE. Тестирование может быть выполнено с использованием API

тестирования Android. Можно развернуть приложения Android в магазине Google Play.

iOS - мобильная ОС для устройств Apple. Разработка для iOS происходит с использованием цели C. iOS поддерживает множество функций, таких как iMessage, iCloud, Siri и т. Д. IOS предоставляет встроенные приложения, такие как почта, уведомления, календарь контактов, закладки, синхронизация и т. д.

Разработка для iOS включает iOS SDK (XCode), iOS Simulator, XCode IDE и другие платформы для создания приложений iOS.

Для разработки мобильных веб-приложений мы могли бы использовать многочисленные JavaScript - фреймворки, такие как SenchaTouch, jQuery Mobile, jQTouch и программное обеспечение, такое как PhoneGap, которое используется для разработки мобильных веб-приложений.

Мобильная операционная система - это системное программное обеспечение, работающее на мобильных устройствах. Это программное обеспечение, которое обеспечивает базу, на которой приложения или программы могут эффективно выполняться на мобильных телефонах, смартфонах, планшетах и портативных устройствах. Помимо основных функций операционной системы, мобильная ОС также поддерживает сотовую и Wi-Fi связь.

В наши дни мобильные телефоны часто используются для проверки электронной почты, игр, новостей и т. д. Существуют сотни тысяч приложений, каждое со своей собственной целью. Вы можете скачать приложение новостей, которое сообщает вам текущие новости или обновления в вашем городе. Действительно, мы можем найти приложение для любой функции, которую мы ищем

Ниже приведены различные уровни мобильной операционной системы (рисунок 1.5):

Уровень 0 включает в себя аппаратное и стороннее оборудование.

Уровень 1 включает системные библиотеки, сторонние библиотеки и системные службы.

Уровень 2 включает в себя приложения для устройств и сторонние приложения.

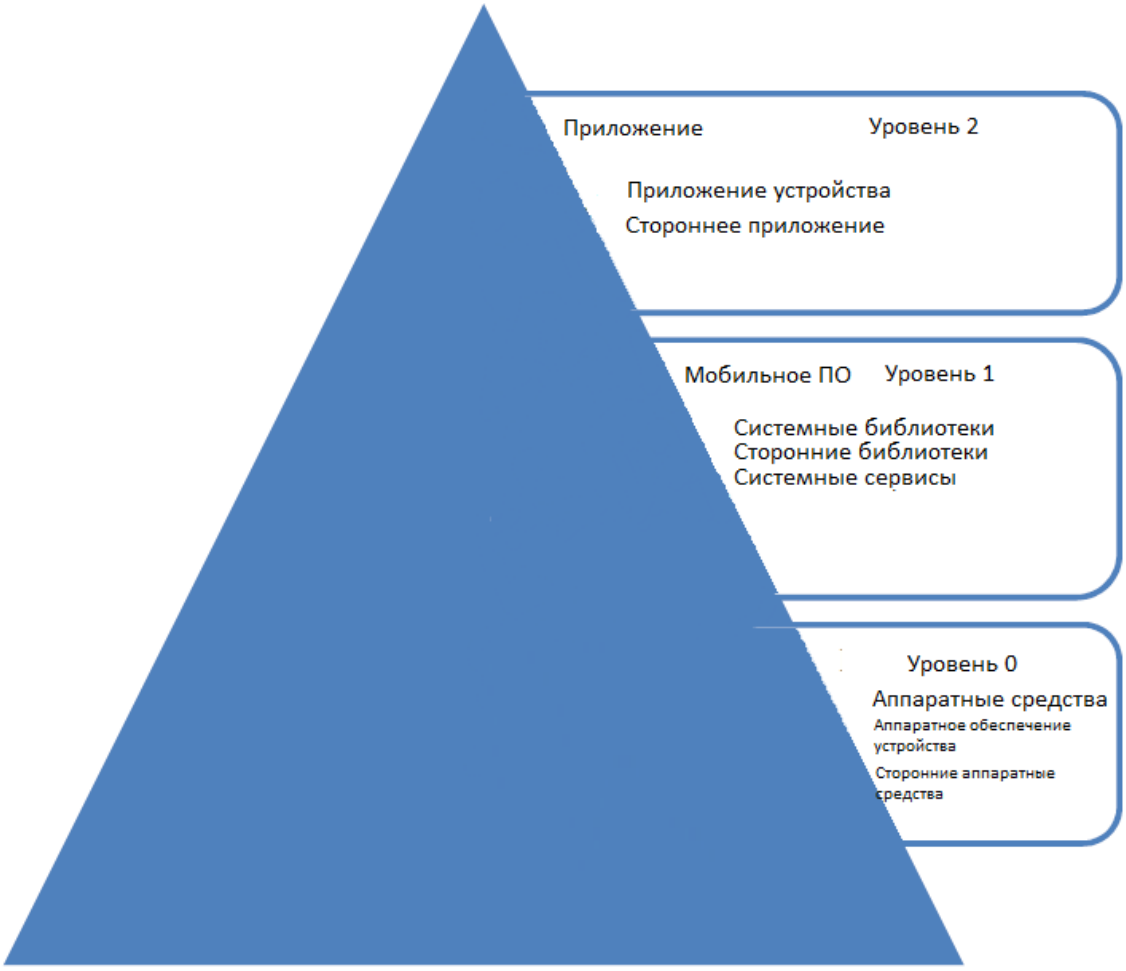

Рисунок 1.5 – Уровни мобильной ОС

Архитектура Android включает в себя следующие четыре слоя:

- Заявка
- Платформа приложений
- Библиотеки или Android Runtime
- Ядро Linux

Мобильные операционные системы тесно связаны, интерфейсом и функциями мобильного телефона, имя и версию операционной системы программного обеспечения можно найти только в соответствующих настройках телефона. Есть несколько примеров мобильных операционных систем, которые включает в себя Apple iOS, Google Android и ОС Windows Phone от Microsoft.

Большинство мобильных операционных систем связаны с определенным оборудованием. Например, IOS от Apple позволяет пользователям устанавливать приложения только из магазина приложений Apple. В случае с Android, это Play Store. Вот список наиболее часто используемых мобильных операционных систем:

- Apple iOS
- Google Android
- ОС Windows Phone от Microsoft

- Nokia Symbian
- BlackBerry OS (исследования в движении)

Ниже приведены некоторые из существующих мобильных операционных систем. Таблица 1.2 описывает их следующим образом:

- iOS
- Android
- Windows 10 Mobile
- Sailfish OS
- Тизен
- Ubuntu Touch

Таблица 1.2 – Существующие мобильные операционные системы

iOS	Мобильная операционная система iOS была изначально представлена Apple Inc для ее развертывания в устройствах iPhone. Теперь данная ОС совместима с различными устройствами Apple, такими как iPhone, iPad, iPod Touch и т. д. Вы можете видеть мобильное устройство iOS операционная система только на устройствах Apple собственного производства, потому что appapple не выдает лицензию на ее развертывание на стороннее оборудование. Операционная система iOSmobile расширенное использование Mac OS X операционной системы Apple Inc.
Android	Android - самая известная мобильная операционная система. Система, которая была введена Google под категорией открытого и свободного хранилища программного обеспечения Google. Репозиторий свободного программного обеспечения состоит из операционной системы, промежуточного программного обеспечения. Разработка версии для Android вдохновленна словом «десерт», каждая версия приходит в алфавитном порядке с дальнейшими обновлениями и усовершенствования. Существующие несколько версий Android - Cupcake - Donut - Eclair - Gingerbread - Honeycomb - Ice Cream Sandwich - Jelly Bean - KitKat - Lollipop - Marshmallow - Nougat

Продолжение таблицы 1.2

Windows 10 Mobile	Windows 10 Mobile является новейшей версией Windows операционная система, где Microsoft пытается унифицировать их настольный компьютер, планшет и мобильная операционная система в одна ОС. Windows 10 Mobile должна была нести те же функции, что и настольная версия. В ноябре 2015 года Microsoft запустила ОС Windows 10 для мобильных устройств с Lumia 950, Lumia XL и Lumia 550. Однако в данное время операционная система не занимает лидирующих позиций на рынке

Android является одной из наиболее широко используемых мобильных операционных систем, которая поддерживает различные смартфоны и планшеты. Изначально Android был разработанный компанией Android Inc., которую Google купил в 2005 году. В 2007 году Google выпустила первую бета-версию Android Software Development Kit (SDK), тогда как первая публичная версия, Android 1.0, была анонсирована в Сентябрь 2008. Исходный код Android доступен в свободном и открытом исходном коде. лицензии. Для мобильных устройств Android предоставляет единый подход к разработка приложений, что означает, что мобильные разработчики разрабатывают приложения только для Android и их приложения должны иметь возможность работать на разных устройствах работает на Android. Это отдельная программная база, которая может быть развернута легко на любом типе конфигурации оборудования.

Особенности Android. Android - это надежная операционная система, которая бросает вызов Apple 4GS и поддерживает хорошее количество функций. В таблице 1.3 перечислены некоторые функции Android

Таблица 1.3 – Функции Android

Характеристики	Описание
Интерфейс	Довольно простой и привлекательный пользовательский интерфейс
База данных	SQL Lite, облегченная СУБД (реляционная база данных
Мультитач	Мультитач доступен, можно коснуться двух или более объектов
Многозадачность	Два или более приложений могут быть запущены одновременно

Приложения, которые обычно называются («APPS»), расширяют функциональность устройства и написаны с использованием комплекта

разработки программного обеспечения Android (SDK) и, часто, язык программирования Java. SDK в комплекте с набором инструментов разработки, таких как отладчик, библиотеки программирования, эмулятор телефонного аппарата, документация, учебные упражнения и тестовый код. Android превратился в развивающееся определение сторонних приложений, которые могут приобрести, загрузив и установив APK (приложение для Android) пакет), либо загрузив их с помощью магазина приложений который помогает пользователям устанавливать, обновлять и удалять приложения из своих гаджеты. Google Play Store - самый известный магазин приложений представлены на телефонах Android, которые отвечает условиям Google и позволяет пользователям просматривать, загружать и обновлять приложения, которые сделаны доступны для Google и других сторонних разработчиков приложений. Открытая природа Android, позволяет различные сторонние магазины приложений, также доступные для Android, либо дать альтернативу гаджетам, которые предварительно не сложены Google Play Store и приложения, которые нельзя предлагать в Google Play Store из-за нарушения политики или по другим причинам.

Основы iOS. iOS - мобильная операционная система, разработанная Apple. Ранее была известна как iPhone OS. Эта операционная система работает на различных устройствах, таких как iPhone,iPad, iPod Touch и т. д., и он доступена на 40 языках. Это вторая самая популярная операционная система для портативных устройств с долей 27% по всему миру. iOS изначально была разработана для телефонов в 2007 году он был расширен для поддержки других устройств, таких как iPod Touch в сентябре 2007 года и iPad в январе 2010 года. Есть множество версий iOS,. Текущая версия iOS 12, которая была выпущен 13 сентября 2018 года. Поддерживается на iPhone 10 и выше. Эта платформа написана на C, C ++, Objective-C и Swift. Это очень безопасная платформа, и она использует множество функций безопасности. Ниже приведены функции безопасности iOS:

- Безопасная загрузка
- Пароль
- Touch ID
- Неисполняемая память
- Шифрование
- Безопасность приложения
- Сетевая безопасность
- Двухфакторная аутентификация
- Биометрическая аутентификация

В данном разделе было начато обсуждение основных соображений и проблем при разработке мобильных приложений. Затем рассмотрены приложения для ПК, включая автономные и клиент-серверные приложения. Были также рассмотрены различные этапы разработки мобильных приложений. В первой главе сравнили мобильные приложения с веб-приложениями и рассмотрели контент и протоколы мобильных приложений.

2 Компоненты мобильного приложения

2.1 Архитектура мобильного приложения

Мобильное приложение состоит из различных клиентских компонентов и компонентов инфраструктуры поддержки. Важно понимать различные компоненты мобильного приложения для проектирования и разработки эффективного мобильного приложения. Знание архитектуры различных типов мобильных приложений и их применимости поможет нам выбрать наиболее подходящее мобильное приложение для данного сценария.

Архитектура мобильного приложения также дает представление о дизайне пользовательского интерфейса, аспектах интеграции и других подобных соображениях.

В данном разделе будут подробно рассмотрены различные компоненты мобильного приложения и подробно изучена архитектура различных мобильных приложений.

Данная глава позволяет рассмотреть следующие концепции и архитектуры мобильного приложения:

- Ключевые концепции архитектуры мобильных приложений,
- Детали архитектуры нативного мобильного приложения,
- Подробности архитектуры гибридного мобильного приложения,
- Детали архитектуры мобильного веб-приложения,
- Детали различных компонентов клиентской стороны и инфраструктуры поддержки мобильного приложения

Мобильные приложения можно разделить в основном на 3 категории:

- Мобильные веб-приложения на основе браузера: это веб-приложения, разработанные с использованием методов адаптивного веб-дизайна, которые могут обслуживать различные устройства и форм-факторы.

- Собственные мобильные приложения. Эти приложения созданы для определенных мобильных платформ (таких как iOS, Android), которые могут полностью использовать возможности устройства. Обычно нативные мобильные приложения создаются с использованием SDK, предоставляемых мобильными платформами.

- Гибридные приложения. Эти приложения могут быть разработаны с использованием веб-технологий, таких как JavaScript, и могут также частично использовать возможности устройства с использованием веб-уровня на собственном уровне.

Таблица 2.1 – Предоставляет ключевые отличия трех типов архитектур мобильных приложений

Показатель	Мобильный веб-интерфейс на основе браузера	Native-applications	Hybrid applications
Технология разработки	Технология разработки Обычно использует адаптивный веб-дизайн (RWD) с использованием CSS3, медиа-запросы.	Собственные приложения разрабатываются с использованием iOS, Android и других мобильных платформ	Разработаны с использованием единой технологии (обычно JavaScript) и могут быть перенесены на любую нативную технологию
Характеристики	Используется только для отзывчивого и интерактивного интерфейса. Не удается полностью использовать все функции мобильного устройства.	Обеспечивает лучшую портативность, время выхода на рынок, ремонтопригодность. Может в полной мере использовать все возможности устройства, такие как камера, датчик, контактные книги и тому подобное.	Обеспечивает лучший пользовательский опыт. Ограниченное использование собственных функций устройства. применимость
Применение	Может использоваться для отображения информации без какой-либо поддержки в автономном режиме и для размещенных решений.	Может использоваться для функций, требующих богатого пользовательского опыта, таких как игры. Может использоваться для автономных требований и для функций, требующих определенных функций устройства.	Может использоваться для облегчения переносимости между мобильными платформами и кроссплатформенной совместимости. Мы могли бы использовать это для более быстрого выхода на рынок с частыми выпусками

Продолжение таблицы 2.1

Совместимость	Работает на мобильных устройствах и платформах. Оффлайн не поддерживается.	Родные приложения работают на определенных мобильных платформах; поддержка автономных возможностей.	Гибридные приложения можно переносить на различные мобильные платформы.
Разработка	Быстрая разработка и сопровождение	Более высокая стоимость	Более низкая стоимость разработки из-за портативности. Сборка раз развертывается на любой модели платформы.
Преимущества	Соответствие стандартам, меньшая стоимость разработки.	Высокая производительность, богатый пользовательский интерфейс, могут использовать собственные мобильные функции	легкая переносимость
Производительность	Относительно низкая производительность	Оптимальная производительность	Относительно низкая производительность
Пользовательский опыт	Обеспечивает оптимальное взаимодействие с веб-интерфейсом	Обеспечивает превосходные функции пользовательского интерфейса, максимально используя возможности устройства.	Обеспечивает приличный интерфейс
Хостинг	Облако или на площадочных площадках	Магазины приложений	Магазины приложений

Архитектура нативного мобильного приложения

Приложение Native Mobile полностью использует возможности устройства. Рисунок 2.1 и рисунок 2.2 изображают архитектуры для собственных приложений для iOS и Android:

