СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем».

В условиях современной цифровой экономики эффективное управление проектами становится ключевым фактором успешной деятельности любой организации. Растущая сложность бизнес-процессов, необходимость координации работы распределенных команд и требования к оперативности принятия решений обуславливают потребность в современных автоматизированных системах управления проектами.

Актуальность темы исследования определяется несколькими факторами. Во-первых, традиционные методы управления проектами, основанные на использовании офисных приложений общего назначения (Microsoft Excel, Word) или устаревших информационных систем, не обеспечивают необходимого уровня интеграции, оперативности и контроля. Во-вторых, существующие коммерческие решения (Microsoft Project, Jira, Asana) характеризуются высокой стоимостью владения, сложностью настройки под специфические потребности организации и зависимостью от постоянного интернет-соединения. В-третьих, мобильные версии большинства систем управления проектами предоставляют ограниченную функциональность по сравнению с веб-версиями, что снижает эффективность работы сотрудников в условиях мобильности.

Объектом исследования является процесс автоматизации управления проектами в корпоративной среде с использованием современных информационных технологий.

Предметом исследования выступают методы и средства разработки интегрированной системы управления проектами, включающей мобильное приложение для платформы Android и серверную часть на базе веб-технологий.

Цель дипломной работы заключается в разработке и реализации комплексной системы управления проектами и отчетностью, обеспечивающей эффективную координацию проектной деятельности через мобильное Android-приложение с серверной поддержкой.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ предметной области и существующих решений в сфере управления проектами, выявить их преимущества и недостатки;
2. Разработать архитектуру системы управления проектами, определить функциональные и нефункциональные требования к программному продукту;
3. Обосновать выбор технологий и инструментов разработки, обеспечивающих оптимальное соотношение производительности, надежности и простоты сопровождения;
4. Спроектировать и реализовать структуру базы данных для хранения информации о пользователях, проектах и отчетах с обеспечением целостности и безопасности данных;
5. Разработать серверную часть системы на базе фреймворка Flask, реализующую REST API для взаимодействия с клиентскими приложениями;
6. Создать мобильное Android-приложение с использованием современных архитектурных паттернов и библиотек, обеспечивающее полнофункциональный доступ к системе управления проектами;
7. Реализовать систему аутентификации и авторизации с ролевой моделью доступа, обеспечивающую информационную безопасность;
8. Разработать модули управления пользователями, проектами и отчетностью с возможностью создания, редактирования и удаления записей;
9. Внедрить систему аналитической отчетности и статистики для контроля эффективности проектной деятельности;
10. Провести тестирование разработанной системы, выполнить отладку и оптимизацию производительности.

Методы исследования включают анализ предметной области, сравнительный анализ существующих решений, объектно-ориентированное проектирование, методы разработки программного обеспечения, тестирование и отладку программных систем.

Научная новизна работы заключается в разработке интегрированного подхода к созданию системы управления проектами, сочетающего преимущества

мобильных технологий с мощностью серверных решений. Предложенная архитектура обеспечивает автономную работу мобильного приложения с синхронизацией данных при восстановлении соединения, что повышает надежность системы в условиях нестабильного интернет-соединения.

Практическая значимость работы определяется возможностью использования разработанной системы в реальных условиях корпоративной среды для автоматизации процессов управления проектами. Система обеспечивает снижение временных затрат на административные процедуры, повышение качества планирования и контроля проектов, улучшение координации между участниками проектных команд.

Структура работы включает введение, три основных раздела, заключение, список литературы и приложения. В первом разделе представлено теоретическое обоснование разрабатываемого программного продукта, включающее анализ предметной области, сравнение с аналогами, моделирование системы и обоснование выбора технологий. Во втором разделе описана практическая разработка программного продукта, включая проектирование архитектуры, создание структуры данных, разработку пользовательского интерфейса и алгоритмов программы. Третий раздел посвящен вопросам охраны труда при работе с компьютерной техникой.

Технические характеристики разработанной системы:

- Серверная часть реализована на языке Python с использованием фреймворка Flask версии 2.1.0;

  База данных MySQL обеспечивает надежное хранение информации с поддержкой транзакций;
- Мобильное приложение разработано для платформы Android (API уровень 24+) на языке Kotlin;
- Система поддерживает одновременную работу до 50 пользователей;
- Реализована ролевая модель доступа с четырьмя уровнями прав: начальник, аналитик и программист;
- Обеспечена возможность работы в автономном режиме с последующей синхронизацией данных;

Разработанная система представляет собой современное решение для автоматизации управления проектами, сочетающее простоту использования с мощными функциональными возможностями и обеспечивающее эффективную поддержку проектной деятельности в организациях различного масштаба.

