ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc193651251)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ 5](#_Toc193651252)

[1.1 Общие сведения 5](#_Toc193651253)

[1.2 Классификация информационных систем 7](#_Toc193651254)

[1.3 Обзор методов управления проектами и задачами 8](#_Toc193651255)

[1.4 Роль мобильных приложений в управлении проектами 12](#_Toc193651256)

[1.5 Технологии разработки мобильных приложений 14](#_Toc193651257)

[1.6 Архитектура мобильных приложений 16](#_Toc193651258)

[1.7 Выбор стека технологий 20](#_Toc193651259)

[1.8 Этапы разработки 24](#_Toc193651260)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 26](#_Toc193651261)

[2.1 Проектирование БД 26](#_Toc193651262)

[2.2 Разработка мобильного приложения и описание пользовательского интерфейса 30](#_Toc193651263)

[2.3 Тестирование мобильного приложения 42](#_Toc193651264)

[ЗАКЛЮЧЕНИЕ 45](#_Toc193651265)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 46](#_Toc193651266)

# ВВЕДЕНИЕ

В современном мире, в условиях динамично развивающегося рынка, всё больше компаний нуждаются в оптимизации рабочих процессов и организации контроля выполнения задач. Особенно с этой проблемой сталкиваются рекламные агентства, которым важны оперативное выполнение заказов и одновременная работа с несколькими клиентами. Обычные методы управления в виде электронных таблиц и чатов являются недостаточно эффективными, а большинство современных приложений не адаптированы под специфику рекламных фирм, которым важны гибкость и быстрое взаимодействие между работниками. Именно поэтому возникает необходимость в создании специализированного мобильного приложения, которое позволит организовать продуктивную совместную работу между сотрудниками.

Актуальность проекта, заключается в том, что создание такого мобильного приложения – важный шаг для улучшения работы рекламного агентства. Оно поможет оперативно получать информацию о текущих проектах, координировать действия внутри команды и повысить удовлетворённость клиентов за счёт оперативного выполнения заказов.

Практическая значимость курсовой работы заключается в создании удобного инструмента, который позволит сотрудникам и руководителям эффективно распределять задачи, контролировать их выполнение и обеспечить внутри компании высокую производительность работы.

Цель – разработка мобильного приложения для управления проектами и задачами между сотрудниками и их руководителями. В рамках исследования будут рассмотрены существующие методы управления задачами и проанализированы возможности современных технологий для автоматизации этих процессов.

Для достижения цели необходимо выполнить следующие задачи:

* Провести анализ предметной области и рассмотреть методы управления проектами.
* Придумать концепцию и функционал для мобильного приложения, с учетом потребностей рекламного агентства.
* Проанализировать современные технологии для создания мобильных приложений.
* Разработать архитектуру базы данных приложения.
* Разработать базовую версию приложения с удобным пользовательским интерфейсом.
* Провести тестирование приложения.

Объектом исследования является процесс управления учёта задач, взаимодействие между сотрудниками и обеспечение прозрачности рабочих процессов рекламного агентства.

Предметом исследования является разработка мобильного приложения для учёта проектов и задач сотрудников.

Методы исследования: статистические, теоретические и эмпирические.

Краткая характеристика глав

Курсовая работа состоит из двух глав. В первой главе озвучиваются теоретические аспекты управления проектами и задач, рассматривается роль мобильных приложений в мире и технологии их разработки. Во второй главе описан процесс разработки, тестирования и оценки результатов.

Характеристика курсового проекта включает 50 страниц, 28 литературных источников, 18 картинок и 7 таблиц.

В курсовой работе используются исследования в области организации управления проектами и разработки мобильных приложений. Используются труды таких авторов, как Д. Наттинг, М. Фаулер и Э. Бурнет.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

# Общие сведения

Рекламное агентство специализируется на оказании услуг интернет-рекламы, работая с заказами различного масштаба и уровня сложности. Чтобы повысить эффективность работы и улучшить координацию внутри команды, руководство агентства приняло решение о разработке информационной системы. В качестве решения было выбрано мобильное приложение для устройств на базе операционной системы Android, которое призвано упростить выполнение клиентских заказов за счёт организации контроля над проектами и задачами.

Разрабатываемое мобильное приложение предназначено для управления проектами и задачами в рекламном агентстве. Его основная цель — обеспечить эффективное взаимодействие между сотрудниками, упростить координацию внутри команды и повысить общую производительность. Приложение должно предоставлять руководителю команды возможность создавать, редактировать и управлять проектами и задачами, что позволит оптимизировать рабочие процессы и улучшить качество выполнения заказов.

Основные функции:

* Создание проектов: возможность добавлять новые проекты и указывать их название.
* Создание задач: возможность добавлять задачи в рамках проекта, указывать их название, описание, назначать исполнителей.
* Статусы задач: поддержка различных статусов задач (например, «Выполняется», «Завершена», «Отменена»).
* Гибкость: возможность быстро изменять параметры задач и проектов в зависимости от изменений в требованиях клиентов.
* Экран «Мои задачи»: возможность каждому сотруднику просматривать задачи, в которых он является исполнителем.

Приложение должно быть удобным для использования на мобильных устройствах:

* Адаптивный интерфейс: поддержка различных размеров экранов и ориентаций (портретной и ландшафтной).
* Интуитивность: простой и понятный интерфейс, не требующий длительного обучения.
* Быстродействие: минимальное время загрузки и отклика приложения.

Для защиты данных приложение должно обеспечивать безопасность:

* Разграничение прав доступа: разные уровни доступа для сотрудников, руководителей и администраторов.
* Экран администратора: личный аккаунт администратора для добавления и удаления новых сотрудников.
* Шифрование данных: защита данных при передаче и хранении.

Перечисленные требования к функциональности мобильного приложения направлены на создание удобного и эффективного инструмента для управления проектами и задачами в рекламных агентствах. Приложение должно быть гибким, безопасным и простым в использовании, чтобы соответствовать потребностям современных компаний.

Основанием для разработки технического задания является задание к курсовой работе по теме «Разработка системы для управления проектами и задачами для сотрудников рекламного агентства».

Данная курсовая работа помогает автоматизировать такие процессы, как:

* Управление проектами и задачами
* Управление пользователями и правами доступа
* Работу в реальном времени

# Классификация информационных систем

**Информационная система (ИС)** – это комплекс взаимосвязанных элементов, работающих совместно для сбора, обработки, хранения и передачи данных. Сегодня ИС чаще всего подразумевает использование персональных компьютеров как основной технической платформы для работы с информацией. Тем не менее, стоит помнить, что техническая реализация ИС теряет смысл без человеческого фактора: именно человек создаёт и интерпретирует информацию, которую генерирует система. Важно отличать компьютеры от информационных систем: хотя ПК с установленным специализированным ПО служат основой для функционирования ИС, сами системы невозможны без операторов, управляющих ими через компьютеры и телекоммуникационные сети.