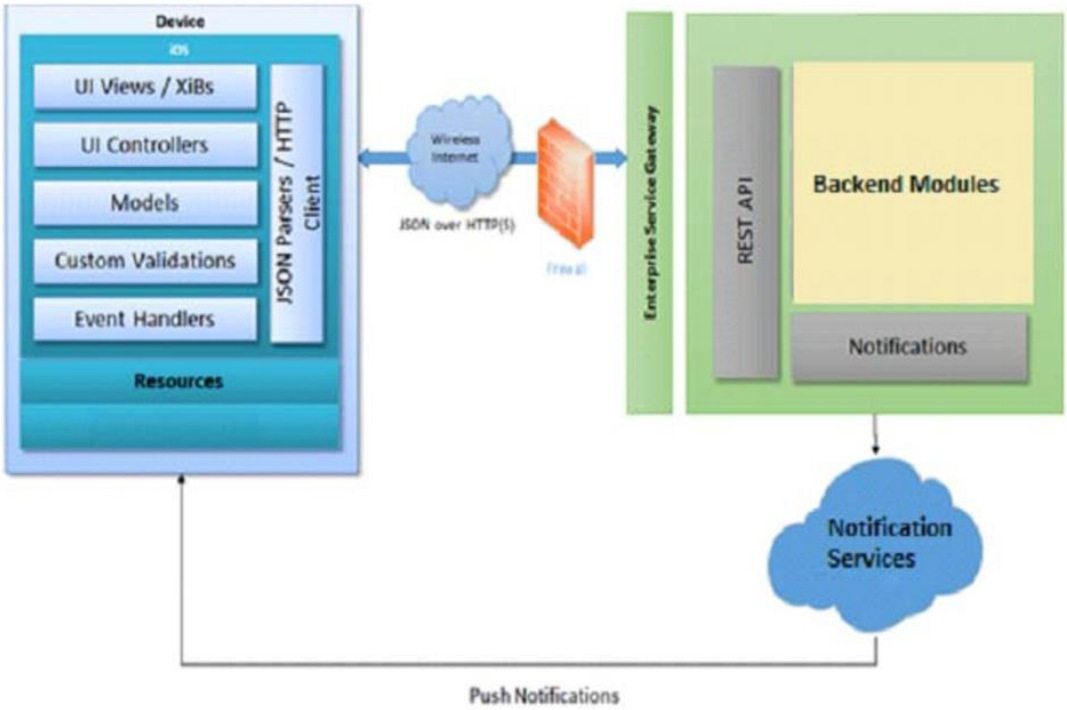

Рисунок 2.1 – Нативное мобильное приложение на базе iOS

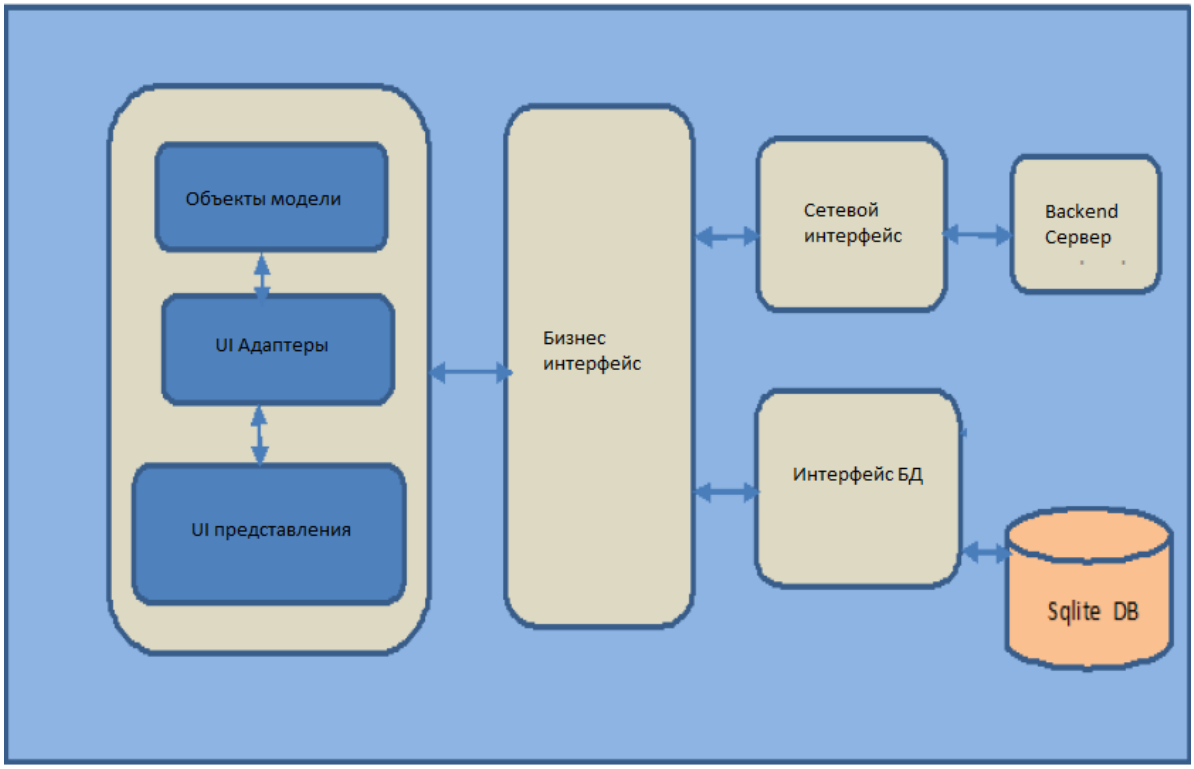

Рисунок 2.2 – Архитектура нативного мобильного приложения на базе Android

Архитектура гибридного мобильного приложения

Гибридное приложение предназначено для различных мобильных ОС с очень тонкой оболочкой приложения с веб-представлением. Требуется разовая разработка приложений с публикацией в различных магазинах мобильных приложений. Гибридные приложения очень популярны при разработке адаптивных приложений для одностраничных приложений для планшетов и смартфонов.

Ключевые компоненты гибридного приложения приведены на рисунке 2.3. Разработка мобильных приложений с использованием гибридных приложений имеет следующие преимущества:

- Удобство использования: возможность использовать специальные функции устройства для повышения удобства использования.
- Поддержка: простота обслуживания кода и простота будущих улучшений.
- Расширяемость: поддержка нескольких платформ
- Разнообразие устройств: никаких дополнительных усилий для поддержки новых устройств
- Переносимость: на различные мобильные платформы
- Более быстрое время выхода на рынок: благодаря более быстрому развертыванию в магазинах мобильных приложений.

У нас могут быть специальные мобильные сервисы, которые объединяют необходимые базовые сервисы и предоставляют оптимизированные данные для мобильного приложения. Рисунок 2.3 представляет собой расширенный вид решения для мобильности.

Он детализирует функциональные возможности и интерфейсы с базовыми компонентами, как указано ниже:

- Пользовательский интерфейс приложения: все экраны будут разработаны с использованием HTML5, CSS3 и JavaScript. Пользовательский опыт будет специально разработан для мобильных приложений с целевой совместимостью платформы.
- Бизнес-правила: операционные процедуры для принятия решений, определения рабочих процессов и оперативных решений.
- Уровень интеграции сервисов: Интеграция всех бизнес-сервисов с серверной системой будет управляться этим уровнем.
- Структура безопасности. Все данные, сохраненные на устройстве, будут надежно храниться с использованием методов шифрования, специфичных для каждой платформы. Случайно сгенерированный уникальный ключ будет использоваться для шифрования и дешифрования. Ключ, используемый для шифрования, будет надежно сохраняться в устройстве. Advanced Encryption Standard (AES) - это алгоритм, используемый в целях безопасности. Данные, передаваемые по сети, будут привязаны к бэкэнд-сервису.

- Аутентификация / Управление пользователями: управление данным пользователем и бизнес-логики будут обрабатываться этим уровнем. Аутентификация будет выполняться в режиме онлайн / офлайн.

- Локализация: это уровень, который обеспечит многоязычную поддержку приложения. Все статические литералы, специфичные для каждого языка, будут поддерживаться локально внутри приложения. Литералы, специфичные для динамического контента, будут извлечены из серверных служб.

- Уровень доступа к данным: все данные, хранящиеся локально, будут доступны через этот уровень. Он состоит из объектов данных и локального хранилища. SQLite будет использоваться в качестве локального хранилища для приложения.

- Native Bride / Wrapper - этот уровень предоставит возможность доступа к собственным функциям устройства. Это сочетание платформы мобильного ядра и JavaScript.

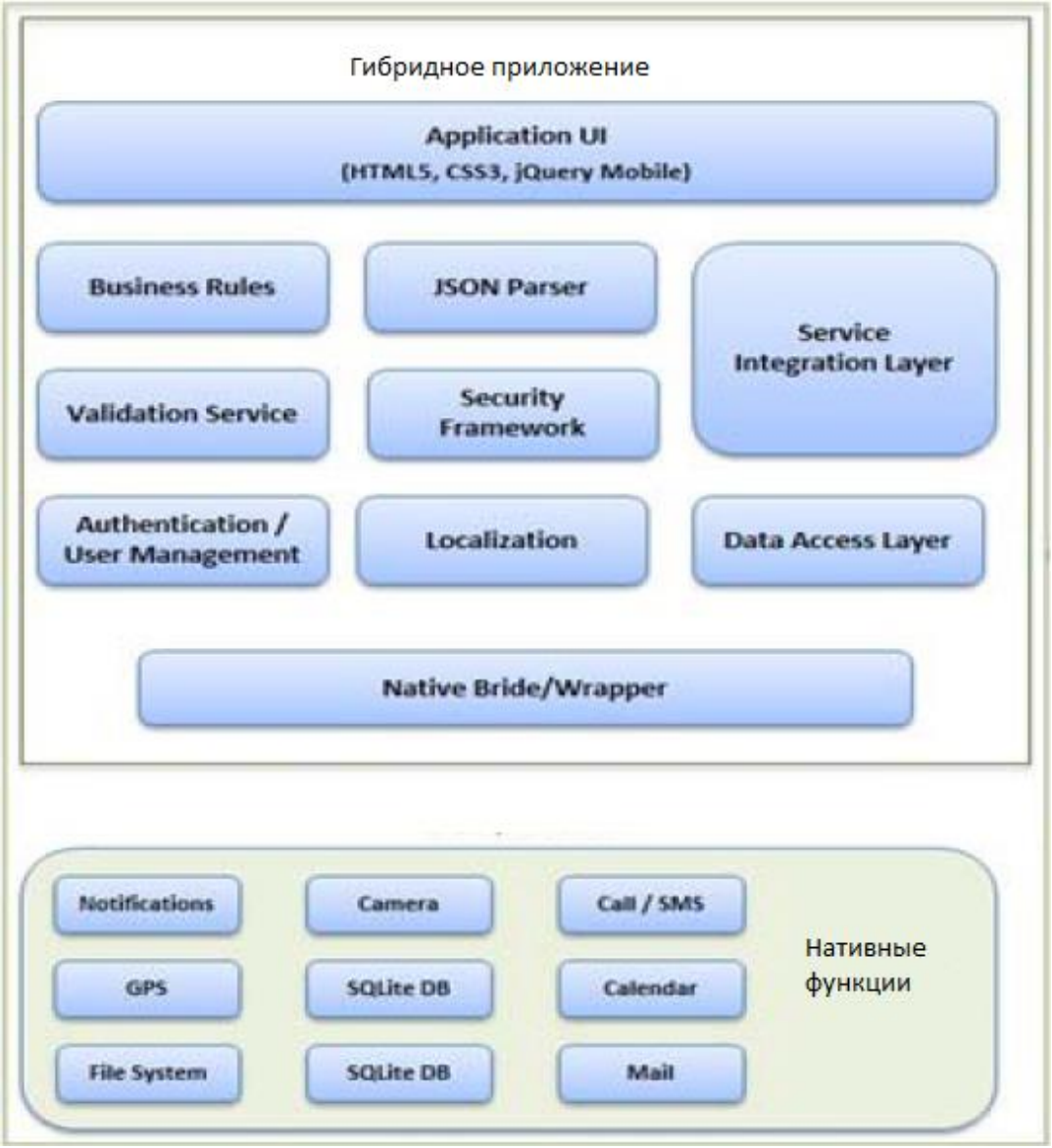

Рисунок 2.3 – Архитектура гибридного мобильного приложения

Сквозной поток гибридного мобильного приложения показан на рисунке 2.4 и пояснены соответствующие уровни.

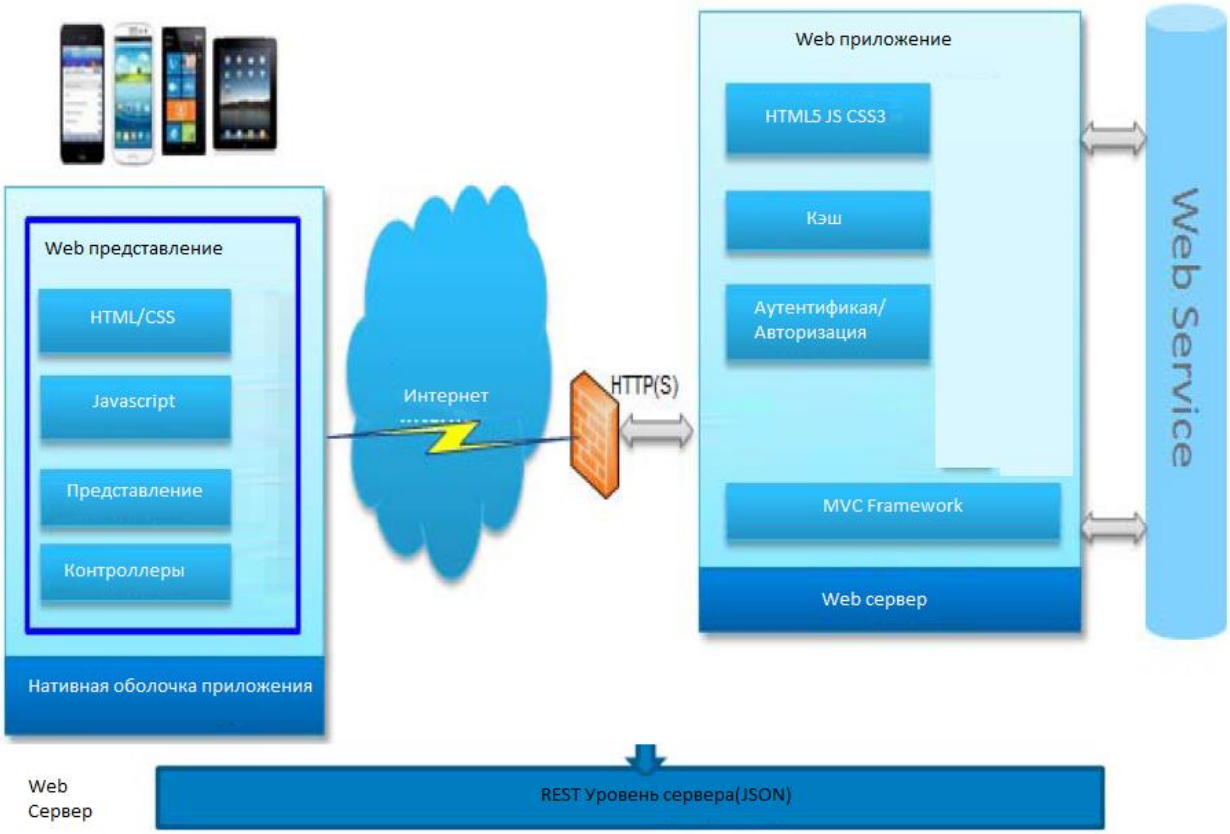

Рисунок 2.4 – Сквозной поток гибридного мобильного приложения

Уровень представления

- Пользовательский интерфейс будет соответствовать адаптивному веб-дизайну (RWD) и будет разрабатываться с использованием HTML5.0, CSS3, Twitter Bootstrap с JQuery.

- Пользовательский интерфейс может быть упакован с использованием Apache Cordova для создания гибридного приложения. Веб-сервисы REST / JSON будут задействованы.

- Гибридные приложения будут иметь два компонента: собственный и веб-компонент.

Бизнес-уровень

- Веб-службы REST / JSON будут использоваться для доступа к приложениям на стороне сервера.

- Бизнес-уровень состоит из серверных компонентов для интеграции служб и других настраиваемых бизнес-служб. Уровень безопасности

- Авторизация для пользователя приложения будет усилена службами реестра пользователей.

Архитектура мобильного веб-приложения

Мобильные веб-приложения в основном разрабатываются с использованием адаптивного дизайна. RWD предоставляет гибкие и гибкие

макеты, которые автоматически подстраиваются под различные устройства и форм-факторы, которые включают следующие методы:

Fluid Design содержит размер страницы в относительных единицах.

Гибкое изображение и мультимедиа Гибкие изображения будут автоматически масштабироваться в зависимости от разрешения экрана.

Медиа-запросы CSS3 обеспечивают гибкие макеты и используют различные правила стиля CSS, зависящие от устройства.