1 Теоретическое обоснование разрабатываемого программного продукта

1.1 Описание предметной области

В современных условиях цифровой трансформации бизнеса управление проектами и отчетностью становится критически важным аспектом успешной деятельности любой организации. Предметной областью разрабатываемого программного продукта является автоматизация процессов управления проектами, пользователями и отчетностью в корпоративной среде IT-отдела.

Система управления проектами представляет собой комплексное решение, которое объединяет в себе функции планирования, контроля выполнения, отчетности и анализа проектной деятельности. В рамках данного дипломного проекта разрабатывается интегрированная система, состоящая из мобильного Android-приложения и серверной части на базе Flask API.

Основными участниками процесса управления проектами в рамках разрабатываемой системы являются:

Начальники – руководители высшего звена с максимальными правами доступа, которые могут управлять всеми аспектами системы, включая создание и редактирование пользователей, проектов, назначение ролей и контроль доступа к функциональности. Начальники отвечают за стратегическое планирование проектов, распределение ресурсов, контроль бюджетов и принятие ключевых решений по развитию проектной деятельности.

Программисты – специалисты по разработке программного обеспечения, которые непосредственно участвуют в реализации технических задач проектов. Они имеют права на просмотр назначенных им проектов, создание детальных технических отчетов о выполненной работе, отслеживание статуса выполнения задач разработки, документирование кода и участие в процессах тестирования и отладки.

Аналитики – специалисты по анализу требований и бизнес-процессов, ответственные за планирование, координацию и контроль выполнения проектов с точки зрения соответствия бизнес-требованиям. Они имеют права на создание и редактирование проектов, анализ требований, составление технических заданий, контроль сроков выполнения, анализ отчетности и взаимодействие с заказчиками.

Ключевые бизнес-процессы, автоматизируемые системой:

Управление пользователями и ролями – регистрация новых сотрудников IT-отдела, назначение ролей (начальник, программист, аналитик) и прав доступа, управление профилями пользователей.

Управление IT-проектами – создание новых проектов разработки, определение технических требований и задач, назначение ответственных программистов и аналитиков, установка временных рамок, контроль статуса выполнения разработки.

Система технической отчетности – создание отчетов о ходе выполнения разработки, фиксация достигнутых результатов, анализ эффективности работы команды разработчиков, документирование проблем и их решений.

Статистический анализ проектной деятельности – формирование аналитических отчетов по производительности команды, визуализация данных о прогрессе разработки, анализ трендов и показателей эффективности IT-проектов.

Разрабатываемая система решает следующие проблемы традиционного управления IT-проектами:

Отсутствие централизованного хранения информации о проектах разработки – все данные о проектах, участниках команды и технических отчетах хранятся в единой базе данных MySQL с обеспечением целостности и безопасности информации.

Сложность координации между программистами и аналитиками - мобильное приложение обеспечивает мгновенный доступ к актуальной информации о проектах разработки и возможность оперативного обновления статусов выполнения задач.

Недостаток аналитической информации о ходе разработки – система предоставляет комплексные инструменты для анализа эффективности проектной деятельности IT-отдела и формирования технических отчетов.

Проблемы с контролем доступа к техническим данным – реализована гибкая система ролей и прав доступа, обеспечивающая информационную безопасность и разграничение доступа к конфиденциальной информации о проектах.

1.2 Сравнительный анализ программ аналогов

Для разработки мобильного приложения учета проектов IT-отдела были проанализированы специализированные системы управления проектами и корпоративные решения. В сравнении участвуют:

1. Jira;
2. Trello;
3. Microsoft Teams (Planner);
4. Redmine.

Эти платформы частично охватывают функционал, требуемый в задании, но разрабатываемое приложение предлагает уникальные решения для учета проектов в контексте разработки корпоративных информационных систем.

1. Jira (Atlassian) – управление IT-проектами, задачами и Agile-процессами.

Ключевые функции:

- Создание задач, спринтов, дорожных карт;
- Гибкая настройка workflows и ролей (разработчик, тестировщик, менеджер);
- Интеграция с Confluence, Bitbucket, GitHub;
- Генерация отчетов (Burndown, Velocity).