ИС обладает ключевыми характеристиками: она может быть проанализирована, спроектирована и управляема согласно общим принципам системного подхода; она развивается и изменяется во времени; требует системного подхода при разработке; результатом её работы является информация, используемая для принятия решений. Процесс функционирования ИС можно представить схематически: поступление данных извне или изнутри, их обработка и преобразование в удобный вид, выдача результатов пользователям или передача другим системам, а также обратная связь, когда обработанная информация возвращается для корректировки исходных данных.

ИС классифицируют по разным признакам. В зависимости от характера использования информации существуют справочно-информационные системы, которые не требуют сложной обработки данных, и системы обработки данных, решающие специфичные задачи. По функциям ИС подразделяются на производственные, маркетинговые, финансовые, кадровые и прочие, часто поддерживающие дополнительные процессы. По объектам управления выделяются системы управления ресурсами, предприятиями, технологическими процессами, базами данных, автоматизированные системы управления и проектирования.

По уровню автоматизации ИС разделяют на полностью автоматизированные (автоматические) и частично автоматизированные (автоматизированные). С точки зрения архитектуры, ИС бывают локальные (настольные), где все элементы находятся на одном устройстве, и распределенные, где компоненты размещены на нескольких машинах.

Основная работа ИС заключается в обработке двух типов информационных потоков: поступающей информации и запросов на предоставление текущих данных. Главной задачей ИС является обслуживание пользователей, поэтому система должна оперативно и полно отвечать на запросы. Это достигается благодаря стандартным процедурам поиска и структурированности данных.

# Обзор методов управления проектами и задачами

Главные аспекты быстрого и качественного достижения целей — это грамотное управление проектами и рациональная структуризация задач. Для решения данной проблемы были разработаны разные эффективные методологии, каждая из которых имеет свои преимущества перед остальными.

Рассмотрим основные методы:

* **Waterfall** (Каскадная модель или «водопад»)

Waterfall - одна из старейших методологий. Она предусматривает последовательное выполнение этапов: планирование, проектирование, реализация, тестирование и завершение. Каждый этап должен завершаться до перехода к следующему. Это обеспечивает быструю разработку и заранее определённые сроки и стоимость.

Преимущества: чёткая структура разработки, запланированная заранее. Подходит для проектов с фиксированными требованиями.

Недостатки: низкая гибкость и невозможность вносить изменения во время выполнения этапов. Waterfall не подходит для проектов с динамически меняющимися требованиями.

Этот подход не годится для работы рекламного агентства из-за недостаточной гибкости и невозможности адаптироваться под требования клиентов.

* **Agile** (Гибкая методология)

Agile - один из самых популярных подходов к управлению проектами. Он ориентирован на гибкость и быструю адаптацию к изменениям. Проект делится на небольшие этапы, после каждого из которых клиент видит промежуточный результат и может оценить его соответствие ожиданиям.

Преимущества: высокая гибкость разработки и постоянный контакт с клиентом улучшают качество работы и позволяют корректировать ход проекта.

Недостатки: требуются высокая вовлечённость команды и клиента, регулярные встречи и обсуждения. Отсутствие чётко сформулированных конечных результатов затрудняет оценку стоимости и трудозатрат, особенно для больших проектов с жёсткими сроками.

Тем не менее Agile идеально подходит для рекламных агентств, где клиенты часто меняют свои требования.

* **Scrum**

Scrum - самая популярная система выполнения задач, основанная на принципах Agile. Работа ведётся короткими итерациями (спринтами) длительностью от одной до четырёх недель. Команда выбирает задачи из приоритизированного бэклога продукта (динамического списка всех задач, идей и требований проекта). После выполнения задач клиент даёт обратную связь, проводится анализ процесса, вносятся улучшения, и цикл повторяется.

Преимущества: высокая гибкость и прозрачность процесса обеспечивают быструю адаптацию к изменениям благодаря постоянной обратной связи с клиентом. Регулярные улучшения повышают эффективность команды.

Недостатки: требуют высокой дисциплины и самоорганизации, что не подходит для команд с низкой вовлечённостью.

Scrum популярен в работе рекламных агентств, несмотря на свою сложность. Высокая организованность и способность адаптироваться к новым идеям помогают успешно управлять творческими проектами.

* **Kanban**

Kanban - это методология, основанная на визуализации рабочего процесса. Задачи представляются в виде карточек на доске, что позволяет отслеживать их статус («В работе», «Завершено» и т.д.).

Преимущества: простота и высокая гибкость. Возможность легко распределять задачи среди сотрудников.

Недостатки: меньшая структурированность по сравнению со Scrum и необходимость постоянного контроля выполнения работ командой.

Kanban хорошо подходит для рекламных агентств, позволяя эффективно контролировать выполнение задач и оперативно реагировать на изменения. Простая визуализация облегчает организацию работы команды.

* **Hybrid** (Гибридный подход)

Hybrid сочетает элементы Waterfall и Agile. Планирование и бюджетирование выполняются по Waterfall, а реализация и тестирование - по Agile. Такой подход сохраняет структурированность, обеспечивая гибкость работы.

Преимущества: сочетание предсказуемости проекта и способности адаптироваться к новым условиям и задачам. Подходит для проектов с частично меняющимися требованиями.

Недостатки: требует тщательного планирования, как и Waterfall.

Гибридный подход стоит применять для крупных проектов, где нужен строгий контроль и гибкость для отдельных задач.

* **Lean** (Бережливое производство)

Lean - методология рационального подхода, позволяющая сосредоточиться на снижении возможных потерь и оптимизации основных процессов. Она работает в условиях ограниченных ресурсов, разбивая проект на подзадачи.

Преимущества: сокращение затрат и постоянное улучшение процессов. Меньше требований к срокам и конфигурации подзадач по сравнению со Scrum.

Недостатки: требует значительной перестройки организационных процессов, что осложняет работу над креативными проектами.

Lean применяется в рекламных агентствах, стремящихся снизить затраты и оптимизировать рабочие процессы.

* **PRINCE2**

PRINCE2 - подход к управлению проектами, основанный на распределении ролей и ответственности между сотрудниками.

Преимущества: чёткое структурирование задач и контроль их выполнения. Широко используется в крупных компаниях.

Недостатки: сложность из-за высокой бюрократичности и низкая гибкость. Не применим в небольших проектах.