На рисунке 2.5 показан адаптивный макет, отображаемый на нескольких устройствах

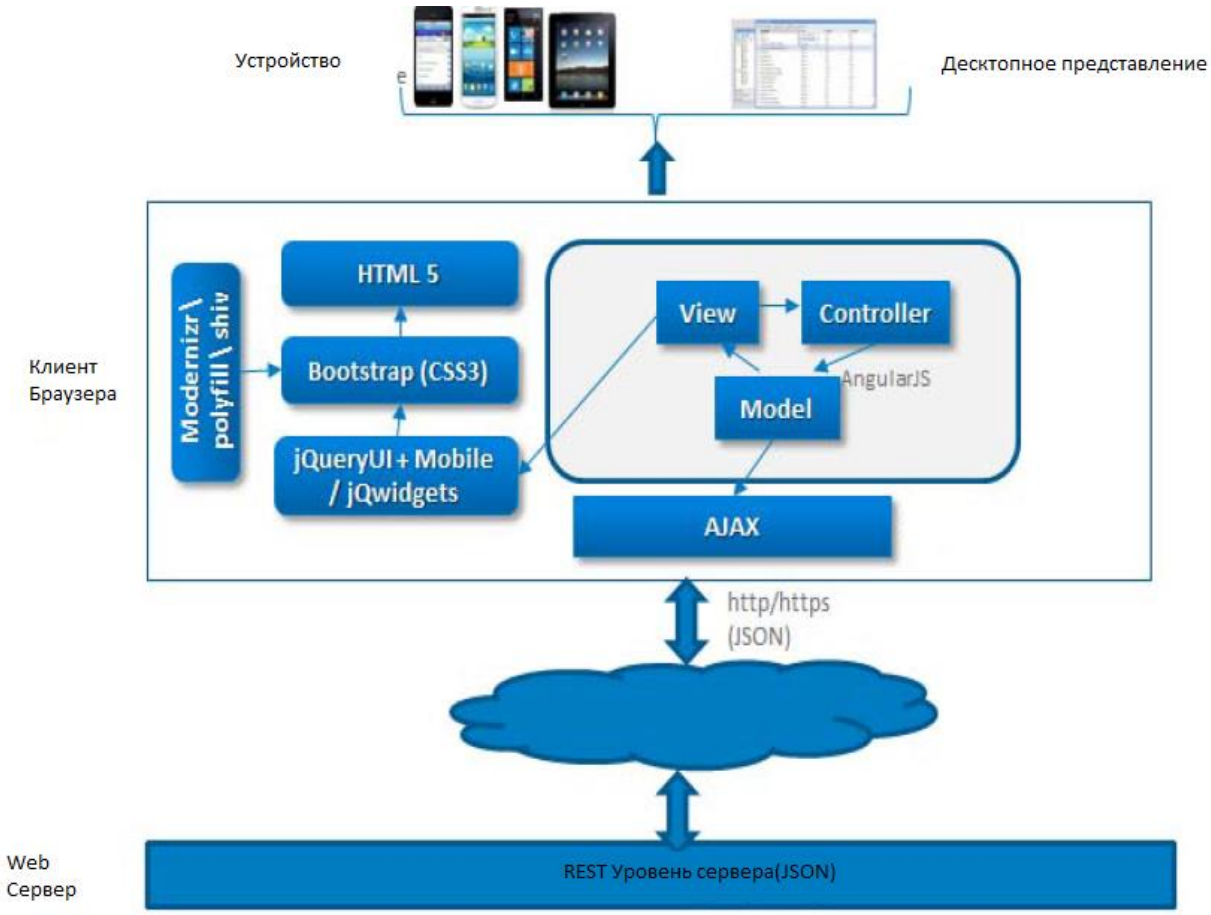

Рисунок 2.5 – Мобильное веб-приложение с использованием RWD

Существенные особенности RWD приведены ниже:

- Пользовательский опыт - отзывчивый веб-сайт является гибким и адаптирует макет экрана для обеспечения оптимального пользовательского опыта.

- Экономически эффективный - только одна версия исходного кода и только одна система управления контентом для обновления контента, что означает, что RWD сэкономит как время, так и деньги.

- Улучшенная SEO - структура URL останется неизменной на всех устройствах, что улучшит видимость и рейтинг вашей поисковой системы.

Вместо создания ссылок или оптимизации контента для нескольких веб-сайтов вам потребуется продвигать только один адаптивный веб-сайт.

- Повышение конверсии. Потенциальный клиент может с легкостью получить доступ к вашему веб-сайту с помощью предпочитаемого им устройства, что улучшит взаимодействие с пользователем, что повысит продажи и повысит коэффициент конверсии.

Ключевые архитектурные принципы, которым следовали при разработке мобильного веб-приложения, следующие:

- Технология, основанная на открытых стандартах.
- Многоуровневая архитектура с использованием шаблона MVC.
- Модульное и расширяемое проектирование компонентов.
- Принятие сервис-ориентированной архитектуры для интеграции.
- Использование технологий с открытым исходным кодом, где это возможно.
- Проектирование на основе производительности.
- Непрерывный подход к сборке и интеграции для выполнения.

Пример архитектуры мобильного веб-приложения с адаптивными компонентами в архитектуре MVC изображен на рисунке 2.6.

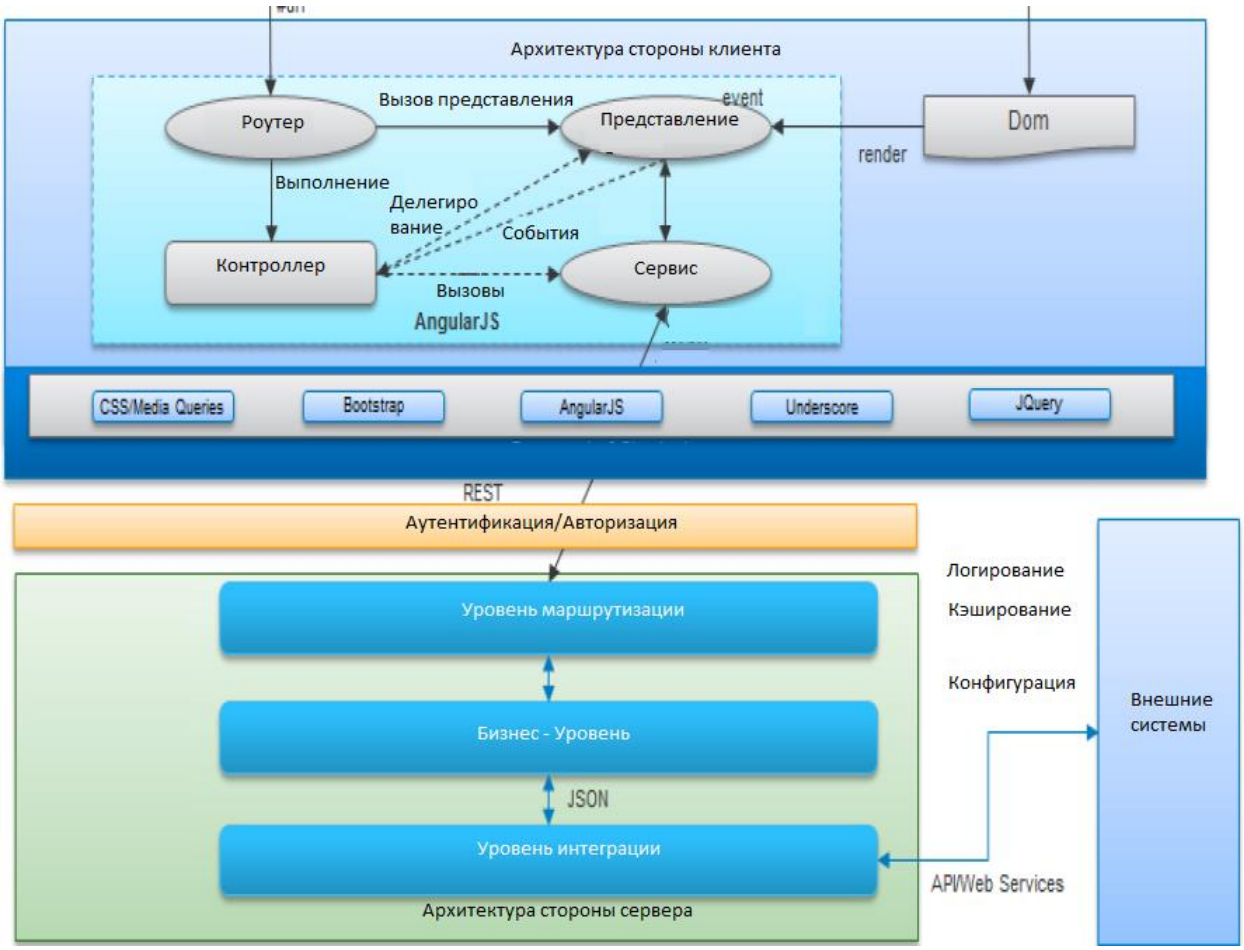

Рисунок 2.6 – MVC-архитектура мобильного веб-приложения

Архитектура MVC, изображенная выше, состоит из компонентов на стороне сервера и на стороне клиента, где клиентские компоненты будут иметь логику представления, основанную на шаблоне проектирования MVC, с использованием утвержденных библиотек JS (Java Script), тогда как уровень сервера будет иметь всю бизнес-логику, которая будет представлена через сервисы RESTful. Клиентские компоненты будут взаимодействовать с серверными компонентами, используя службы REST для извлечения и сохранения данных.

Уровень клиента MVC: Клиентская структура MVC (Angular JS) будет использоваться для структурирования приложения, что обеспечит:

- Поддержку маршрутизации в приложении для обеспечения навигации в приложении;
- Управляемое событиями взаимодействие между представлениями и моделью;
- Упрощенные вызовы CRUD (создание, чтение, обновление и удаление) в сервисах RESTful.

Уровень сервера

- Уровень маршрутизатора сопоставляет URL-адрес с конечной точкой службы. Этот слой будет интерфейсом со стороны клиента. Эти службы REST передают параметры запроса на бизнес-уровень.
- Бизнес-уровень будет проверять входные данные, выполнять любые манипуляции с данными или реализацию бизнес-логики для данных, извлеченных из уровня интеграции.
- Уровень интеграции будет в основном интегрироваться с внешней системой, которая будет предоставлять API / веб-сервисы. Передача данных будет осуществляться в формате JSON (Java Script Object Notation). Роли различных слоев изображены в таблице 2.2:

Таблица 2.2 – Слои в веб-мобильном приложении

Уровень	Клиент/Сервер	Описание
Маршрутизатор	На стороне клиента	Связывание URL с контроллерами и представлениями
Представление	На стороне клиента	Состоит из угловых директив, шаблонов, файлов CSS и JS.
Контроллер	На стороне клиента	Контроллер связывает представление с внутренними данными.
Сервисы	На стороне клиента	Сервисы взаимодействуют с внутренними данными

Продолжение таблицы 2.2

Роутер	На стороне сервера	Маршрутизатор сопоставляет URL с конечной точкой службы
Бизнес	На стороне сервера	Заботится о бизнес-правилах, проверках, преобразовании данных и таких требованиях.
Интеграция	На стороне сервера	Интегрируется с внешней системой, которая будет выставлять API / веб-сервисы

Пример технологического стека для реализации мобильного веб-приложения показан в таблице 2.3.

Таблица 2.3 – Технологический стек для мобильного веб-приложения

Технология исопльзования	Сторона использования	Описание	Преимущества
AngularJS	Клиентская сторона	а основе MVC JavaScript	Введен чистый шаблон проектирования MVC для клиентских веб-приложений. Расширяется с помощью директив.
Bootstrap	Клиентская сторона	Клиентский интерфейс на основе HTML и CSS	Делает интерфейсную разработку быстрее и проще
HTML / CSS	Клиентская сторона	Стандартные технологии для веб-разработки	Серверная часть REST API на стороне сервера, созданная с использованием Spring или Node.js.

Продолжение таблицы 2.3

Spring	Серверная сторона	Серверная часть REST API на стороне сервера, созданная с использованием Spring или Node.js	Обширная и масштабируемая среда
Службы REST	Серверная сторона	Серверные веб-службы на основе URL-адресов REST (http / https)..	Легкие, поддерживаемые и масштабируемые

2.2. Компоненты мобильного клиента

Ниже перечислены ключевые компоненты мобильного клиентского приложения:

- Компоненты пользовательского интерфейса, такие как виджеты, кнопки, экраны и компоненты навигации;
- Библиотеки JavaScript в случае мобильных веб-приложений;
- Библиотеки виджетов для управления виджетами на стороне клиента;
- Контроллеры пользовательского интерфейса для клиентских платформ MVC.
- Валидаторы и обработчики событий для обработки прерываний и уведомлений.
- Компоненты модели для обработки данных.
- Уровень интерфейса служб для доступа к службам.
- Легкая база данных для сохранения пользовательских данных.

Компоненты инфраструктуры мобильной поддержки

Ниже перечислены основные компоненты инфраструктуры мобильной поддержки:

Управление мобильными устройствами (MDM) используется для настройки и управления мобильными устройствами. MDM в основном используется для распространения мобильных приложений, применения политик. Другие ключевые функции программного обеспечения MDM:

- Управление активами и группировка устройств.
- Удаленное управление программным обеспечением.
- Управление конфигурацией.
- Диагностика производительности и проверка работоспособности устройств, включая память, батарею, сетевую информацию с возможностью создания отчетов.

- Восстановление и резервное копирование данных устройств, таких как календарь, контакты, заметки и тому подобное.
- Предоставление приложений и услуг.
- Обновление микропрограммного обеспечения.
- Регистрация и создание отчетов.
- Устранение неполадок.
- Управление безопасностью на мобильных устройствах для обеспечения соблюдения политик безопасности.
- Мобильное промежуточное программное обеспечение для использования в качестве адаптеров и предоставления услуг. Компоненты мобильного промежуточного программного обеспечения также отвечают за преобразование данных, маршрутизацию, кэширование, управление.
- Другими элементами мобильной инфраструктуры являются: о Спектры для 3G и 4G.

Сквозное тематическое исследование изучения android mobile app

Ниже приведен подробный пример сквозного потока мобильного приложения Android для домена розничной торговли. Подробный пример из практики поможет нам понять различные компоненты и механизм интеграции для создания собственного мобильного приложения. Архитектура высокого уровня приложения Android представлена на рисунке 2.7.

Ниже перечислены основные требования к приложению:

- Приложение должно сканировать штрих-код для розничного решения и обеспечивать поддержку сопоставления.
- Приложение должно поддерживать несколько языков.
- Приложение должно поддерживать шифрование данных во время передачи.
- Приложение должно поддерживать функции синхронизации.

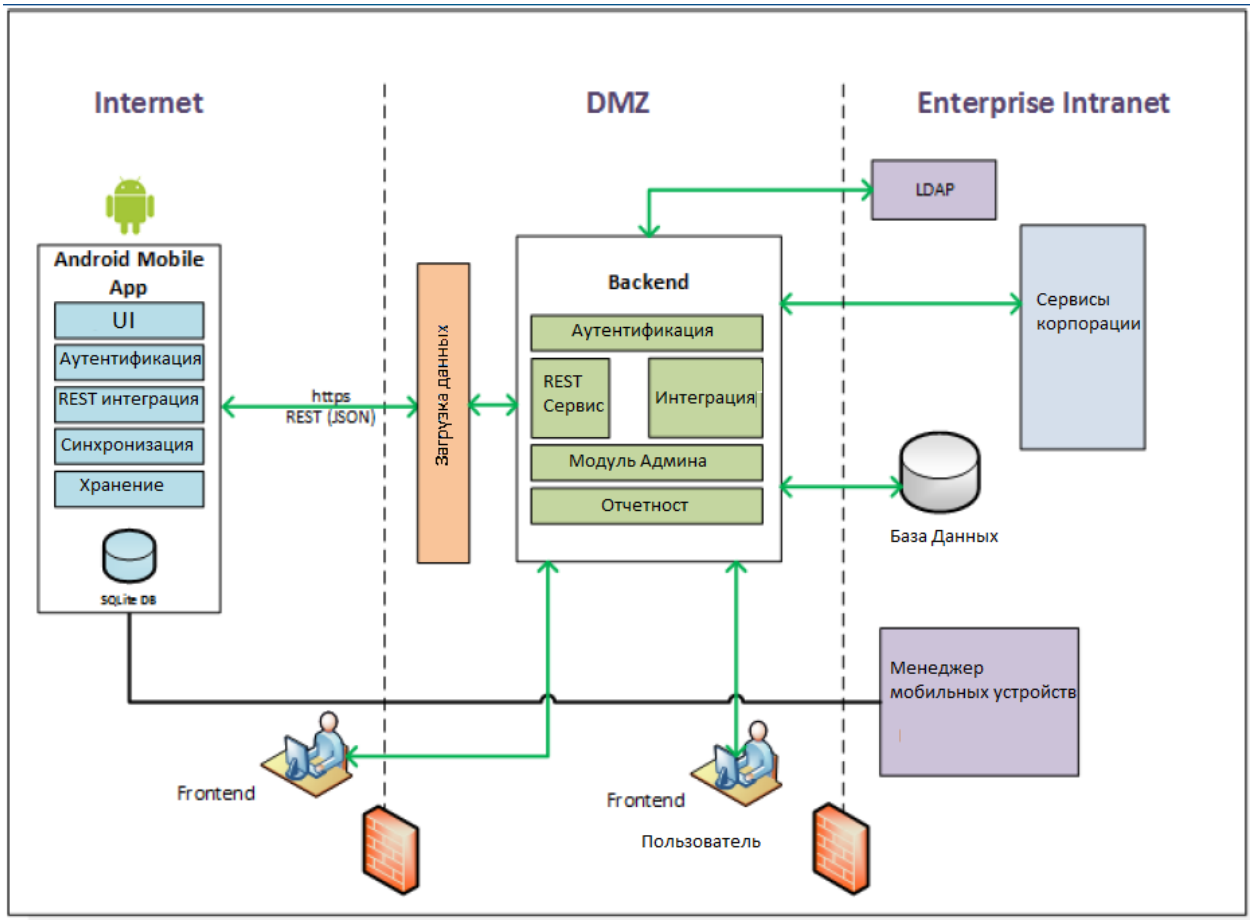

Рисунок 2.7 – Архитектура приложения Android.