Преимущества:

- Поддержка Scrum и Kanban;
- Мощная аналитика и кастомизация;
- Мобильное приложение для iOS и Android.

Недостатки:

- Сложность настройки для небольших команд
- Высокая стоимость лицензии;
- Отсутствие локального хранения данных (облачная архитектура);
- Нет поддержки генерации отчетов в Word.

2. Trello (Atlassian) – управление проектами через Kanban-доски.

Ключевые функции:

- Создание карточек задач с описанием, метками и сроками;
- Коллаборация в режиме реального времени;
- Интеграция с Slack, Google Drive.

Преимущества:

- Простой и интуитивный интерфейс;
- Бесплатный базовый тариф;
- Кроссплатформенность (веб, iOS, Android).

Недостатки:

- Отсутствие иерархии проектов (дерева);
- Нет ролевой системы с разграничением прав;
- Ограниченная аналитика и отчетность (только экспорт в PDF/CSV).

3. Microsoft Teams (Planner) – управление задачами в рамках корпоративной экосистемы Microsoft.

Ключевые функции:

- Создание планов с задачами и сроками;
- Интеграция с Office 365 (Excel, Word, SharePoint);
- Ролевой доступ через Active Directory.

Преимущества:

- Глубокая интеграция с Office;
- Поддержка корпоративной безопасности;
- Генерация отчетов в Excel.

Недостатки:

- Нет иерархического представления проектов;
- Ограниченная кастомизация интерфейса;
- Зависимость от экосистемы Microsoft.

4. Redmine – open-source система управления проектами.

Ключевые функции:

- Гибкая настройка ролей и прав доступа;
- Учет времени, контроль версий;
- Интеграция с Git, SVN.

Преимущества:

- Бесплатность и открытый исходный код;

- Поддержка множества плагинов;
- Локальное развертывание (без облака).

Недостатки:

- Устаревший интерфейс;
- Нет мобильной версии;
- Сложность генерации отчетов в Word/Excel.

Таблица 2.1 – ключевые характеристики аналогов и разрабатываемого приложения

Критерий	Jira	Trello	MS Teams (Planner)	Redmine
Дерево проектов	Нет (только задачи)	Нет	Нет	Нет
Ролевая система	Да (кастомизируемая)	Нет	Да (через AD)	Да
Кроссплатформенность	Да (iOS/Android)	Да	Да	Нет
Генерация отчетов	PDF, Excel	CSV, PDF	Excel	PDF
Локальное хранение данных	Нет (облако)	Нет (облако)	Нет (облако)	Да
Современный интерфейс	Да	Да	Умеренно	Нет
Бесплатная версия	Нет	Да	Нет (только с Office 365)	Да
Поддержка офлайн-режима	Нет	Нет	Нет	Нет
Интеграция с корпоративными системами	Да (сторонние сервисы)	Ограниченно	Да (Microsoft)	Да (плагины)

Анализ показал, что существующие аналоги:

- Не поддерживают иерархическое представление проектов — большинство решений ограничиваются плоскими списками задач (Trello, Jira);
- Не предлагают готовой ролевой системы для IT-отделов — кастомизация ролей в Jira и Redmine требует настройки, а в Trello и Planner роли отсутствуют;

- Зависимы от облачной инфраструктуры — отсутствие офлайн-доступа и локального хранения данных;
- Ограничены в генерации отчетов — поддержка Word отсутствует, Excel-отчеты часто требуют ручной доработки.

Преимущества разрабатываемого решения:

- Специализация на IT-проектах: Учет аналитиков, программистов, сроков и деталей проектов в структурированном дереве;
- Гибкая система ролей: Четкое разграничение прав между аналитиками, программистами и руководством;
- Локальная база данных: Безопасное хранение данных в MySQL, работа в офлайн-режиме;
- Кроссплатформенность: Единая кодовая база на Kotlin Multiplatform для iOS и Android;
- Удобная отчетность: Экспорт в Word и Excel с автоматическим формированием сводок и премиальных расчетов.

Таким образом, приложение закрывает пробел в инструментах для IT-отделов, предлагая специализированные функции, отсутствующие в популярных аналогах.