Данный подход редко используется в рекламных агентствах из-за сложности и сильной зависимости от бюрократических процедур.

Исходя из рассмотренных методологий, наиболее подходящими для работы рекламных агентств являются Agile, Scrum и Kanban. Эти подходы обладают необходимой гибкостью и эффективностью, что позволяет быстро адаптироваться к изменениям и оперативно реагировать на запросы клиентов. Правильный выбор методологии поможет ускорить и рационализировать процесс разработки.

# Роль мобильных приложений в управлении проектами

Мобильные приложения стали неотъемлемой частью жизни каждого человека. Они играют ключевую роль в управлении проектами, позволяя упростить отслеживание и координацию задач.

Рассмотрим преимущества их использования:

* Основным преимуществом по сравнению с остальными технологическими источниками коммуникации является мобильность. Сотрудники могут использовать мобильные приложения, находясь в любой точке мира, где есть доступ к интернету. Это особенно важно для тех, кто часто находится в разъездах.
* Все сотрудники имеют доступ к актуальным данным и задачам проекта в режиме реального времени, что позволяет оставаться в курсе последних изменений. Большим преимуществом является возможность просматривать информацию о статусах задач, графиках и других важных данных прямо с мобильного телефона.
* Мобильные приложения дают пользователю возможность получать уведомления о важных событиях или изменениях в задачах. Уведомления помогают сотрудникам оставаться в курсе изменений и не пропускать ключевые этапы проекта.
* Использование мобильных приложений помогает упростить процесс коммуникации между сотрудниками, позволяя обмениваться важной информацией, комментариями, файлами и т.д. Возможность быстрого обсуждения различных проблем очень важна для эффективной работы команды.
* Мобильные приложения повышают продуктивность сотрудников, необходимую им для выполнения задач и проверки статуса проекта в дороге или во время ожидания деловой встречи. Также они позволяют взаимодействовать с другими инструментами, например, календарями, почтой, облачными хранилищами и т.д.
* Руководители проектов могут легко отслеживать прогресс выполнения задач, анализировать данные и вносить изменения в текущий вектор разработки проекта.
* Современные мобильные приложения обеспечивают пользователям высокий уровень защиты данных, используя шифрование и двухфакторную аутентификацию. Это важно при работе с конфиденциальной информацией компании.

В настоящее время существует большое количество приложений, помогающих управлять проектами различной сложности. К таким приложениям относятся:

* Trello - удобное управление задачами с помощью досок и карточек.
* Asana - планирование задач, отслеживание прогресса и коммуникация в команде.
* Monday - гибкая платформа для управления проектами и рабочими процессами.
* Microsoft Teams - коммуникация и совместная работа в одном приложении.
* Jira - управление проектами, особенно популярное в IT-сфере.

Мобильные приложения для управления проектами и задачами остаются незаменимыми инструментами для организации продуктивной совместной работы между сотрудниками. Их использование помогает участникам команды следить за обновлениями информации и статусами выполнения задач, а компаниям — быстро адаптироваться к динамично меняющемуся рынку и условиям бизнеса.

# Технологии разработки мобильных приложений

При создании мобильных приложений важно выбрать подходящие технологии и платформы, чтобы обеспечить высокую производительность и совместимость с различными устройствами. Рассмотрим ключевые отличия, плюсы и минусы различных методов разработки мобильных приложений:

* **Разработка нативных приложений**

Создание нативных приложений подразумевает разработку специализированного ПО для определенной платформы - iOS или Android. Такой подход гарантирует максимальную производительность, полное использование возможностей устройства и глубокую интеграцию с экосистемой платформы. Программирование ведется с применением официальных языков и инструментов, предоставленных производителями платформ, что позволяет создать продукт высокого качества, соответствующий потребностям пользователей и возможностям устройств.

Для **iOS**: Основной язык программирования - Swift, представленный Apple в 2014 году взамен устаревшего Objective-C. Swift отличается современностью, безопасностью и удобством для разработчиков, сочетая элементы функционального и объектно-ориентированного программирования. Ключевым инструментом разработки является Xcode, включающий Interface Builder для визуального проектирования интерфейса и фреймворк SwiftUI для декларативного описания UI. SwiftUI облегчает создание анимации и адаптивных интерфейсов, обеспечивая автоматическое обновление UI при изменениях данных. Для работы с локальной базой данных используется Core Data, а для сетевого взаимодействия - URLSession.

Для **Android**: Основным языком программирования здесь выступает Kotlin, официально поддерживаемый Google с 2017 года. Kotlin, пришедший на смену Java, обладает преимуществами безопасности и удобства, поддерживая функциональное программирование. Основной инструмент разработки - Android Studio, с поддержкой редактора кода, эмулятора и инструментов тестирования. Современная разработка Android-приложений часто ведётся с использованием Jetpack Compose, обеспечивающего декларативный подход к созданию интерфейсов. Для работы с базами данных используется библиотека Room, а для сетевой коммуникации - Retrofit.

Нативные приложения обеспечивают высокую производительность, глубокий доступ к API и отличную интеграцию с экосистемами платформ. Однако они требуют значительных затрат на разработку и поддержку, особенно если нужно поддерживать несколько платформ. Кроме того, для разработки под iOS необходимы устройства Apple (Mac).

* **Разработка гибридных приложений**

Гибридные приложения объединяют веб-технологии и нативные компоненты, позволяя использовать единый код для разных платформ (iOS, Android и веб). Создаются они с использованием HTML, CSS и JavaScript, после чего запускаются в нативном контейнере, что даёт возможность задействовать одну кодовую базу для множества платформ.

Популярные фреймворки:

**React Native** (от Facebook) - использует JavaScript и библиотеку React для создания приложений для iOS и Android с высоким уровнем производительности.

**Flutter** (от Google) - применяет язык Dart и собственный движок рендеринга Skia для создания красивых и производительных интерфейсов.

**Ionic** - основан на веб-технологиях и запускается в нативном контейнере, предоставляя возможность использовать один код для iOS, Android и веба.

Основные части гибридного приложения – это веб-компоненты (интерфейсы и логика) и нативный контейнер, взаимодействующий с устройством через плагины вроде Cordova или Capacitor. Такие приложения позволяют быстро создавать прототипы и реализовывать новые функции, обеспечивая кроссплатформенность и экономию времени и ресурсов.

Однако гибридные приложения уступают нативным в плане производительности, особенно при работе с графикой и сложными вычислениями. Также они могут иметь ограничения в доступе к некоторым функциям устройства и зависеть от обновлений используемых фреймворков.

Для реализации мобильного приложения с высокими требованиями к производительности лучше всего подойдут нативные решения, поскольку они обеспечивают наилучшую интеграцию с платформой и максимальный доступ к её функционалу.