Ниже описаны различные компоненты приложения Android:

Клиентский уровень:

- Решение использует зашифрованное хранилище для безопасного хранения конфиденциальных данных на устройстве. Данные могут храниться в хранилище данных SQLite для доступа в автономном режиме.

- Приложение поддерживает многоязычные экраны. Приложение будет разработано для использования языковых строк, которые будут добавлены в файлы свойств. В зависимости от языка устройства вошедшего в систему пользователя загружается соответствующий файл языковой строки.

- Приложение использует камеру устройства для сканирования штрих-кодов.

- Приложение будет использовать собственные библиотеки устройств для поддержки таких функций, как отслеживание, предоставляя координаты GPS в формате широты / долготы.

- Приложение будет использовать стандартные API-интерфейсы Google Map для всех рабочих процессов, специфичных для карты.

Интеграция

- Аутентификация приложения будет выполняться в системе Mobile Backend.

- Приложение будет интегрировано с сервисами RESTful, предоставляемыми бэкэнд-системой Mobile Delivery. Формат передачи данных будет JSON

- Канал связи будет защищен с помощью https для передачи всех данных. Приложение использует автономное хранилище данных.

- Приложение ожидает, что бэкэнд-система обеспечит извлечение дельта-данных (для автономной синхронизации). Извлечение дельта-данных необходимо для уменьшения объема данных, передаваемых по сети.

- Синхронизация данных будет осуществляться как инициируемая вручную синхронизация. Это должно обеспечить больше контроля пользователю и предотвратить периодические опросы к бэкэнду (что увеличило бы использование батареи на устройстве).

- Последнее синхронизированное время и количество объектов, измененных на устройстве, которые ожидают синхронизации с серверной частью, могут отображаться в заголовке. Это обеспечит лучшее представление о свежести данных для пользователей.

На рисунке 2.8 показаны различные компоненты в бэкэнде мобильного приложения.

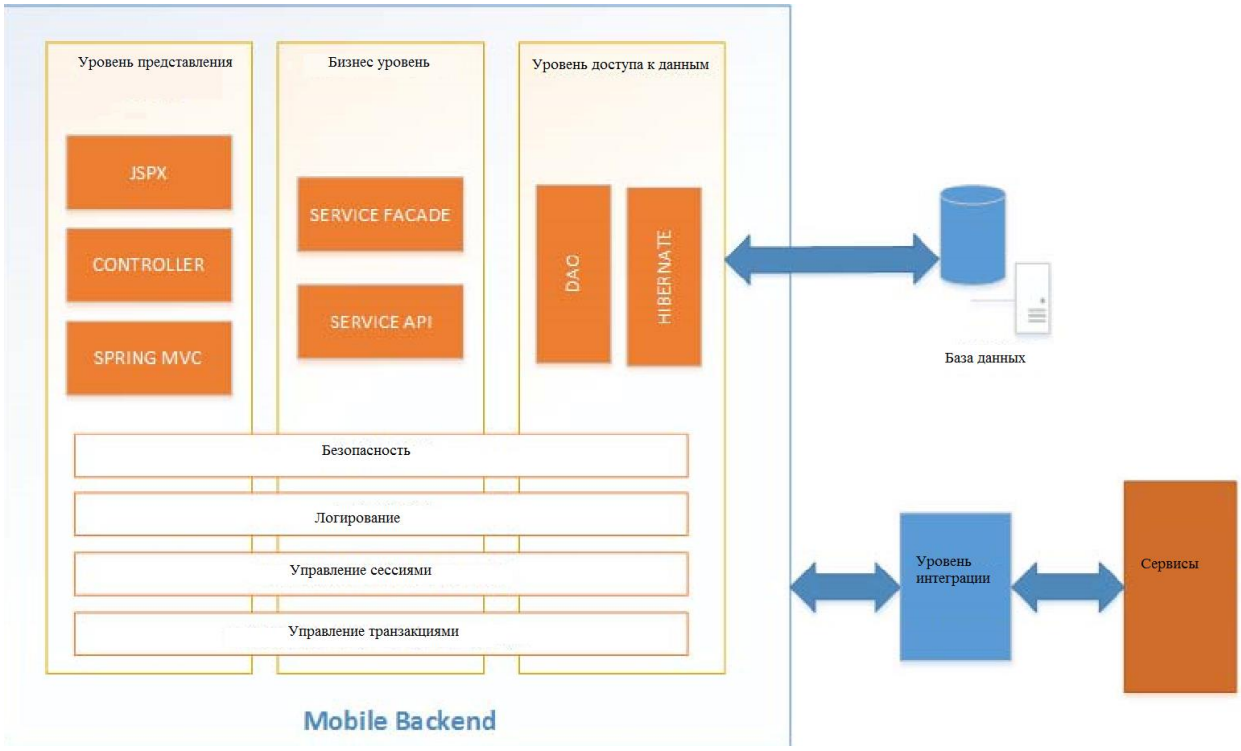

Рисунок 2.8 – Мобильные серверные компоненты

Компоненты описаны ниже:

- Веб-страницы JSPX, разработанные для мобильной платформы;
- Spring MVC для достижения слабой связи между компонентами;
- Бизнес-уровень, состоящий из шаблонов «Фасад службы» и «Реализация услуги»;

- Уровень доступа к данным будет состоять из DAO (доступ к данным). Объекты) и будет подключаться к базе данных с использованием Hibernate - Spring Framework будет использоваться для обслуживания ведения журналов, безопасности, управления сеансами и управления транзакциями
- Уровень интеграции обеспечит интерфейс со службами предприятия
- Службы REST будут доступны мобильному клиентскому приложению

2.3 Основы дизайна мобильного приложения

Разработка эффективного мобильного приложения включает в себя различные аспекты. Разработчик должен учитывать различные элементы, такие как пользовательский интерфейс, лучшие методики проектирования, оптимальные методологии интеграции, шаблоны, безопасность и т. д. Успешное развертывание мобильного приложения также включает в себя эффективное тестирование и надежные методы развертывания.

В этой главе мы рассмотрим ключевые рекомендации по разработке, глубокое погружение в дизайн пользовательского интерфейса, аспекты безопасности и элементы тестирования мобильного приложения.

Мы также предоставляем контрольный список для мобильных приложений и основываемся на аспектах энергопотребления и синхронизации мобильного приложения.

Цели для исследования:

- Понять ключевые соображения дизайна и лучшие практики мобильного приложения,
- Знать анти-шаблоны дизайна мобильного приложения,
- Глубже понять дизайн пользовательского интерфейса мобильного приложения,
- Знать сценарии развертывания и энергопотребления;
- Знать стандарты безопасности;
- Знать различные виды тестирования.

Основы дизайна и лучшие практики. Ниже приведены основные рекомендации по разработке мобильных приложений:

- Разработка приложений для обработки потерянной сети с использованием автономных функций.
- Протестируйте мобильные приложения для сценариев с низкой производительностью и корректно обрабатывайте транзакции.
- Используйте наименее ограничительную модель безопасности и предоставляйте разрешения мобильным приложениям только тогда, когда это необходимо и когда это разрешено пользователем.
- Используйте функции уведомлений, события, сообщения и индикаторы выполнения, где это необходимо.
- Обязательное шифрование данных во время передачи.

- Мобильное приложение должно использовать автономное кэширование и локальное кэширование для многофункциональных и интерактивных приложений.

Ниже приведены рекомендации по взаимодействию с пользователем:

- Обеспечьте визуальную обратную связь, используя индикатор выполнения или шаги во время обработки шагов рабочего процесса.
- Не предоставляйте глубоких меню. Минимизируйте глубину меню. Сохраняйте простую навигацию и минимальную глубину страницы.
- Обеспечить интуитивно понятный интерфейс, который не требует пояснений и прост в использовании.
- Пользователь должен иметь возможность получить необходимую информацию с минимальным количеством кликов.

Анти-паттерны Ниже приведены некоторые из распространенных антипаттернов, которых следует избегать в мобильных приложениях:

- Беспорядочная информация на страницах.
- Слишком тяжелая и глубокая иерархия страниц со сложной навигацией.
- Вызов слишком большого количества сервисов для данной страницы.
- Не тестирование мобильных приложений на всех поддерживаемых устройствах и платформах

Ниже приведен контрольный список, который можно использовать при разработке мобильного приложения:

- Обеспечить доступность элементов управления и кнопок призыва к действию на всех экранах.
- Убедитесь, что приложение обрабатывает сбой корректно(обработка исключений).
- Проверьте все проверки (максимальная длина, проверка типа, минимальная длина и т. д.) для всех форм полей ввода.
- Убедитесь, что все виды тестирования завершены для подтверждения соответствия требованиям.
- Проверьте все выравнивание, изменение размера и поведение элементов пользовательского интерфейса на всех поддерживаемых мобильных платформах и браузерах.
- Убедитесь, что ни одна из ссылок и экранов не повреждена.
- Убедитесь, что нет случайной утечки информации во время исключений и сбоев.
- Убедитесь, что мобильное приложение протестировано на все утечки памяти и выпуски ресурсов после закрытия приложений.
- Тестируется ли приложение на предмет переполнения ресурсов, пиковой нагрузки и других сценариев, которые приводят к сбою приложения?
- Поддерживает ли приложение многозадачность и многопроцессорность?

Ниже приведены точки контрольного списка с точки зрения пользовательского интерфейса:

- Использует ли приложение стандартные цвета (согласно визуальным спецификациям)?

- Является ли шрифт одинаковым для всех страниц. Правильно ли выровнен весь текст?

- Правильно ли обтекание текста вокруг изображений / графики?

- Правильно ли написан текст сообщения об ошибке на этом экране?

- Отображается ли сообщение Progress при загрузке экранов с вкладками (активных)?

Ниже приведены точки контрольного списка с точки зрения удобства использования:

- Проверьте, работает ли приложение должным образом, если устройство наклонено (книжная / альбомная).

- Проверьте, плавно ли перемещается страница.

- Убедитесь, что размер шрифта и интервал обеспечивают хорошую читаемость.

- Проверить если на каждой странице текст надписей и кнопок четкий и лаконичный

- Убедитесь, что элементы пользовательского интерфейса обеспечивают визуальную обратную связь при нажатии

Пользовательский интерфейс для мобильного приложения является одним из важнейших элементов дизайна во время его воспроизведения.

Главная роль в привлечении пользователей. Основные цели дизайна пользовательского интерфейса для мобильного приложения выглядят следующим образом:

- Простота использования: пользовательский интерфейс должен быть прост в освоении и прост в использовании.

- Высокая производительность: пользователь должен быть в состоянии найти необходимую информацию

- Более простая навигация: пользователь должен иметь возможность перемещаться по экранам

- Информационная архитектура должна быть интуитивно понятной

- Минимализация ошибок: приложение должно минимизировать частоту ошибок для конечного пользователя.

2.3 Жизненный цикл

На рисунке 2.9 показаны сквозные этапы жизненного цикла мобильного приложения. развертывание.

Рисунок 2.9 – Шаги, связанные с развертыванием мобильного приложения

Требования и бизнес-анализ: на этом этапе анализируются подробные требования пользователей и составляются все функциональные и нефункциональные требования. Также необходимо провести требования, связанные с безопасностью, производительностью, доступностью, локализацией, стандартами, модульности и т. д.

Нативный дизайн приложения: на этом этапе проектируется пользовательский опыт, информационная архитектура, необходимая для мобильного приложения. сквозная архитектура и детальный дизайн для мобильное приложение. Другим ключевым результатом, на этом этапе являются визуальный дизайн и HTML-прототипы.

Разработка нативных приложений: на этом этапе необходимо следовать всем указанным рекомендациям по разработке для разработки собственного мобильного приложения для конкретной платформы. Разработанное приложение должно быть проверено на различных эмуляторах.

Проверка мобильного приложения: различные формы тестирования, такие как тестирование устройства, интеграционное тестирование, системное тестирование, тестирование безопасности, регрессионное тестирование и Выпуск мобильного приложения: после завершения тестирования мобильное приложение должно быть развернуто в соответствующем магазине приложений. Затем следует поддержка приложения, выпуск новых версий и тд.

3 Стандарты безопасности

Во время разработки мобильных приложений важно оценивать и принимать активные меры безопасности для снижения риска безопасности. Ниже приведены некоторые из практик, которые приведут к разработке безопасного мобильного приложения:

- Все мобильные устройства должны быть защищены паролем.
- Все ключевые приложения (например, банковские приложения) должны быть защищеныы паролем.
- Все конфиденциальные и пользовательские данные должны быть зашифрованы.
- Все мобильные приложения должны представлять политику конфиденциальности, обмена данными
- Политики, юридические политики через лицензионные соглашения с конечным пользователем.
- Политики паролей должны включать ограничение длины (не менее 8 символы), сложность (использование специальных символов и буквенно-цифровой), частота смены пароля и тому подобное.
- Все основные события, такие как неудачные попытки входа в систему, сбои приложений и
- Системные события должны быть зарегистрированы.
- Для защищенных приложений и функциональных возможностей система должна
- Возможность использования многофакторную аутентификацию или управление мобильными устройствами (MDM).
- Данные при хранении и при передачи должны быть зашифрованы с использованием соответствующих стандартов шифрования.

Различные механизмы аутентификации для мобильных приложений:

- Однофакторная аутентификация: здесь пользователю предлагается ввести пароль каждый раз, когда приложение запускается или выполняется какая-либо защищенная деятельность
- Двухфакторная аутентификация: аутентификация выполняется дважды, один раз с учетными данными пользователя и второй с использованием OTP.
- Единый вход (SSO): мобильное приложение интегрировано с SSO предприятия решения для беспрепятственного доступа ко всем защищенным корпоративным приложениям.
- Другие режимы аутентификации:
- Аутентификация с использованием популярных социальных каналов, таких как Google+,Facebook, Twitter и т. Д.
- Аутентификация с использованием биометрии, например, черты лица, речь, узоры, отпечатки пальцев и т. д.

Следующие политики управления информационными рисками (IRM) должны быть применяется:

- Периодическая оценка угроз и уязвимостей всех приложений.
- Удаленные методы очистки данных для администраторов.
- Политики утилизации мобильных устройств для защиты конфиденциальных и важных для бизнеса данных.
- Сканирование на наличие вирусов и вредоносных программ следует проводить на периодической основе.
- Фильтры и сканеры должны быть установлены для предотвращения таких уязвимостей как фишинг, утечка данных, межсайтовый скриптинг и т. д.
- Должны применяться политики блокировки экрана.
- Ограничить установку приложений, таких как Jailbreak, чтобы предотвратить несанкционированное использование.

4 Исследование и создание системы оценки персонала

Оценка эффективности, также называемая обзором эффективности, оценкой эффективности, (обсуждение карьеры) развития, или оценкой работника, является методом, с помощью которого эффективность работы сотрудника документируется и оценивается. Оценки эффективности являются частью развития карьеры и состоят из регулярных проверок эффективности работы сотрудников в организациях.