1.3 Моделирование проектируемой системы

Архитектура разрабатываемой системы основана на клиент-серверной модели с четким разделением ответственности между компонентами и адаптацией под роли пользователей. Система состоит из трех основных уровней:

Уровень представления (Presentation Layer) – мобильное Android-приложение, разработанное на языке Kotlin с использованием современных архитектурных паттернов. Приложение реализует принципы Material Design и обеспечивает адаптивный пользовательский интерфейс для разных ролей:

– Интерфейс начальника: фокус на аналитике, общем контроле проектов, управлении командой

– Интерфейс программиста: фокус на технических задачах, детальных отчетах, статусах разработки

– Интерфейс аналитика: фокус на требованиях, планировании, координации между командой и заказчиками

Уровень бизнес-логики (Business Logic Layer) – серверная часть, реализованная на Python с использованием фреймворка Flask. Этот уровень отвечает за обработку бизнес-правил IT-отдела, валидацию данных, управление сессиями и авторизацией пользователей с учетом их ролей.

Уровень данных (Data Layer) – система управления базой данных MySQL, обеспечивающая надежное хранение и быстрый доступ к информации о сотрудниках IT-отдела, проектах разработки и технических отчетах.

На рисунке 1.1 представлена контекстная диаграмма IDEF0, показывающая главную задачу, которую решает выполнение бизнес-процесса.

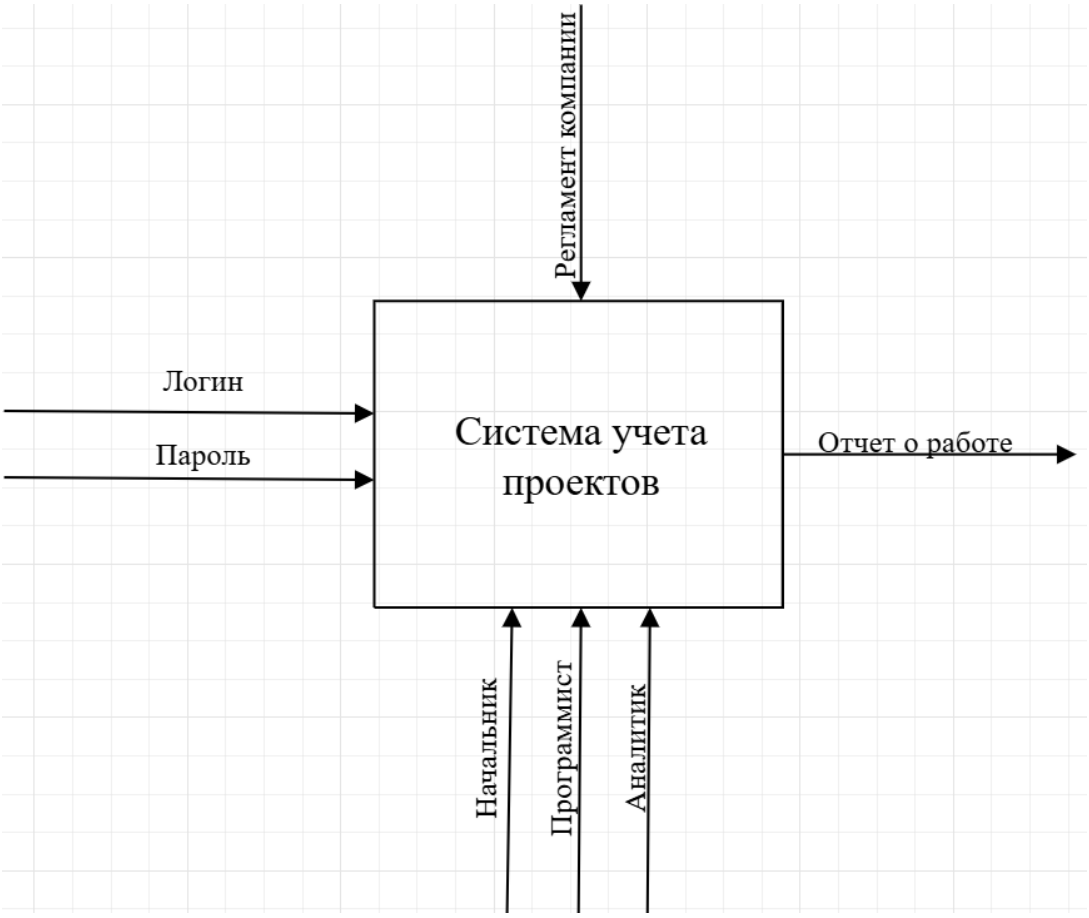

Рисунок 1.1 – Контекстная диаграмма IDEF0

1.4 Разработка функциональных требований к программной системе

1.4.1 Функциональное назначение

Разрабатываемая программная система представляет собой комплексное решение для управления проектами в организации. Основное назначение системы заключается в автоматизации процессов учета, контроля и анализа проектной деятельности. Система обеспечивает:

1) Централизованное хранение информации:

- полные данные о проектах (наименование, сроки выполнения, этапы реализации);
- информация о сотрудниках (аналитики, разработчики, начальник) с указанием их ролей и компетенций;
- информация об отчетах (дата, электронный документ, кроки, статус);
- история изменений и версий проектов.