# Архитектура мобильных приложений

При создании мобильного приложения важно выбрать подходящие архитектурные принципы и спроектировать базу данных для эффективного хранения и обработки данных. Архитектурные паттерны помогают структурировать код, распределяя ответственность между компонентами и облегчая поддержку приложения. Проектирование базы данных должно учитывать все необходимые сущности и их взаимосвязи, чтобы обеспечить удобное управление данными и масштабируемость приложения.

**Архитектурные паттерны**

**MVC (Model-View-Controller)**

MVC - один из наиболее популярных архитектурных шаблонов разработки приложений. Он делит приложение на три основные части: Модель (Model), Представление (View) и Контроллер (Controller).

Модель (Model) - отвечает за данные и бизнес-логику. Она взаимодействует с базой данных, обрабатывает запросы и передает результаты другим компонентам. Модель независима от View и Controller, что упрощает тестирование и повторное использование кода.

Представление (View) - выводит данные на экран пользователя. Оно получает информацию от контроллера и отображает ее в удобочитаемом виде. View не содержит бизнес-логику и не взаимодействует непосредственно с моделью, а работает через контроллер.

Контроллер (Controller) - связующее звено между Моделью и Представлением. Обрабатывает действия пользователя (нажатия кнопок, выбор элементов), запрашивает данные у модели и передает их представлению.

**Преимущества MVC:**

* Разделение обязанностей: каждая часть отвечает за свою функциональность.
* Упрощенное обслуживание и тестирование: благодаря независимости компонентов, изменения одного модуля не влияют на остальные.
* Повторное использование кода: MVC способствует созданию многократно используемого кода в Моделях и Контроллерах.
* Гибкость: легко добавлять или изменять функциональные возможности без нарушения структуры приложения.

**Недостатки MVC:**

* Сложность контроллеров: в крупных приложениях контроллеры могут стать перегруженными логикой.
* Тесная связь между контроллером и представлением: иногда контроллер и представление оказываются сильно взаимосвязаны, что затрудняет тестирование и сопровождение.
* Ограниченность для сложных приложений: в случае сложной логики интерфейса MVC может оказаться менее эффективным.

**MVVM (Model-View-ViewModel)**

MVVM - современная архитектура, часто применяемая в мобильной и веб-разработке. Она делит приложение на три компонента: Модель, Представление и Модель Представления.

Модель (Model) - аналогична MVC: хранит данные и бизнес-логику, взаимодействуя с базой данных.

Представление (View) - визуализирует данные, получаемые от ViewModel.

Модель Представления (ViewModel) - играет роль посредника между Моделью и Представлением. В отличие от контроллера в MVC, ViewModel не управляет представлением напрямую, используя механизм двусторонней привязки данных.

**Преимущества MVVM:**

* Четкая структура: Модель отвечает за данные, Представление - за интерфейс, а ViewModel - за взаимодействие между ними.
* Простота тестирования: ViewModel независима от Представления, что облегчает тестирование.
* Двусторонняя привязка данных: автоматическая синхронизация данных между Представлением и ViewModel.
* Многократное использование: ViewModel может применяться в разных частях приложения или проекте.
* Гибкость: MVVM позволяет добавлять новые функции без нарушения работы других частей системы.

**Недостатки MVVM:**

* Сложность внедрения: MVVM сложнее реализовать, особенно при работе с механизмом привязки данных.
* Зависимость от инструментов: требуются дополнительные библиотеки для привязки данных (например, LiveData, RxJava).
* Излишний функционал для малых проектов: для простых приложений MVVM может показаться избыточным.

**Clean Architecture (Чистая архитектура)**

Clean Architecture делит систему на слои, минимизируя зависимость бизнес-логики от внешних факторов. Это улучшает тестируемость, масштабируемость и гибкость приложений.

**Принципы Clean Architecture:**

* Независимость от фреймворков: логика приложения отделяется от конкретных библиотек и платформ.
* Независимость от интерфейса: изменение интерфейса не влияет на бизнес-логику.
* Независимость от базы данных: бизнес-логика не зависит от конкретного типа хранилища данных.
* Тестируемость: чистое разделение слоев упрощает написание тестов.

**Слои Clean Architecture:**

* Domain Layer (Слой доменной логики) - содержит сущности и бизнес-правила, независимые от внешнего окружения.
* Application Layer (Слой приложения) - описывает сценарии взаимодействия пользователей с системой (use cases).
* Infrastructure Layer (Слой инфраструктуры) - отвечает за взаимодействие с внешними сервисами, такими как базы данных и сети.
* Presentation Layer (Слой представления) - включает пользовательские интерфейсы и модели представлений.

**Преимущества Clean Architecture:**

* Независимость от технологий: легкость адаптации к новым технологиям.
* Удобство тестирования: изолированная бизнес-логика упрощает тестирование.
* Масштабируемость: легко расширяемая и поддерживаемая система.

**Недостатки Clean Architecture:**

* Высокая сложность реализации: требует значительного усилия для правильной организации.
* Избыток функционала для маленьких проектов: избыточна для простейших приложений.

В качестве архитектуры для приложения по управлению проектами, был выбран подход MVVM, поскольку в нем сочетается гибкость и простота реализации, что крайне необходимо для динамического интерфейса. MVVM позволяет управлять состоянием приложения, обеспечивая синхронизацию данных между интерфейсом и бизнес-логикой.

# Выбор стека технологий

Перед началом разработки любого программного обеспечения крайне важно правильно подобрать технологии, поскольку именно от этого выбора будет зависеть успешность всего проекта, его способность к расширению и итоговое качество продукта. В процессе создания мобильного приложения для учета проектов и задач было решено использовать нативный формат, предполагающий применение технологий, созданных специально для платформы Android. Основными инструментами стали Android Studio, Jetpack Compose и Firebase, обеспечивающие высокую производительность, простоту разработки и надежную защиту данных.

* **Android Studio**

Android Studio представляет собой официальную среду разработки для создания приложений под Android, основанную на платформе IntelliJ IDEA и разработанную компанией Google. Эта среда предоставляет полный набор необходимых инструментов для проектирования, тестирования и отладки приложений. Одно из главных достоинств Android Studio заключается в поддержке современных языков программирования, таких как Java и Kotlin. В рамках данного проекта предпочтение было отдано языку Kotlin, который отличается большей современностью, лаконичностью и безопасностью по сравнению с Java.