Оценка эффективности - это систематический, общий и периодический процесс, который оценивает эффективность работы и производительность отдельного сотрудника в отношении определенных заранее установленных критериев и целей организации. Рассматриваются также другие аспекты отдельных сотрудников, такие как поведение гражданских лиц, достижения, потенциал для будущего улучшения, сильные и слабые стороны и т.д. Для сбора данных по РА существуют три основных метода: объективное производство, персонал и оценочная оценка.

Оценки чаще всего используются с большим разнообразием методов оценки. Исторически РА проводилась ежегодно (оценки по длинному циклу); однако многие компании переходят на более короткие циклы (каждые шесть месяцев, каждый квартал), а некоторые переходят на короткоцикловые (еженедельные, двухнедельные). Интервью могло бы функционировать как «обеспечение обратной связи с работниками, консультирование и развитие сотрудников, а также передача и обсуждение вознаграждения, статуса работы или дисциплинарных решений». Оценка показателей часто включается в системы управления эффективностью. Данная оценка помогает подчиненному ответить на два ключевых вопроса: во-первых, «Каковы ваши ожидания от меня?» во-вторых, «Как я справляюсь с вашими ожиданиями?»

Системы управления эффективностью используются для «управления и выравнивания» всех ресурсов организации с целью достижения максимально возможной производительности. «Как управление эффективностью в организации определяет в значительной степени успех или неудачу организации. Поэтому улучшение РА для всех должно быть одним из самых высоких приоритетов современных организаций». Некоторые приложения РА - это компенсация, повышение производительности, повышение по службе, завершение, проверка теста и т.д. Хотя есть много потенциальных преимуществ РА, есть также и некоторые потенциальные недостатки. Например, РА может помочь облегчить коммуникацию между менеджером и сотрудником; тем не менее, РА может привести к юридическим проблемам, если не будет выполнено надлежащим образом, так как многие сотрудники, как правило, не удовлетворены процессом РА. РА, созданные в Соединенных Штатах и признанные ими полезными, не обязательно могут передаваться в разных культурах.

Основной причиной использования оценок эффективности (РА) является повышение производительности («первоначально на уровне отдельного сотрудника и, в конечном итоге, на уровне организации»). Другие фундаментальные причины включают «в качестве основы для принятия решений о трудоустройстве (например, повышение в должности, увольнение, переводы), в качестве критериев в исследованиях, чтобы помочь с коммуникацией, установить личные цели для обучения «программам для передачи объективной обратной связи для личностного развития» в качестве средства документирования, помогающего отслеживать решения и юридические требования », а также в области управления заработной платой. Кроме того, РА могут помочь в формулировании критериев работы и отборе лиц, «которые лучше всего подходят для выполнения требуемых организационных задач». ЗП может быть частью руководства и мониторинга развития карьеры сотрудника. РА также могут быть использованы для содействия мотивации работы посредством использования систем вознаграждения

Потенциальные выгоды

Существует ряд потенциальных преимуществ управления эффективностью организации при проведении формальной оценки эффективности (РА). Было достигнуто общее согласие в том, что РА приводят к положительным последствиям для организаций. того, РА могут повысить эффективность организации. Одним из способов является то, что РА часто могут давать отдельным работникам обратную связь о своей работе. Из этого могут появиться некоторые потенциальные выгоды, такие как повышение производительности отдельных работников

Другие потенциальные выгоды включают в себя:

- Облегчение общения: общение в организациях считается важной функцией мотивации работников. Было предложено, чтобы отзывы от РА помогли минимизировать восприятие сотрудниками неопределенности. По сути, обратная связь и коммуникация между руководителями и служащими могут служить ориентиром в эффективности работы.
- Повышение внимания к сотрудникам за счет повышения доверия: поведение, мысли и / или проблемы могут отвлекать сотрудников от их работы, и проблемы доверия могут быть среди этих отвлекающих факторов. Такие факторы, которые потребляют психологическую энергию, могут снизить производительность труда и привести к тому, что работники упустят из виду организационные цели. Правильно построенные и используемые РА способны снижать отвлекающие факторы и стимулировать доверие в организации.
- Постановка целей и желаемое повышение эффективности: организации считают эффективным согласование целей и результатов работы отдельного работника с целями организации. РА предоставляют пространство для обсуждения в сотрудничестве этих индивидуальных и организационных

целей. Сотрудничество также может быть выгодным, поскольку приводит к тому, что сотрудники принимают и удовлетворяют результаты оценки.

- Улучшение производительности: хорошо сконструированные РА могут быть ценными инструментами для общения с сотрудниками в том, что касается эффективности их работы в соответствии с ожиданиями организации. «На организационном уровне многочисленные исследования показали положительную связь между практикой управления человеческими ресурсами (HRM)» и улучшением работы как на индивидуальном, так и на организационном уровнях.

- Определение потребностей в обучении: «Обучение и развитие сотрудников являются важнейшими компонентами, помогающими организации реализовать стратегические инициативы». Утверждалось, что для того, чтобы РА были действительно эффективными, должны быть предложены возможности после обучения для обучения и развития в проблемных областях, как определено оценкой. РА могут особенно способствовать выявлению потребностей в обучении новых сотрудников. Наконец, РА могут помочь в установлении и контроле карьерных целей сотрудников.

Методы сбора данных. Существует три основных метода, используемых для сбора данных об оценке эффективности (РА): объективное производство, персонал и оценочная оценка. Судебные оценки чаще всего используются с большим разнообразием методов оценки.

Объективное производство. Метод объективного производства состоит из прямых, но ограниченных мер, таких как показатели продаж, производственные показатели, электронный мониторинг производительности работников, занимающихся вводом данных, и т. д. Меры, используемые для оценки эффективности, будут зависеть от работы и ее обязанностей. Хотя эти меры имеют дело с однозначными критериями, они, как правило, являются неполными из-за загрязнения критерия и недостатка критерия. Критерий загрязнения относится к той части фактических критериев, которая не связана с концептуальными критериями. Другими словами, изменчивость в работе может быть вызвана факторами, не зависящими от работника. Дефицит критерия относится к той части концептуальных критериев, которая не измеряется фактическими критериями. Другими словами, количество продукции не обязательно указывает на качество продукции. Оба типа недостаточности критериев приводят к снижению достоверности меры.

Независимо от того, что объективные производственные данные не являются полным отражением эффективности работы, такие данные имеют отношение к производительности работы.

Успешная продуктивная гипотеза. Гипотеза о счастливой продуктивной работе гласит, что самые счастливые рабочие - самые продуктивные, а самые продуктивные - самые счастливые. Тем не менее, после десятилетий исследований взаимосвязь между удовлетворенностью работой и производительностью работы дает лишь слабую положительную корреляцию.

Опубликованный в 2001 году «Психологическим бюллетенем» метаанализ 312 научных исследований выявил неисправленную корреляцию 0,18. Эта корреляция намного слабее, чем предсказывает гипотеза о счастливой продуктивной работе.

Кадровый метод - это регистрация поведения, связанного с отзывом (то есть прогулы, несчастные случаи). Большинство организаций считают неоправданные пропуски показателями плохой работы, даже при прочих равных условиях, однако это обусловлено недостатком критериев. Количество отсутствий работника не отражает его преданность работе и обязанностям. Особенно для рабочих мест, несчастные случаи часто могут быть полезным индикатором плохой работы, но это также зависит от загрязнения по критерию, потому что ситуационные факторы также способствуют несчастным случаям. Еще раз, оба типа неадекватности критериев приводят к снижению достоверности меры. Хотя чрезмерное отсутствие на работе и / или несчастные случаи часто указывают на плохую работу, а не на хорошую работу, такие данные о персонале не являются всеобъемлющим отражением работы сотрудника.

Судебная оценка, по-видимому, представляет собой совокупность методов и, как таковая, может рассматриваться как методология. Общий подход к получению ООПТ осуществляется с помощью оценщиков. Поскольку оценщики являются людьми, в данных всегда будут присутствовать некоторые ошибки. Наиболее распространенными типами ошибок являются ошибки снисхождения, ошибки центральной тенденции и ошибки, возникающие в результате эффекта ореола. Эффект ореола характеризуется тенденцией оценивать человека, который является исключительно сильным в одной области выше, чем заслуженный в других областях. Это противоположно эффекту Рога, когда человек оценивается ниже, чем заслуженный в других областях из-за чрезвычайного дефицита в одной дисциплине. Эти ошибки возникают главным образом из социального познания, и теория заключается в том, что то, как мы оцениваем и оцениваем других людей в различных контекстах, связано с тем, как мы «собираем, обрабатываем и классифицируем информацию».

Важной частью этого метода является обучение. Обучение рейтеров - это «процесс обучения рейтеров для более точной оценки производительности, обычно достигаемой за счет уменьшения частоты ошибок ореола, снисхождения и центральной тенденции». Обучение Rater также помогает оценщикам «разработать общую систему ориентиров для оценки» отдельных показателей. Многие исследователи и респонденты поддерживают стремление к эффективному обучению. Тем не менее, отмечается, что такое обучение является дорогостоящим, трудоемким и действительно функциональным только для оценки поведения.

Еще одна вещь, которую нужно иметь в виду, это влияние мотивации оценщиков на оценку судей. Нередки случаи, когда рейтинговая инфляция возникает из-за мотивации рейтингов (т. Е. «Организационно вызванное

давление, которое вынуждает оценщиков позитивно оценивать ставки»). Как правило, оценщики мотивированы давать более высокие оценки из-за отсутствия организационных санкций в отношении точных / неточных оценок, желания оценщика поощрять повышение по службе, повышения заработной платы и т. Д., Склонности оценщика избегать негативных реакций со стороны подчиненных и наблюдения, что выше рейтинги ставок благоприятно отражаются на оценщике.

Основными методами, используемыми при оценке работы судей, являются:

- Графическая рейтинговая шкала: графические рейтинговые шкалы (см. Шкалу (социальные науки)) являются наиболее часто используемой системой в РА. По нескольким различным факторам подчиненные оцениваются по «сколько» этого фактора или черты они обладают. Как правило, оценщики используют 5- или 7-балльную шкалу; тем не менее, существует целых 20 баллов.
- Методы сравнения сотрудников: вместо того, чтобы оценивать подчиненных по заранее установленным критериям, их сравнивают друг с другом. Этот метод устраняет основные ошибки, связанные с тенденцией и снисходительностью, но все же допускает ошибки эффекта ореола. Метод ранжирования имеет рейтинг, оценивающий подчиненных от «лучшего» до «худшего», но насколько хорошо или плохо действительно оценивается производительность, неизвестно. Метод парного сравнения требует, чтобы оценщик выбрал двух «лучших» подчиненных из группы в каждом измерении, а затем оценил индивидуумов в соответствии с количеством раз, когда каждый подчиненный был выбран в качестве одного из «лучших». Метод принудительного распределения подходит для больших групп тарифов. Оценщики оценивают каждого подчиненного по одному или нескольким измерениям, а затем помещают (или «принудительно», если хотите) каждого подчиненного в нормальное распределение категории 5–7.
- Метод топ-градации может быть применен к методу принудительного распределения. Этот метод идентифицирует подчиненных с самыми низкими показателями на 10% в соответствии с принудительным распределением и отклоняет их, оставляя подчиненных с более высокими показателями на 90%.
- Поведенческие контрольные списки и шкалы: поведение более определенное, чем черты характера.
- Метод критических инцидентов (или метод критических инцидентов) касается «специфического поведения, указывающего на хорошую или плохую работу». Супервайзеры фиксируют поведение, которое, по их мнению, имеет отношение к эффективности работы, и постоянно учитывают хорошее и плохое поведение. Затем может последовать обсуждение производительности. Оценочные шкалы поведения (BARS) объединяют метод критических инцидентов с методами оценочной шкалы путем оценки эффективности по шкале, но с точками шкалы, привязанными к поведенческим инцидентам. Обратите внимание, что BARS являются специфическими для работы. В

подходе шкалы поведенческих наблюдений (BOS) к оценке производительности сотрудников также оценивают с точки зрения критических инцидентов. В этом отношении он похож на БАРС. Тем не менее, показатель оценки BOS зависит от частоты критических инцидентов, поскольку они наблюдаются в течение определенного периода. Рейтинги присваиваются по пятибалльной шкале.

- Поведенческие инциденты для рейтинговой шкалы разрабатываются так же, как и для BARS, путем идентификации супервизорами или другими экспертами в данной области. Точно так же методы BOS встречают равные возможности трудоустройства, потому что они связаны с фактическим поведением, необходимым для успешного выполнения работы.

В то время как оценка может быть выполнена по отчетным отношениям (обычно сверху вниз), чистая оценка может включать в себя оценку равных и самооценку. Коллегиальная оценка - это когда коллеги проводят оценку по горизонтальным (аналогичная функция) и вертикальным (различная функция) отношениям. Самооценки - это когда люди оценивают себя. Существует три распространенных метода оценки коллегами. Номинация «равный-равный» включает в себя назначение каждого члена группы, который, по его мнению, является «лучшим» в определенном измерении эффективности. В одноранговых рейтингах каждый член группы оценивает друг друга по набору показателей производительности. Равное ранжирование требует, чтобы каждый член группы оценил всех своих членов от «лучших» до «худших» по одному или нескольким параметрам производительности.

Самооценки: для самооценки люди оценивают и оценивают свое собственное поведение и работу.

Товарищеские оценки: члены группы оценивают и оценивают работу своих коллег-членов группы. Там обычно используется графическая шкала оценок для самооценки. Позитивная снисходительность имеет тенденцию быть проблемой с самооценками. Оценки сверстников от нескольких членов группы часто называют обзорами эффективности, основанными на толпе, и решают многие проблемы с оценками сверстников только от одного участника.

360-градусная обратная связь: 360-градусная обратная связь - это множественные оценки сотрудников, которые часто включают в себя оценки старшего (-их), сверстников и самого себя.

Согласованная оценка производительности: согласованная оценка производительности (NPA) является новым подходом для улучшения связи между руководителями и подчиненными и для повышения производительности труда сотрудников, а также может быть адаптирована к альтернативной модели посредничества для конфликтов подчиненного руководителя.

Фасилитатор встречается отдельно с руководителем и подчиненным, чтобы подготовить три списка. Что сотрудники делают хорошо, где работник недавно улучшился, и области, где работнику все еще нужно улучшаться.

Поскольку подчиненный будет представлять свои списки первыми во время совместного сеанса, это снижает защитное поведение. Кроме того, подчиненный приходит на совместную сессию не только готовым поделиться областями необходимого улучшения, но и приносит конкретные идеи относительно того, как эти улучшения могут быть сделаны. НПА также очень сильно фокусируется на том, что у сотрудников хорошо, и требует не менее двадцати минут похвалы при обсуждении того, что сотрудник делает хорошо. Роль фасилитатора - это роль тренера на предварительных совещаниях, а на совместных сессиях руководитель и подчиненные в основном разговаривают друг с другом с небольшим вмешательством фасилитатора.

В общем, оптимальный процесс РА включает в себя сочетание нескольких методов оценки. Одна общая рекомендация состоит в том, что оценка переходит от самооценки к экспертной оценке к оценке руководства - в этом порядке. Начиная с самооценки, помогает избежать конфликтов. Обратная связь с коллегами обеспечивает подотчетность коллег, что может дать лучшие результаты, чем подотчетность руководству. Оценка руководства идет последним в связи с необходимостью признания властью и предотвращения конфликтов в случае разногласий. Как правило, рекомендуется, чтобы РА проводился в более короткие циклы, чтобы избежать обсуждений с высокими ставками, как это обычно бывает при оценках длинных циклов

5 Построение архитектуры базы системы оценки персонала и создание WEB API

Для разработки данного программного обеспечения были составлены следующие бизнес модели системы

Первая схема включает в себя основные сущности системы: людей и их оценку. Бизнес правило построено следующим:

Сущность 1. User

Таблица включает в себя следующие поля:

- User_Id (Number)
- User_Name (Varchar2)
- User_Resp (Varchar2)
- Employee_Number (Varchar2)
- Email (Varchar2)

Первичный ключ User_Id

Сущность 2. User_Appraisal_Tree

Таблица велючает в себя следующие поля:

- User_Appraisal_Tree_Id (Number)
- User_Id (Number)
- Appraisal_Id (Number)
- User_Manager (Number)
- Job_Level (Number)

Первичный User_Id

Сущность 3. Appraisal

Таблица включает в себя следующие поля:

- Appraisal_Id (Number)
- Begin_Date (Date)
- End_Date (Date)
- KPI_Begin_Date (Date)
- KPI_End_Date (Date)

Первичный ключ Appraisal_Id

Каждый сотрудник комании имеет свое текущее положение на период оценки, где сотрудник компании – это таблица user, положение сотрудника user_appraisal_tree, а период оценки ѳ appraisal.

Сущности связаны между собой отношением один ко многим:

Один сотрудник может иметь несколько положений и периодов оценки, в свою очередь, каждое положение сотрудника могут относится только к одному сотруднику и периоду. Схема представлена ниже(Рисунок 4.1)

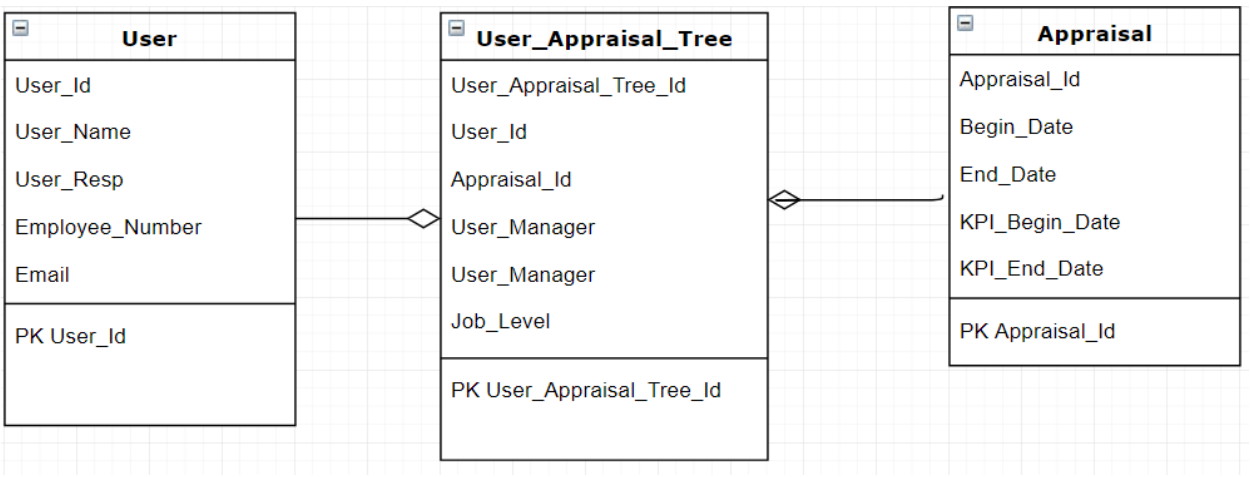

Рисунок 5.1 – Первая схема

Вторая схема включает в себя и дополняет первую описанием целей и задач сотрудника.

Сущность 4 KPI

Таблица включает в себя

- KPI_Id (Number)
- User_Appraisal_Tree_Id (Number)
- Status_Id (Number)
- Comment (Varchar2)

Первичный ключ PK KPI_Id

Сущность 5 KPI_Details

Таблица включает в себя

- KPI_Details_Id (Number)
- KPI_Id (Number)
- KPI_Name (Varchar2)

Первичный ключ PK KPI_Details

Сущность 6 KPI_Status

- Status_Id(Number)
- Status_Name (Varchar2)
- Status_Description (Varchar2)

Первичный ключ PK Status_Id

Для каждого положения сотрудника есть заголовок задачи для сотрудника(KPI). Каждому заголовку соотносится линии по данному заголовку. В зависимости от статуса заголовка целей, можно определить этап, на котором выполняется задача. Например, статус ознакомлен означает, что руководитель поставил задачу сотрудику и сотрудник уже ознакомился с задачами, в то время, как статус новый означает, что сотрудник еще не ознакомлен с задачами (Рисунок 5.2)

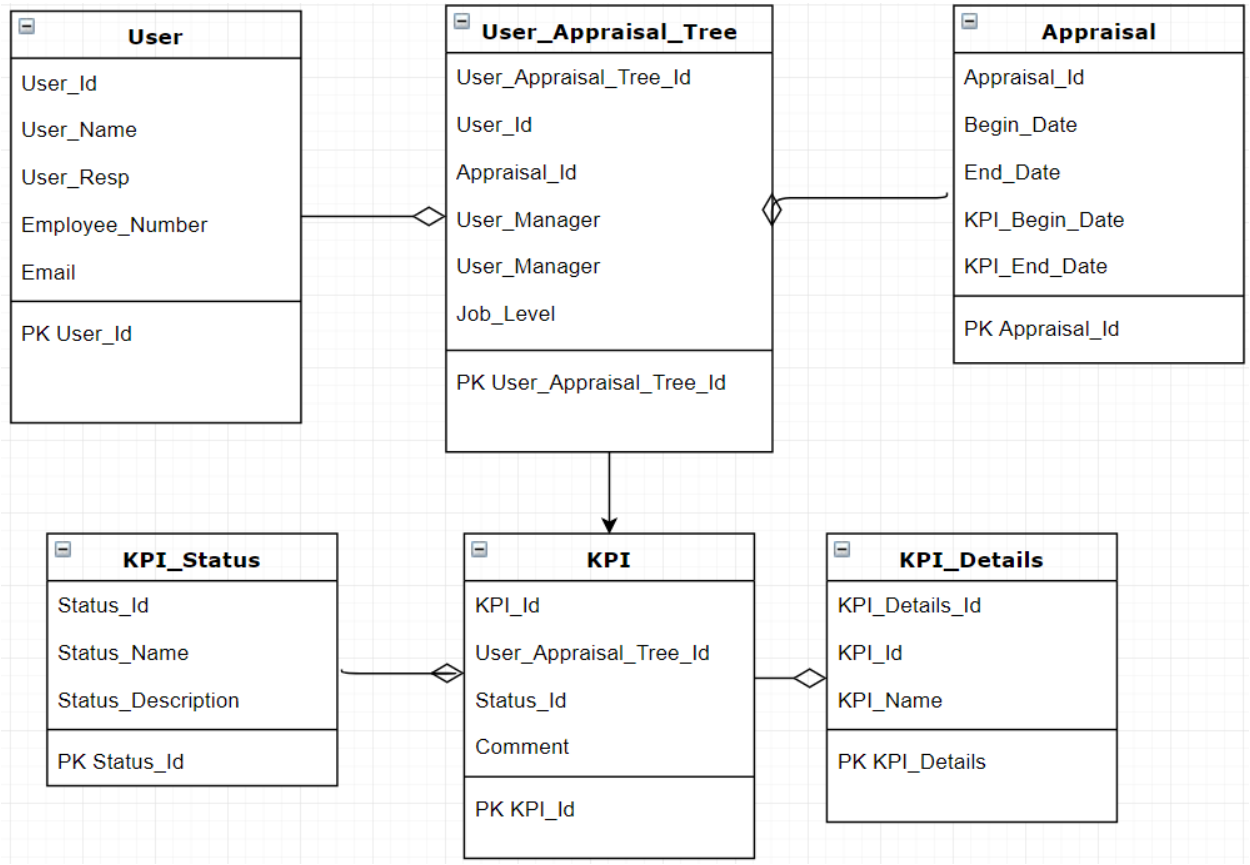

Рисунок 5.2 – Схема с учетом Appraisal и KPI

Третья схема включает в себя бизнес процесс оценки сотрудника, его балл, показатели и т.д.

Сущность 7 Rating

- Rating_Id (Number)
- User_Appraisal_Tree_Id (Number)
- KPI_Id (Number)
- Status_Id (Number)
- Rating_Total (Number)

Первичный ключ PK Rating_Id

Сущность 8 Rating_Details

- Rating_Details_Id (Number)
- Rating_Id (Number)
- Rating (Number)
- KPI_Details_Id (Number)

Первичный ключ PK Rating_Details_Id

Для каждого положения сотрудника есть заголовок оценки, включающий в себя статус и итоговый балл сотрудника за определенный оценочный период.

Каждому заголовку оценки соответствуют детали оценки, которые относятся к KPI_Details, т.е. каждой цели и задачи сотрудника один к одному. Каждая задача имеет только одну оценку(Рисунок 5.3)