Android Studio оснащена мощным визуальным редактором макетов, позволяющим разрабатывать пользовательский интерфейс с использованием XML или посредством перетаскивания элементов (drag-and-drop). Этот инструмент существенно облегчает процесс создания интерфейсов. Более того, интегрированный эмулятор устройств позволяет протестировать приложение на множестве виртуальных устройств с различными версиями операционной системы Android, разрешениями дисплеев и техническими характеристиками.

Еще одна важная особенность Android Studio состоит в интеграции с системой сборки Gradle, что значительно упрощает управление зависимостями и настройку процесса сборки проекта. Кроме того, в Android Studio предусмотрены инструменты для отладки и анализа производительности, помогающие находить и исправлять ошибки, отслеживать производительность приложения и обнаруживать утечки памяти.

Одним из ключевых преимуществ Android Studio является тесная интеграция с платформой Firebase, что упрощает внедрение сервисов вроде Firestore и Authentication, критически важных для данного проекта. В общем и целом, выбор Android Studio обусловлен официальной поддержкой со стороны Google, обширной экосистемой и комфортом разработки, включающим встроенные шаблоны проектов, подсказки и автозавершение кода.

* **Jetpack Compose**

Для создания пользовательского интерфейса была выбрана библиотека Jetpack Compose, отличающаяся современным подходом. В отличие от традиционных методов, где интерфейс создается с использованием XML и метода findViewById, Jetpack Compose основывается на декларативном подходе. Разработчику достаточно описать внешний вид интерфейса в зависимости от текущего состояния, а Jetpack Compose самостоятельно обновит интерфейс при изменениях данных. Это делает код более понятным и удобным для поддержки.

Одной из важнейших характеристик Jetpack Compose является модульность. Интерфейсы разбиваются на мелкие, многократно используемые компоненты, которые можно объединять для формирования сложных экранов. Например, текстовые поля, кнопки и контейнеры (Column, Row и Box) легко комбинируются для создания страниц приложения.

Управление состоянием в Jetpack Compose организовано весьма эффективно. Состояние, определяющее внешний вид интерфейса, можно представить с помощью mutableStateOf. Для сохранения состояния при повторной композиции (например, при смене ориентации экрана) применяется функция remember, а для сохранения состояния даже после перезапуска активности - rememberSaveable. Рекомендуется поднимать состояние на верхний уровень иерархии компонентов, чтобы повысить их переиспользование и облегчить тестирование.

Jetpack Compose также предоставляет удобные средства для создания анимаций, такие как animateColorAsState для плавного изменения цветов и animateDpAsState для анимирования изменений размеров. Это способствует созданию современных и отзывчивых интерфейсов.

Интеграция Jetpack Compose с архитектурой MVVM упрощает разработку. Состояние можно хранить в ViewModel и передавать компонентам через LiveData или StateFlow. Такое четкое разделение логики и интерфейса облегчает тестирование и сопровождение приложения.

Преимущества Jetpack Compose включают упрощение кода, многократное использование компонентов, автоматическое обновление интерфейса при изменении состояний, поддержку современного дизайна Material Design и высокую производительность. Благодаря этим характеристикам Jetpack Compose стал стандартом для создания интерфейсов в приложениях для Android.

* **Firebase**

Для реализации функций бэкенда была выбрана платформа Firebase, предлагаемая Google. Платформа включает множество сервисов, облегчающих создание полноценных приложений без необходимости разворачивать собственный сервер. В данном проекте использовались два ключевых сервиса Firebase: Firestore и Authentication.

**Firestore** – это облачное хранилище данных формата NoSQL, предназначенное для хранения и синхронизации данных в режиме реального времени между клиентом и сервером. Данные в Firestore организованы в коллекции и документы, что обеспечивает гибкость и масштабируемость. Особенностью Firestore является поддержка работы в реальном времени: данные автоматически синхронизируются между всеми подключенными устройствами. Это особенно ценно для приложений, требующих мгновенного обновления информации, таких как мессенджеры или онлайн-игры.

Firestore также поддерживает работу в автономном режиме, позволяя пользователям взаимодействовать с приложением даже при отсутствии подключения к Интернету. После восстановления соединения данные синхронизируются с сервером. Firestore предоставляет гибкую систему правил безопасности, управляющих доступом к данным на уровне пользователей.

**Firebase Authentication** – это сервис для аутентификации пользователей, поддерживающий разнообразные методы входа, такие как электронная почта и пароль, учетные записи социальных сетей (Google, Facebook, Twitter) и телефон. Firebase Authentication легко интегрируется с Firestore, обеспечивая контроль доступа к данным на основании прав пользователей. Безопасность обеспечивается применением стандартов, таких как OAuth 2.0 и OpenID Connect. Firebase Console предоставляет удобный интерфейс для управления пользователями, включая сброс паролей и блокировку аккаунтов.

Выбор Android Studio, Jetpack Compose и Firebase позволит создать современное, масштабируемое и защищённое мобильное приложение. Android Studio предоставляет весь необходимый инструментарий для разработки, Jetpack Compose упрощает создание пользовательского интерфейса, а Firebase обеспечивает функциональность бэкенда, включая хранение данных и аутентификацию пользователей. Эти технологии не только ускоряют процесс разработки, но и гарантируют высокое качество конечного продукта, делая их оптимальным решением для создания мобильных приложений.

# Этапы разработки

Планирование этапов разработки является главным процессом в жизненном цикле ПО. Чёткое разделение разработки на этапы позволяет минимизировать риск ошибок и обеспечить выполнение всех требований в установленные сроки.

Этапы жизненного цикла программного обеспечения включают:

* Анализ требований: определяются требования в виде технического задания разработки ПО.
* Определение спецификаций: выбор технологий разработки.
* Разработка ПО: реализация поставленных требований.
* Тестирование ПО: проверка программного обеспечения на наличие ошибок и соответствие всех указанных требований.

Во время анализа требований определяются потребности заказчика и формулируются требования к приложению. Для рекламного агентства требованиями являются:

* Управление проектами и задачами.
* Создание, редактирование и назначения задач.
* Контроль статусов задач.
* Разграничение прав доступа для сотрудников и руководителей.
* Удобный и интуитивно понятный интерфейс.
* Интеграция с облачной базой данных для хранения и синхронизации данных.

На основе требований определяются технические спецификации приложения. Для разработки данного мобильного приложения были выбраны среда разработки Android Studio и официально поддерживаемый Google современный язык программирования Kotlin. Для реализации отзывчивого пользовательского интерфейса была выбрана библиотека Jetpack Compose. В качестве базы данных для хранения и синхронизации данных в реальном времени была выбрана Firestore и Firebase Authentication позволяющая регистрировать новых пользователей.

На этапе разработки ПО создаётся архитектура приложения и реализуются все необходимые функции.

План разработки:

* Проектирование архитектуры MVVM для обеспечения гибкости и масштабируемости приложения.
* Создание современного пользовательского интерфейса с использование Jetpack Compose.
* Реализация логики работы приложения.
* Интеграция с Firebase. Подключение Firestore для хранения данных и Firebase Authentication для управления пользователями.

Тестирование предназначено для проверки приложения на наличие возможных ошибок и соответствие готового приложения поставленным требованиям. Для этого будут проведены тест-кейсы и unit-тесты, для проверки отдельных модулей и компонентов приложения.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

# Проектирование БД

Проектирование базы данных — важный этап разработки мобильных приложений. Правильная организация структуры базы данных влияет на производительность, масштабируемость и удобство использования приложения. В данном приложении используется облачная база данных NoSQL Firestore для хранения данных. Несмотря на отсутствие жестких требований к соблюдению реляционных моделей в Firestore, при проектировании были учтены основные принципы нормализации для предотвращения дублирования данных и поддержания целостности информации.

База данных включает три ключевые коллекции (аналогично таблицам): **Users**, **Projects** и **Tasks**. Каждая коллекция содержит документы, соответствующие записям с набором определенных полей. Далее приведено описание структуры каждой коллекции.

* **Коллекция Users**

Коллекция Users предназначена для хранения информации о пользователях приложения. Каждый пользователь представлен отдельным документом с уникальными идентификаторами и следующими полями:

**Таблица 1**

**Структура таблицы Users**

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | String | Уникальный идентификатор пользователя. Генерируется автоматически. |
| firstName | String | Имя пользователя. Используется для отображения в интерфейсе. |
| lastName | String | Фамилия пользователя. Используется для отображения в интерфейсе. |
| email | String | Электронная почта пользователя. Используется для входа в систему. |
| role | String | Роль пользователя (например, "Сотрудник", "Руководитель"). |
| password | String | Пароль пользователя. |

Продолжение таблицы 1

* **Коллекция Projects**

Коллекция Projects хранит данные о проектах, выполняемых пользователями. Каждый проект представлен документом с уникальными идентификаторами и такими полями:

**Таблица 2**

**Структура таблицы Projects**

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | String | Уникальный идентификатор проекта. Генерируется автоматически. |
| name | String | Название проекта. Используется для отображения в интерфейсе. |

Продолжение таблицы 2

* **Коллекция Tasks**

Коллекция Tasks содержит информацию о задачах, привязанных к проектам. Каждая задача представлена документом с уникальными идентификаторами и содержит такие поля:

**Таблица 3**

**Структура таблицы Tasks**

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | String | Уникальный идентификатор задачи. Генерируется автоматически. |
| title | String | Название задачи. Используется для отображения в интерфейсе. |
| description | String | Подробное описание задачи. Может содержать дополнительные сведения. |
| status | String | Текущий статус задачи (например, "Выполняется", "Завершена"). |
| assignedUserIds | List<String> | Список идентификаторов пользователей, назначенных на выполнение задачи. |

Продолжение таблицы 3

Связи между коллекциями реализуются посредством идентификаторов документов. Например:

* Каждая задача (Task) связывается с проектом (Project) через поле projectId.
* Назначение задачи одному или нескольким пользователям происходит через поле assignedUserIds, которое содержит идентификаторы соответствующих пользователей (User).

Хотя Firestore является базой данных NoSQL, принципы нормализации применяются для обеспечения целостности данных и минимизации избыточности:

**Первая нормальная форма (1NF)**

* У каждой коллекции есть уникальный идентификатор (id).
* Все поля имеют атомарный характер (нераздельный).

**Вторая нормальная форма (2NF)**

* Все неключевые поля зависят исключительно от основного ключа (id).

**Третья нормальная форма (3NF)**

* Нет транзитивных зависимостей среди неключевых полей.

Для улучшения производительности и удобства использования Firestore приняты во внимание следующие аспекты:

**Денормализация**: Некоторая информация может дублироваться для уменьшения количества запросов к базе данных. Например, имя пользователя может сохраняться непосредственно в задаче для мгновенного отображения без необходимости обращения к коллекции Users.

**Иерархическое строение**: Задачи могут размещаться в подколлекциях внутри документа проекта, что облегчает извлечение всех задач, связанных с определенным проектом.

Разработанная структура базы данных включает три ключевые коллекции: Users, Projects и Tasks. Каждая коллекция обладает уникальными идентификаторами и необходимыми полями для корректной работы приложения. Применение принципов нормализации гарантирует целостность данных, а использование особенностей Firestore повышает производительность. Такая структура позволяет эффективно управлять проектами и задачами в рамках рекламного агентства, обеспечивая удобство эксплуатации и масштабируемость приложения.

# Разработка мобильного приложения и описание пользовательского интерфейса

Разработка мобильного приложения для управления проектами и задачами в рекламном агентстве состояла из нескольких этапов: создание интерфейса, внедрение функциональности, интеграция с базой данных и проведение тестирования.

Интерфейс приложения разрабатывался с акцентом на удобство, простоту и интуитивность. В качестве основной цветовой палитры приложения были выбраны белый и различные оттенки синего.

**Таблица 4**

**Цвета элементов приложения**

|  |  |
| --- | --- |
| Цвет по Hex-коду | Использование в приложении |
| #FF000000 | Цвет текста. |
| #FFFFFFFF | Цвет фона. |
| #057DCDFF | Цвет кнопок входа и выхода из аккаунта пользователя. |
| #43B0F1FF | Цвет текста кнопок добавления проекта и задач. |
| #63B4FFFF | Фон кнопок добавления проекта и задач. |

Приложение включает следующие основные экраны: экран авторизации, экран администратора, экран с задачами, экран «Проекты», экран «Мои задачи» и экран «Настройки».

**Экран авторизации**

При первом запуске приложения перед пользователем открывается экран входа в систему. Этот экран содержит поля для ввода адреса электронной почты («Email») и пароля, а также кнопку «Войти». Если пользователь вводит правильные учетные данные, нажимает синюю кнопку «Войти», происходит успешная авторизация, и пользователь переходит на основной экран с проектами. Если данные введены неправильно, процесс авторизации не пройдет, и пользователь остается на экране входа.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 - Экран авторизации

**Экран администратора**

Для контроля над учетными записями сотрудников рекламного агентства предусмотрен специальный экран администратора. Здесь администратор может добавлять и удалять сотрудников и руководителей компании. На экране представлен список всех зарегистрированных пользователей с указанием имени, фамилии, адреса электронной почты, пароля и занимаемой должности.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 - Экран администратора

Чтобы зарегистрировать нового пользователя, администратору необходимо ввести требуемые данные, указать роль пользователя и нажать кнопку «Зарегистрировать». Чтобы удалить учетную запись сотрудника, достаточно нажать красную кнопку «Удалить» рядом с выбранным сотрудником.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 - Экран регистрации пользователя

**Экран «Проекты»**

После успешной авторизации пользователь попадает на главный экран с проектами. В нижней части экрана расположена панель навигации между различными экранами приложения, включая разделы «Проекты», «Мои задачи» и «Настройки».