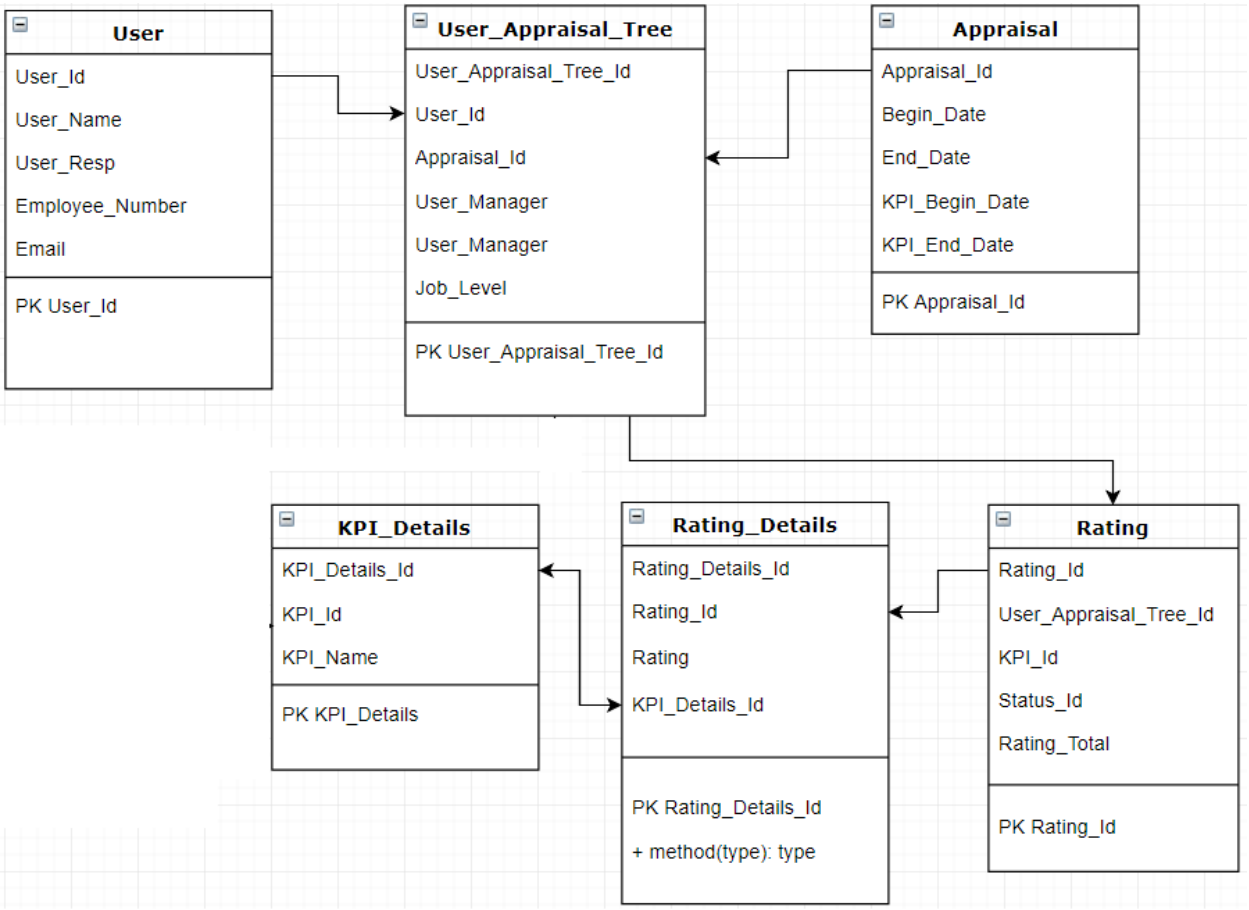

Рисунок 5.3 – Схема с учетом детализированных таблиц

Диаграммы UML

Ниже на рисунках представлены диаграммы UML

Диаграммы использования на рисунках 5.4-5.6

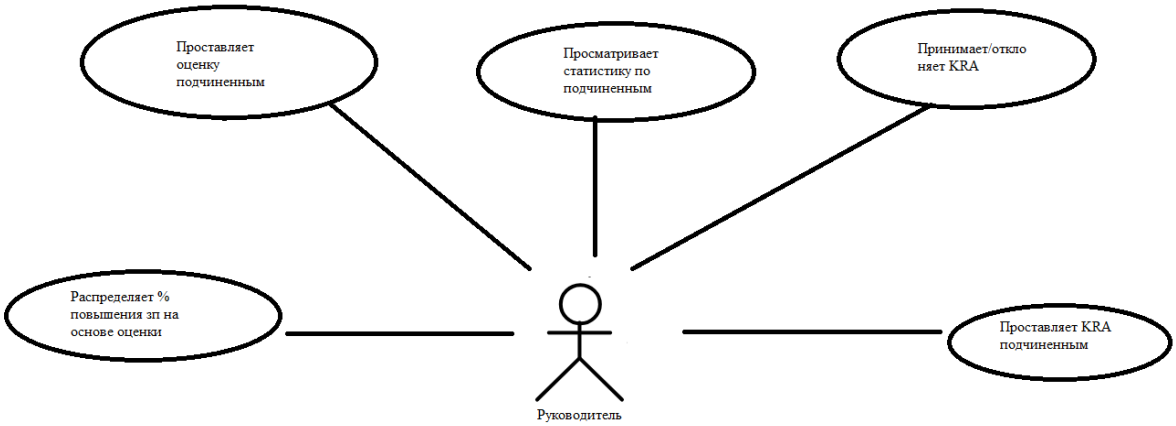

Рисунок 5.4 – Диаграмма использования

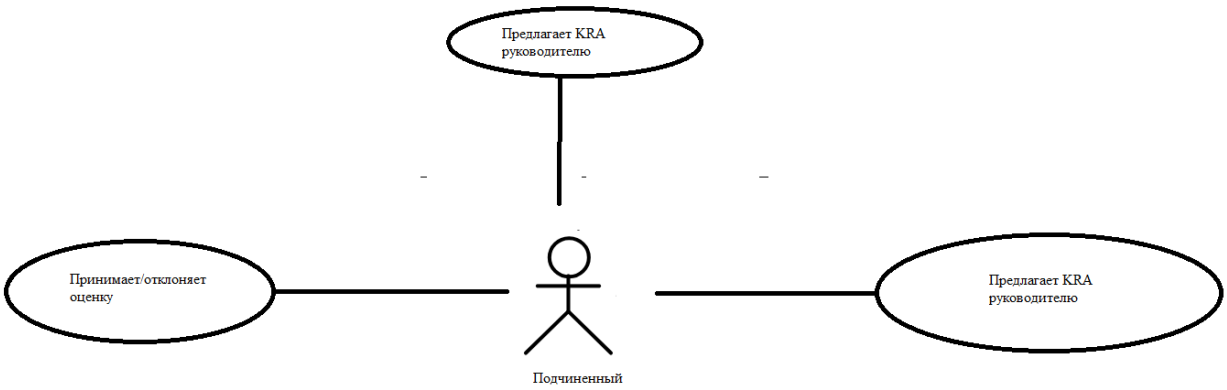

Рисунок 5.5 – Диаграмма использования

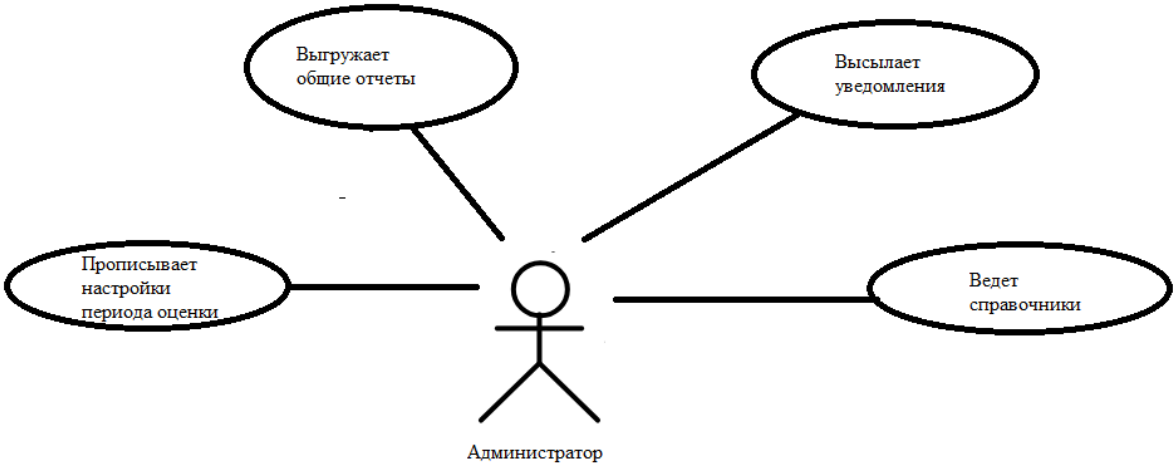

Рисунок 5.6 – Диаграмма использования

Диаграммы последовательностей представлены на рисунках 5.7-5.9

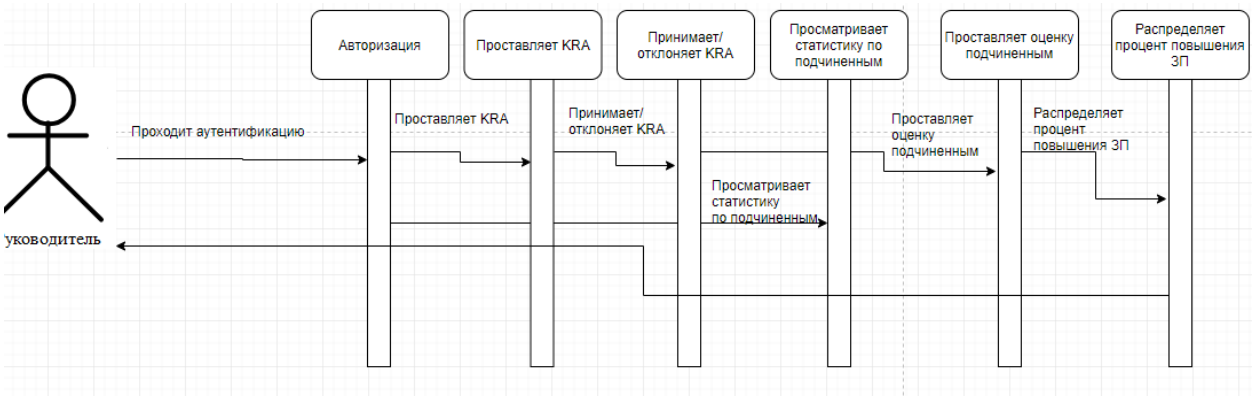

Рисунок 5.7 – Диаграмма последовательностей

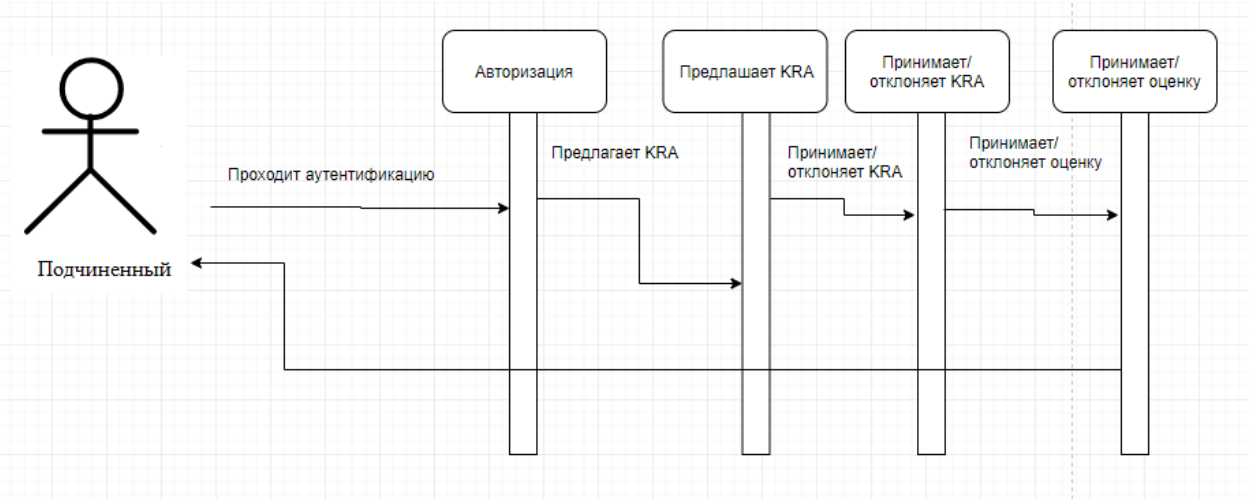

Рисунок 5.8 – Диаграмма последовательностей

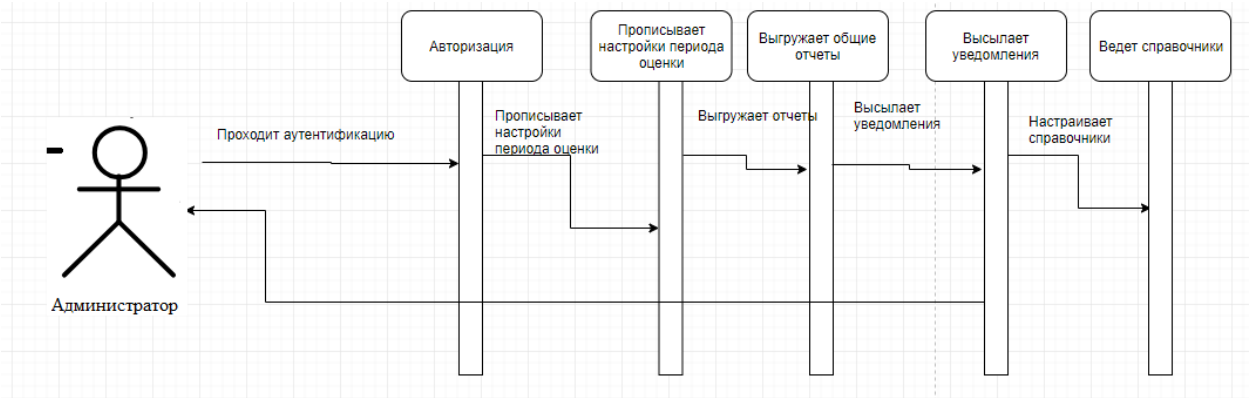

Рисунок 5.9 – Диаграмма последовательностей

Блок схемы представлены на рисунках 5.10-5.12

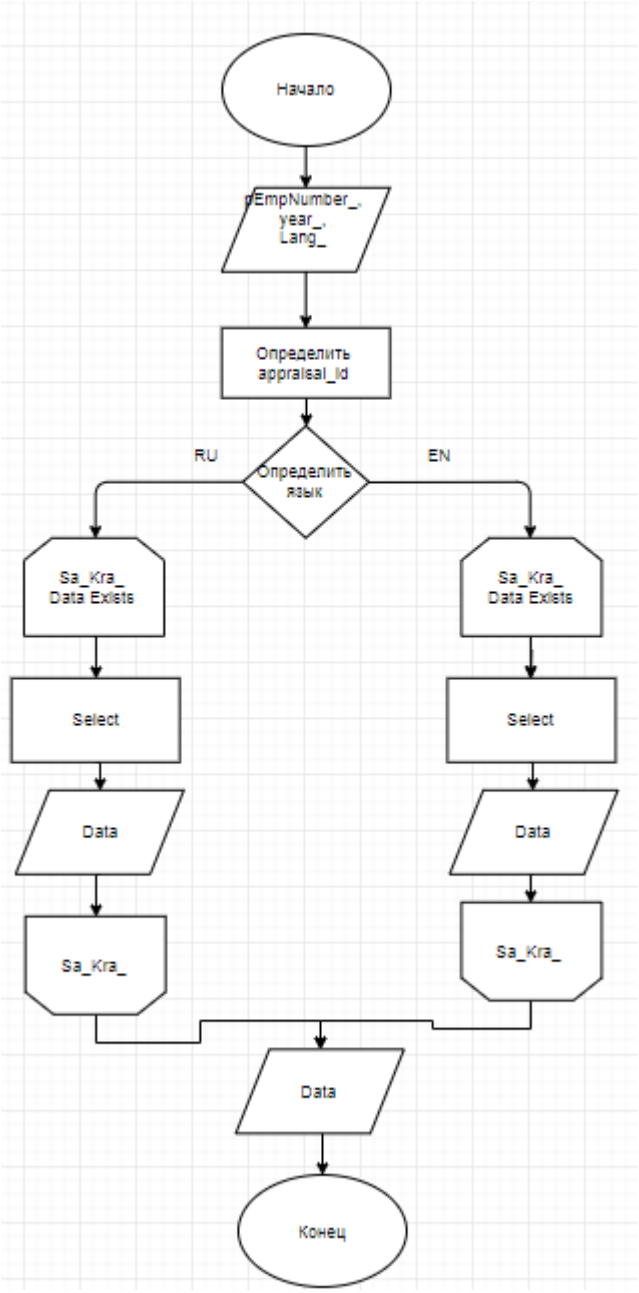

Рисунок 5.10– Блок схема функции Read_Form_By_Id

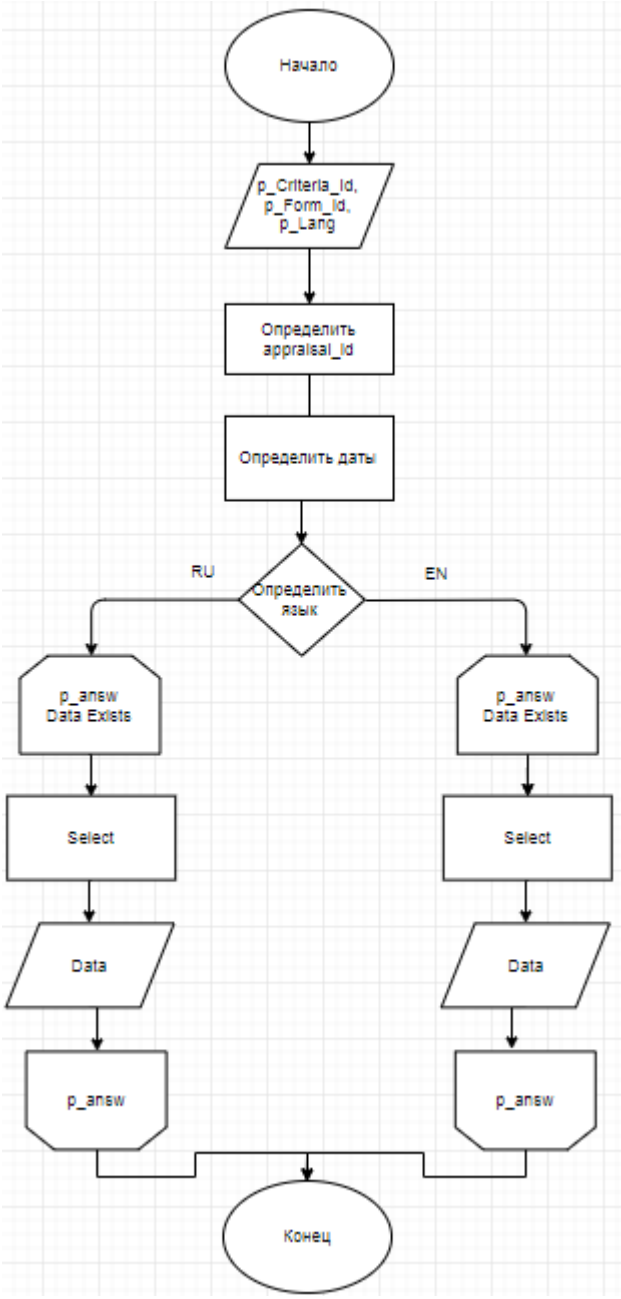

Рисунок 5.11– Блок схема функции Get_Rad_Answ

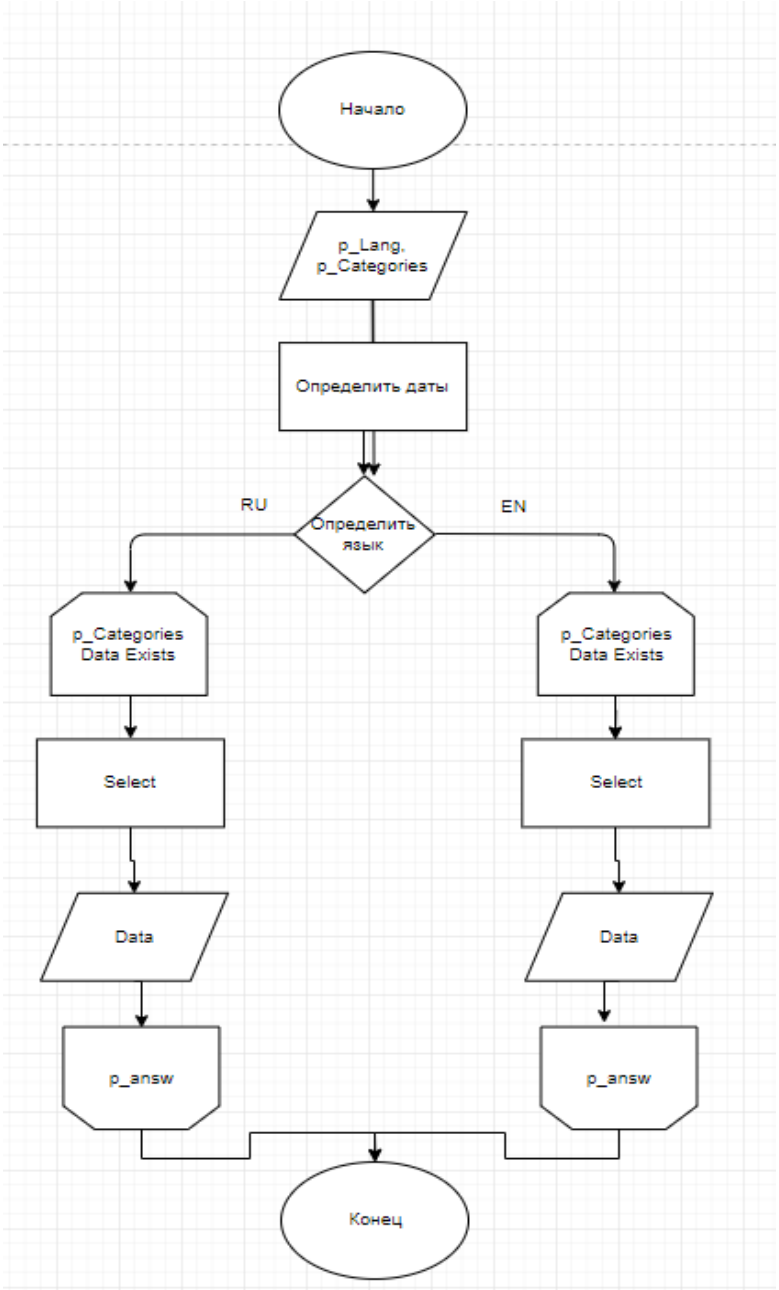

Рисунок 5.12– Блок схема функции Get_Categories

Интерфейс системы будет делится на несколько типов в зависимости от роли пользователя. Роль пользователя будет определяться функций get_user_role (Приложение Б) функция принимает значение табельного сотрудника и возвращает значение false или true, в зависимости от того есть у сотрудника подчиненные или нет. В зависимости от возвращенного значения будет определятся интерфейс программы.