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Нижняя панель навигации

Интерфейс приложения зависит от роли пользователя в компании. Если пользователь имеет роль «Руководитель», на экране «Проекты» будет доступен список текущих проектов и возможность добавить новый проект через кнопку «Добавить проект».

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Экран «Проекты» для роли «Руководитель»

Если пользователь является обычным сотрудником («Сотрудник»), кнопка «Добавить проект» отсутствует, поскольку сотрудники не обладают правами руководства.

Изображение выглядит как текст, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 - Экран «Проекты» для роли «Сотрудник»

**Экран с задачами**

Нажатие на определенный проект открывает список связанных с ним задач, а также кнопку возврата на главную страницу и название текущего проекта. Каждая задача включает название, описание, статус выполнения (с цветовой кодировкой для разных статусов), а также назначенных исполнителей.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 - Экран с задачами проекта для роли «Руководитель»

Руководителям доступны дополнительные функции управления проектом и задачами. Они могут изменять проект и задачи, а также добавлять новые задачи. Для добавления новой задачи руководителю необходимо заполнить форму с названием и описанием задачи, назначить исполнителей и нажать кнопку «Создать задачу». Созданная задача сразу появляется в общем списке.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 - Форма для создания задачи

Чтобы отредактировать существующую задачу, нужно нажать на иконку с тремя точками возле задачи, чтобы открыть меню редактирования. Здесь можно изменить название, описание, статус задачи и назначить новых исполнителей. После внесения изменений следует нажать кнопку «Сохранить».

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 - Окно редактирования задачи

Удаление задачи осуществляется через нажатие на значок корзины и подтверждение операции красной кнопкой «Удалить».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 - Окно удаления задачи

Если руководитель хочет отредактировать сам проект или удалить его, он может воспользоваться опциями, доступными через нажатие трех точек в верхнем левом углу экрана.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 - Меню редактирования проекта

Выбрав вариант «Редактировать проект» руководителю предлагается изменить название проекта.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 - Окно редактирования проекта

Если руководитель решит удалить выбранный проект, то перед ним появится окно с подтверждением удаления.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 - Окно удаления проекта

Сотрудники, заходящие на этот экран, не имеют возможности вносить какие-либо изменения в проекты и задачи.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 - Экран с задачами проекта для роли «Сотрудник»

**Экран «Мои задачи»**

На экране «Мои задачи» отображаются все задачи, где авторизованный пользователь назначен в качестве исполнителя.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 - Экран «Мои задачи» для роли «Сотрудник»

Руководители не видят никаких задач на данном экране, так как они не могут быть включены в списки исполнителей.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 16 - Экран «Мои задачи» для роли «Руководитель»

**Экран «Настройки»**

Этот экран предоставляет информацию о пользователе: имя, фамилию и должность. Также здесь размещена кнопка «Выйти», позволяющая завершить текущую сессию и войти под другой учетной записью.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 - Экран «Настройки»

# Тестирование мобильного приложения

Чтобы протестировать приложение на наличие возможных ошибок и недочётов, проведём несколько тест-кейсов и unit-тестов. Они охватывают основные сценарии использования приложения и позволяют убедиться в корректности работы всех функций.

В процессе функционального тестирования были проверены ключевые сценарии взаимодействия пользователей с приложением.

**Таблица 5**

**Тест-кейс 001**

|  |  |
| --- | --- |
| Идентификатор | 001 |
| Название | Успешная авторизация |
| Описание | Проверка успешной авторизации пользователя с корректными данными. |
| Шаги | 1. Открыть экран авторизации. 2. Ввести корректный email и пароль. 3. Нажать кнопку «Войти». |
| Ожидаемый результат | Пользователь успешно авторизуется и переходит на главный экран. |
| Фактический результат | Пользователь авторизован и переведён на главный экран. |
| Статус | Успешно |

Продолжение таблицы 5

Тест-кейс 001 предназначен для проверки процесса входа в систему. Успешный результат теста подтверждает правильную аутентификацию пользователей при вводе корректных учетных данных. Данная функция имеет ключевое значение, поскольку обеспечивает доступ ко всем другим возможностям приложения.

**Таблица 6**

**Тест-кейс 002**

|  |  |
| --- | --- |
| Идентификатор | 002 |
| Название | Создание новой задачи |
| Описание | Проверка возможности создания новой задачи руководителем. |
| Шаги | 1. Авторизоваться как руководитель. 2. Перейти на экран «Проекты». 3. Выбрать проект. 4. Нажать кнопку «Добавить задачу». 5. Ввести название и описание задачи. 6. Назначить исполнителей. 7. Нажать кнопку «Создать задачу» |
| Ожидаемый результат | Новая задача отображается в списке задач проекта. |
| Фактический результат | Задача успешно создана и отображается в списке задач. |
| Статус | Успешно |

Продолжение таблицы 6

Тест-кейс 002 позволяет проверить возможность создания задач руководителями. Данный тест симулирует действия пользователя с ролью «Руководитель» и удостоверяется, что задача успешно создана и отображается в списке.

Также проводилось модульное тестирование (unit-тестирование) отдельных компонентов приложения.

**Таблица 7**

**Unit-тест**

|  |  |
| --- | --- |
| Название | Проверка валидации email |
| Описание | Unit-тест проверяет корректность работы функции isEmailValid, которая отвечает за валидацию email. |
| Ожидаемый результат | Валидный email (test@example.com) должен вернуть true.  Невалидный email (test.example.com, test@, пустая строка) должны вернуть false. |
| Фактический результат | Проверки email на валидацию прошли успешно. |
| Статус | Успешно |

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 - Результат unit-тестирования

Unit-тест проверяет функцию валидации email, охватывая правильные и неправильные адреса электронной почты, чтобы убедиться в правильности логики проверки введенных данных. Успешное завершение теста подтвердило, что функция isEmailValid работает должным образом и фильтрует неверные форматы email.