Интерфейс со стороны менеджера выглядит следующим образом:

Менеджер имеет две вкладки, «Home» и «My Team», где первая - вкладка как подчиненного. В данной вкладке сотрудник может видеть свои KPI и оценку, а также принимать, отклонять свои задачи и оценку

вторая вкладка содержит значение моей команды где пользователь системы может видеть своих подчиненных, а также проставлять им задачи и оценки(Рисунок 5.13)

Рисунок 5.13 – Моя команда

Цели и задачи определяются в следующей форме при проваливании на сотрудника из раздела KPI (Рисунок 5.14)

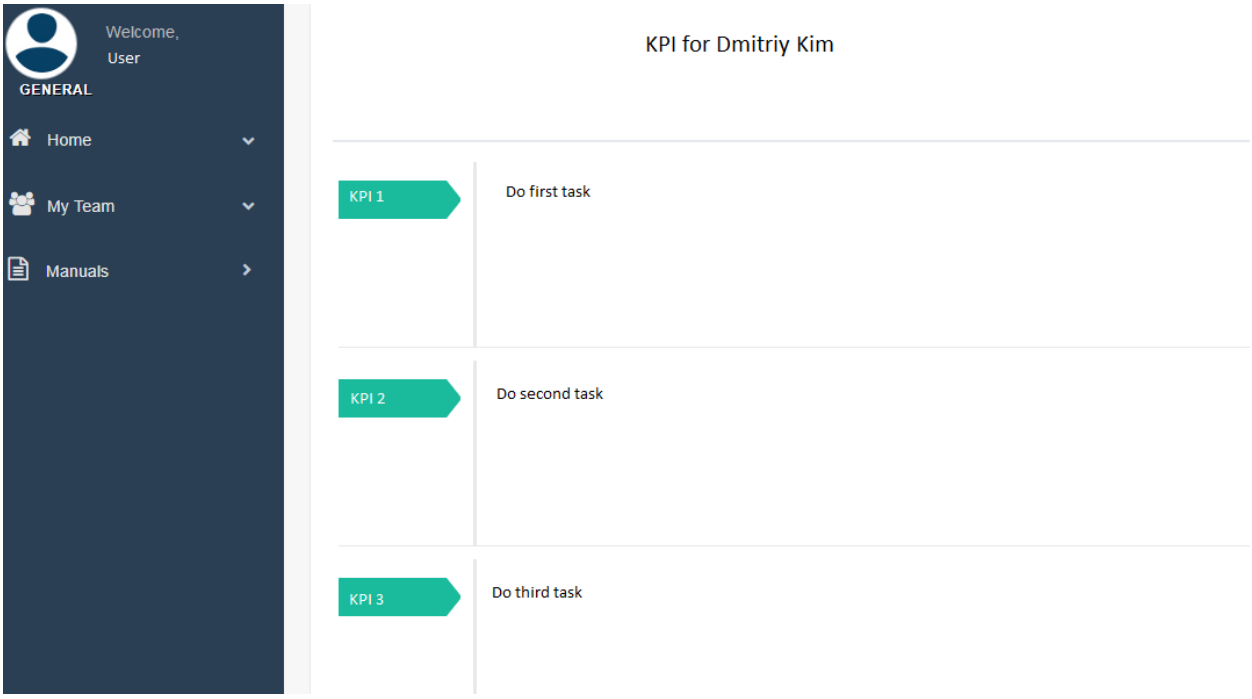

Рисунок 5.14 - KPI

Оценка сотрудника производится на нижеследующей форме оценки (Рисунок 5.15)

Рисунок – 5.15 Форма оценки

Web API - это интерфейс прикладного программирования для веб-сервера или веб-браузера. Это концепция веб-разработки, обычно ограниченная клиентской частью веб-приложения (включая любые используемые веб-платформы), и, таким образом, обычно не включает в себя подробности реализации веб-сервера или браузера, такие как SAPI или API, если они не являются общедоступными для удаленного веб-приложения.

Для создания WEB API используется среда разработки Microsoft Visual Studio. Навигация по созданию:

- В меню «Файл» выбирается «Создать»> «Проект».
- Выбирается основной шаблон веб-приложения ASP.NET.
- В диалоговом окне Новое веб-приложение ASP.NET Core - TodoApi выберите версию ASP.NET Core. Выберите шаблон API и нажмите OK. (Рисунок 5.16)

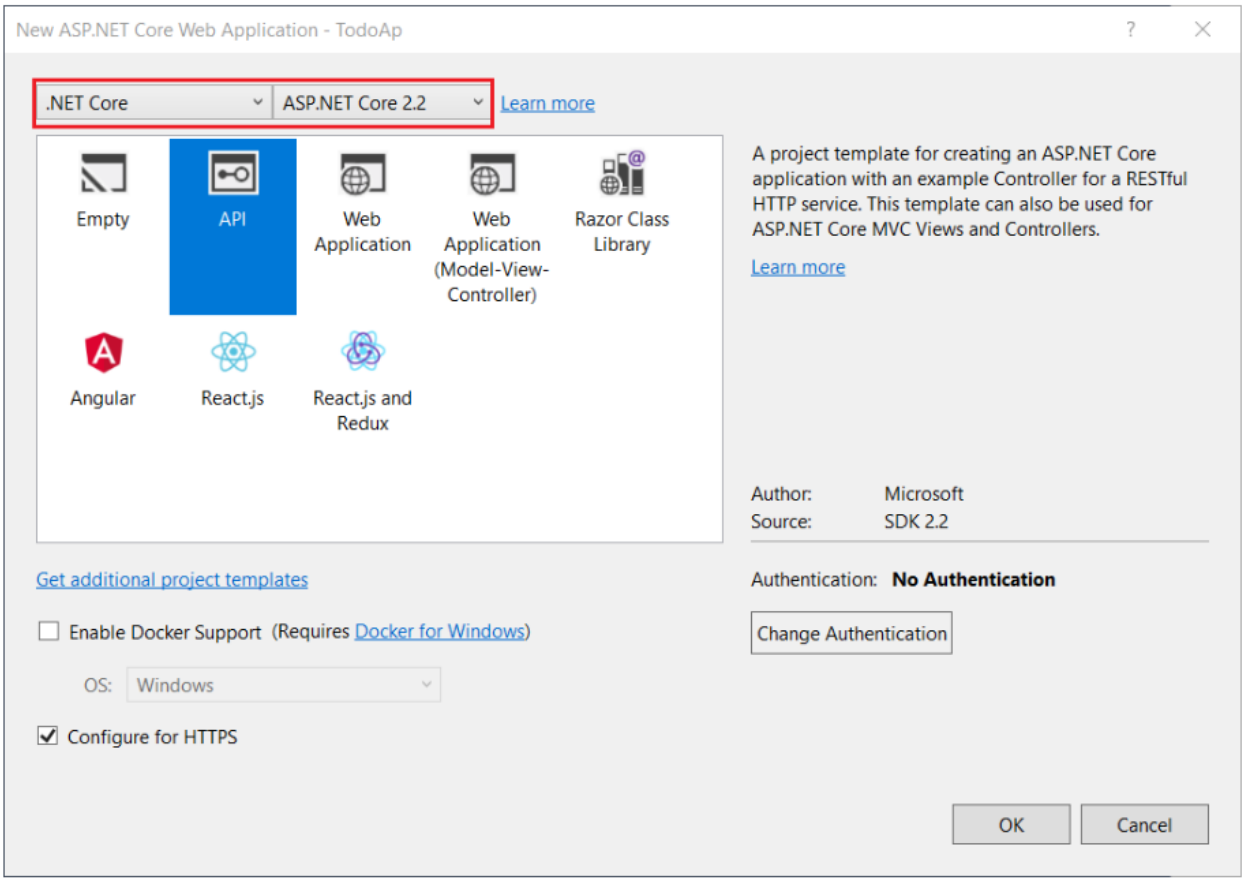

Рисунок 5.16 – Web Api ASP.NET

Модель – это набор классов, представляющих данные, которыми управляет приложение. Модель для этого приложения - единственный класс TodoItem.

Для создания модели классов используем команды ниже

В обозревателе решений правой кнопкой мыши проект. При выборе «Добавить»> «Новая папка». Дается имя папке, н-р, Models.

Правой кнопкой мыши папку «Модели» и выберите «Добавить»> «Класс». Называется класс TodoItem, затем Добавить.

Код шаблона заменяется следующим кодом:

```
namespace TodoApi.Models
{
    public class TodoItem
    {
        public long Id { get; set; }
        public string Name { get; set; }
        public bool IsComplete { get; set; }
    }
}
```

Свойство Id действует как уникальный ключ в реляционной базе данных.

Классы моделей могут находиться в любом месте проекта, но папка Models используется по соглашению.

Контекст базы данных - это основной класс, который координирует функциональные возможности Entity Framework для модели данных. Этот класс создается производным от класса «Microsoft.EntityFrameworkCore.DbContext».

Правой кнопкой мыши папку «Модели» и выбирается «Добавить»> «Класс». называется класс TodoContext и затем «Добавить».

```
using Microsoft.EntityFrameworkCore;

namespace TodoApi.Models
{
    public class TodoContext : DbContext
    {
        public TodoContext(DbContextOptions<TodoContext> options)
            : base(options)
        {
        }

        public DbSet<TodoItem> TodoItems { get; set; }
    }
}
```

Зарегистрировать контекст базы данных

В ASP.NET Core такие службы, как контекст БД, должны быть зарегистрированы в контейнере внедрения зависимостей (DI). Контейнер предоставляет сервис для контроллеров.

```
// Unused usings removed
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using TodoApi.Models;

namespace TodoApi
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
services to the
        //container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<TodoContext>(opt =>
                opt.UseInMemoryDatabase("TodoList"));

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
        }
```

```
        // This method gets called by the runtime. Use this method to configure
the HTTP
        //request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                // The default HSTS value is 30 days. You may want to change this
for
                // production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseMvc();
        }
    }
}
```

Добавление контроллера

- Щелкните правой кнопкой мыши папку «Контроллеры».
- Выберите Добавить> Новый элемент.
- В диалоговом окне «Добавить новый элемент» выберите шаблон класса контроллера API.
- Назовите класс TodoController и выберите Добавить.(Рисунок 5.17)

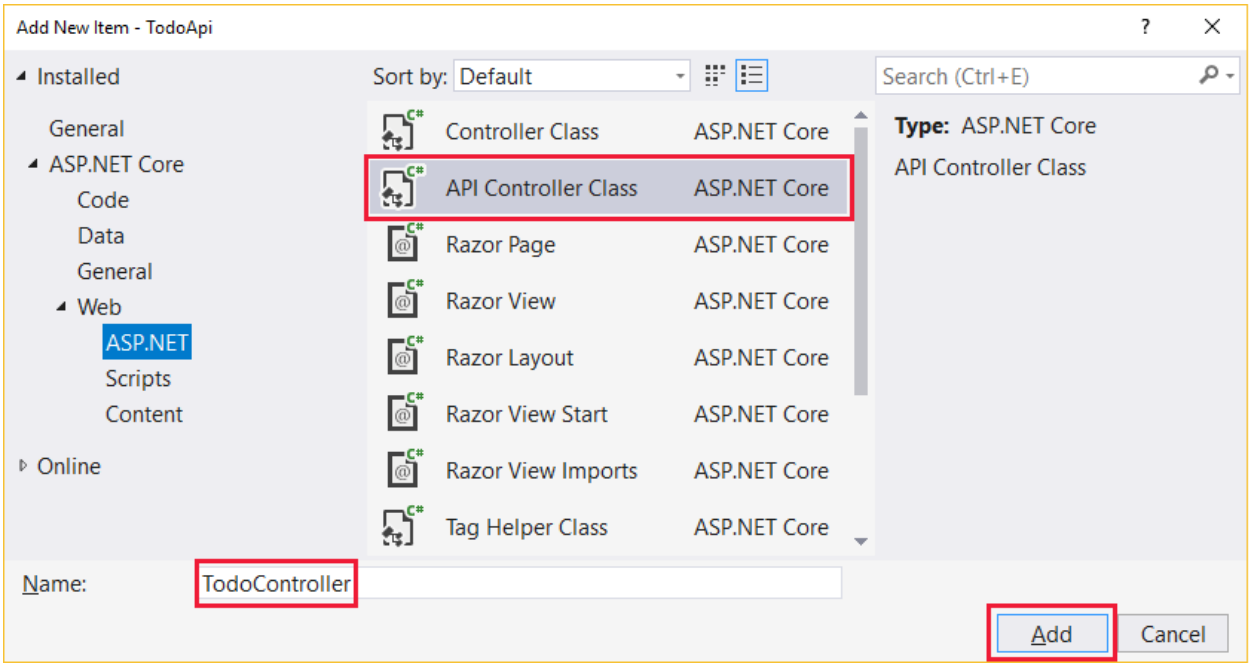

Рисунок 5.17 – Создание класса

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using TodoApi.Models;

namespace TodoApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TodoController : ControllerBase
    {
        private readonly TodoContext _context;

        public TodoController(TodoContext context)
        {
            _context = context;

            if (_context.TodoItems.Count() == 0)
            {
                // Create a new TodoItem if collection is empty,
                // which means you can't delete all TodoItems.
                _context.TodoItems.Add(new TodoItem { Name = "Item1" });
                _context.SaveChanges();
            }
        }
    }
}
```

Определяет класс контроллера API без методов.

Украшает класс атрибутом [ApiController]. Этот атрибут указывает, что контроллер отвечает на запросы веб-API. Для получения информации о конкретных поведениях, которые включает атрибут, см. Аннотация с атрибутом ApiController.

Используется DI для внедрения контекста базы данных (TodoContext) в контроллер. Контекст базы данных используется в каждом из методов CRUD в контроллере.

Добавляет элемент с именем Item1 в базу данных, если база данных пуста. Этот код находится в конструкторе, поэтому он запускается каждый раз, когда появляется новый HTTP-запрос. Если вы удалите все элементы, конструктор снова создаст Item1 при следующем вызове метода API. Так что может показаться, что удаление не сработало, когда оно действительно сработало.

```
// GET: api/Todo
[HttpGet]
public async Task<ActionResult<IEnumerable<TodoItem>>> GetTodoItems()
{
    return await _context.TodoItems.ToListAsync();
}

// GET: api/Todo/5
[HttpGet("{id}")]
public async Task<ActionResult<TodoItem>> GetTodoItem(long id)
{
```

```
    var todoItem = await _context.TodoItems.FindAsync(id);

    if (todoItem == null)
    {
        return NotFound();
    }

    return todoItem;
}
```

Заключение

В диссертационной работе проведено исследование базы данных Oracle Database 11g Release 1 (11.1.0.6), языка структурированных запросов Pl SQL, а также использование метода WEB API. Проведен анализ основ и методов разработки мобильных приложений, рассмотрен текущий рынок поставщиков мобильных OS.

Разработанная в рамках диссертационной работы система PAS – является системой оценки персонала, которая позволит оценивать персонал компании АО «Air Astana». Система оценки- это система, которая помогает оценивать результаты работы сотрудников, их потенциал, выявлять таланты, области для будущих улучшений, пробелы в системе управления и т. д. Данная система связана с целями и ценностями компании. Для вышеперечисленных требований был разработан новый инструмент, который позволяет сделать процесс более эффективным и простым в использовании. Система включает в себя KPI: помимо стандартных критерии, также заданные руководителем критерии оценки.

Диссертационная работа с производственной стороны является эффективной и актуальной, система успешно внедрена и используется компанией АО «Air Astana».

В магистерской диссертации были отражены следующие важные тезисы:

- Модель систематизации требований не должна быть ограничена набором конкретных методов и инструментов, а лишь отражает возможные правила их использования с точки зрения профессиональной позиции для каждой заинтересованной стороны.
- Классификация и кодирования требований, имеющих однозначные взаимосвязи на всех уровнях иерархии своих описаний, разрешает проблемы информационного характера, связанные со сложностью восприятия и работы с множеством требований к ИС.
- Набор групп требований определяется в зависимости от концепции разрабатываемой информационной системы и её будущего содержания.

Список литературы

1. Rielau, Serge DB2 9.7: Run Oracle applications on DB2 9.7 for Linux, Unix, and Windows (англ.). Developer Works. IBM (26 May 2010)
2. *McDonald, Connor*. Mastering Oracle PL/SQL: practical solutions. — Springer, 2004. — 605 p. — ISBN 978-1590592175.
3. Jacobson, Daniel; Woods, Dan; Brail, Greg (November 2011). APIs a strategy guide. Sebastopol, Calif: O'Reilly Media. ISBN 978-1-4493-0892-6.
4. *Эндрю Троелсен.* Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание = Pro C# 5.0 and the .NET 4.5 Framework, 6th edition. — М.: «Вильямс», 2013. — 1312 с. — ISBN 978-5-8459-1814-7.
5. *Джозеф Албахари, Бен Албахари.* C# 6.0. Справочник. Полное описание языка = C# 6.0 in a Nutshell: The Definitive Reference. — М.: «Вильямс», 2018. — 1040 с. — ISBN 978-5-8459-2087-4. — ISBN 978-1-491-92706-9.
6. Джозеф Албахари, Бен Албахари. C# 6.0. Справочник. Полное описание языка = C# 6.0 in a Nutshell: The Definitive Reference. — М.: «Вильямс», 2018. — 1040 с. — ISBN 978-5-8459-2087-4. — ISBN 978-1-491-92706-9.
7. Клочков А. К. KPI и мотивация персонала. Полный сборник практических инструментов. — Эксмо, 2010. — 160 с. — ISBN 978-5-699-37901-9.
8. Марк А. Хьюзлид, Дэйв Ульрих, Брайан И. Беккер. Измерение результативности работы HR-департамента. Люди, стратегия и производительность = The HR Scorecard: Linking People, Strategy, and Performance. — М.: «Вильямс», 2007. — С. 304. — ISBN 1-57851-136-4.
9. Клочков А. К. KPI и мотивация персонала. Полный сборник практических инструментов. — Эксмо, 2010. — 160 с. — ISBN 978-5-699-37901-9.
10. Марк А. Хьюзлид, Дэйв Ульрих, Брайан И. Беккер. Измерение результативности работы HR-департамента. Люди, стратегия и производительность = The HR Scorecard: Linking People, Strategy, and Performance. — М.: «Вильямс», 2007. — С. 304. — ISBN 1-57851-136-4.