Все выполненные тесты были успешно пройдены, что подтверждает корректную работу ключевых функций приложения:

* Аутентификация пользователей.
* Создание и управление задачами.
* Валидация входных данных.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы «Разработка системы для управления проектами и задачами для сотрудников рекламного агентства» удалось достичь всех поставленных целей:

* Проведен анализ специфики деятельности и изучены методы управления проектами.
* Разработаны концепция и функциональные возможности мобильного приложения, учитывающие потребности рекламного агентства.
* Произведен обзор современных технологий для разработки мобильных приложений.
* Создана архитектура базы данных приложения.
* Реализована базовая версия приложения с интуитивно понятным интерфейсом.
* Проведено тестирование разработанного приложения.

В рамках курсовой работы создана концепция и реализовано мобильное приложение для управления проектами и задачами в рекламном агентстве. Приложение готово к успешному внедрению в деятельность рекламного агентства, что поможет оптимизировать процессы управления проектами, увеличить производительность труда сотрудников и повысить уровень удовлетворенности клиентов.

В будущем планируется расширить функциональность приложения:

* Разработать систему уведомлений.
* Интегрировать приложение с внешними сервисами (синхронизация задач с календарями).
* Доработать интерфейс на основе отзывов пользователей.
* Добавить функцию формирования отчетов по выполненным задачам и проектам.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Android Developers. Документация по Android [Электронный ресурс]. – URL: https://developer.android.com/docs (дата обращения: 10.10.2023).
2. Jetpack Compose. Документация [Электронный ресурс]. – URL: https://developer.android.com/jetpack/compose (дата обращения: 10.10.2023).
3. Firebase. Документация [Электронный ресурс]. – URL: https://firebase.google.com/docs (дата обращения: 10.10.2023).
4. Swift. Документация от Apple [Электронный ресурс]. – URL: https://developer.apple.com/swift/ (дата обращения: 10.10.2023).
5. Flutter. Документация [Электронный ресурс]. – URL: https://flutter.dev/docs (дата обращения: 10.10.2023).
6. React Native. Документация [Электронный ресурс]. – URL: https://reactnative.dev/docs/getting-started (дата обращения: 10.10.2023).
7. Xcode. Документация от Apple [Электронный ресурс]. – URL: https://developer.apple.com/xcode/ (дата обращения: 10.10.2023).
8. ГОСТ 7.0.100–2018. Robin Wieruch W The Road to Firebase: Your journey to master Firebase in JavaScript / W Robin Wieruch: Independently published, 2021. – 799 с. – ISBN 979-8514853953
9. ГОСТ 7.0.100–2018. Грофф Д. Р. SQL. Полное руководство / Д. Р. Грофф, П. Н. Вайнберг, Э. Д. Опель. – 3-е изд. – СПб.: Диалектика, 2019. – 560 с.
10. ГОСТ 7.0.100–2018. Бьюли А. Изучаем SQL / А. Бьюли. – Москва: Символ-Плюс, 2019. – 308 с. – ISBN 978-5-932860-5-19
11. ГОСТ 7.0.100–2018. Бурнет Э. Привет, Android! Разработка мобильных приложений / Э. Бурнет. – СПБ: Питер, 2016. – 256 с.
12. ГОСТ 7.0.100–2018. Дейтел Пол Android для разработчиков / Пол Дейтел, Харви Дейтел, Александер Уолд. – СПб: Питер, 2020. – 512 с.
13. ГОСТ 7.0.100–2018. Griffiths D. Head First Android Development / D. Griffiths, D. Griffiths. – United States of America: O'REILLY, 2017. – 888 с.
14. ГОСТ 7.0.100–2018. Льюис Шон Нативная разработка мобильных приложений. Перекрестный справочник для iOS и Android / Шон Льюис, Майк Данн. – М.: ДМК-Пресс, 2020. – 376 с. – ISBN 978-5-97060-845-6
15. ГОСТ 7.0.100–2018. Michael В.G. Android. Программирование для профессионалов / В.G. Michael. – Санкт Питербург: Питер, 2019. – 704 с. – ISBN 978-5-4461-1657-7
16. ГОСТ 7.0.100–2018. Скотт А. Д. Разработка на JavaScript. Построение кроссплатформенных приложений с помощью GraphQL, React, React Native и Electron / А. Д. Скотт. – СПб.: Питер, 2021. – 320 с.
17. ГОСТ 7.0.100–2018. Bonnie, Eisenman Learning React Native: Building Native Mobile Apps with JavaScript / Eisenman Bonnie. – 714: O'Reilly Media, 2017. – 240 с.
18. ГОСТ 7.0.100–2018. Windmill, E. Flutter in Action / E. Windmill. – New York: Manning Publications, 2021. – 250 с.
19. ГОСТ 7.0.100–2018. Алеев, А. Быстрый старт Flutter-разработчика / А. Алеев. – Москва: Издательские решения, 2019. – 173 с.
20. ГОСТ 7.0.100–2018. Наттинг Д. Swift. Разработка приложений в среде Xcode для iPhone и iPad с использованием iOS SDK / Д. Наттинг. – Москва: Вильямс, 2019. – 816 с.
21. ГОСТ 7.0.100–2018. Усов В.А. Swift. Основы разработки приложений под iOS, iPadOS и macOS. 6-е изд. дополненное и переработанное / В.А. Усов. – Москва: Прогресс книга, 2021. – 544 с.
22. ГОСТ 7.0.100–2018. Сазерленд Д. Scrum. Революционный метод управления проектами / Д. Сазерленд. – Москва: Манн, Иванов и Фербер, 2022. – 272 с.
23. ГОСТ 7.0.100–2018. Коллектив, авторов Канбан и «точно вовремя» на Toyota. Менеджмент начинается на рабочем месте / авторов Коллектив. – Москва: Альпина Паблишер, 2015. – 240 с.
24. ГОСТ 7.0.100–2018. Майк Кон Scrum: гибкая разработка ПО / Кон Майк. – Москва: Диалектика-Вильямс, 2013. – 256 с.
25. ГОСТ 7.0.100–2018. Кон М. Agile: Оценка и планирование проектов / М. Кон. – Москва: Альпина Паблишер, 2022. – 424 с.
26. ГОСТ 7.0.100–2018. Хинер, П. У. UX Design for Mobile. Design apps that deliver impressive mobile experiences / П. У. Хинер. – Америка: Packt Publishing, 2017. – 354 с. – ISBN 9781787283428.
27. ГОСТ 7.0.100–2018. Martin, H.R Patterns of Enterprise Application Architecture / H.R Martin. – New York: Addison-Wesley Professional, 2002. – 250 с.
28. ГОСТ 7.0.100–2018. Richards, M Fundamentals of Software Architecture: An Engineering Approach / M Richards, N Ford. – Иллюстрированное издание. – Sebastopol, California: O'Reilly Media, 2020. – 419 с. – ISBN 978-1492043454.