2) Разграничение прав доступа:

- трехуровневая система ролей (Аналитик, Программист, Начальник);
- дифференцированный доступ к функционалу в зависимости от роли;
- механизм аудита действий пользователей.

3) Формирование отчетной документации:

- автоматическая генерация отчетов в различных форматах;
- настраиваемые шаблоны отчетных форм;
- возможность экспорта данных для внешней обработки.

4) Интеграционные возможности:

- подключение к серверным СУБД (MySQL);
- API для интеграции с другими корпоративными системами.

1.4.2 Эксплуатационное назначение

Программный продукт предназначен для ежедневного использования в рабочих процессах организации. Основные аспекты эксплуатации:

1) Режим работы:

- круглосуточная доступность системы;
- плановое техническое обслуживание в нерабочие часы;
- возможность горячего резервирования.

2) Пользовательские группы:

- руководящий состав (Начальники отделов):
  - полный контроль над проектами;
  - управление персоналом;
  - анализ эффективности работы отделов;
  - формирование стратегических отчетов.
- Технические специалисты (Программисты):
  - ведение проектной документации;
  - отслеживание сроков выполнения задач;
  - взаимодействие с аналитиками;
  - фиксация рабочего времени.
- Аналитические работники (Аналитики):
  - сбор и систематизация требований;
  - подготовка аналитических материалов;
  - контроль соответствия результатов поставленным задачам;
  - формирование отчетов о ходе работ.

3) Результаты работы системы:

- оперативные отчеты о состоянии проектов;
- аналитические выкладки по эффективности работы отделов;
- автоматизированные расчеты премиальных выплат;
- архив выполненных проектов с возможностью ретроспективного анализа.

1.4.3 Требования к функциональным характеристикам

1) Управление пользователями:
- регистрация новых пользователей с назначением ролей;
- аутентификация через API;
- восстановление доступа при утере учетных данных;
- управление профилями пользователей.

2) Управление проектами:
- создание карточек проектов с указанием:
  - наименования и описания;
  - сроков выполнения (плановых и фактических);
  - ответственных исполнителей;
  - бюджетных показателей.
- календарное планирование задач;
- контроль этапов выполнения.
- система уведомлений о критических изменениях.

3) Работа с персоналом:
- каталог сотрудников с фильтрацией по:
  - отделам;
  - должностям;
  - навыкам;
  - текущей загрузке.
- назначение ответственных за проекты;
- учет рабочего времени;
- анализ загрузки персонала.

4) Отчетность и аналитика:
- стандартные отчетные формы:
  - по проектам;
  - по сотрудникам;
  - финансовые;
  - статистические.
- пользовательские отчеты с настраиваемыми параметрами;

- визуализация данных (графики, диаграммы);
- экспорт в различные форматы (XLSX, DOCX).

5) Интеграционные сервисы:

- flask_api для внешних систем;
- механизм импорта/экспорта данных;
- сервис очередей для асинхронных задач.

1.4.4 Входные и выходные данные

1) Источники входных данных:
- ручной ввод через пользовательский интерфейс;
- пакетная загрузка из файлов (JSON, SQL);
- импорт из внешних систем через API.

2) Типы выходных данных:
- экранные формы с возможностью фильтрации;
- печатные формы документов;
- файлы для внешней обработки;
- уведомления и оповещения;
- дашборды с ключевыми показателями.

3) Форматы данных:
- внутренний JSON формат для обмена между компонентами;
- стандартные офисные форматы для отчетности;
- специализированные форматы для интеграции.

1.4.5 Требования к надежности

1) Обеспечение бесперебойной работы:
- механизм автоматического восстановления после сбоев;
- репликация критически важных данных;
- мониторинг работоспособности компонентов.

2) Защита данных:
- шифрование конфиденциальной информации;
- разграничение прав доступа на уровне данных;
- журналирование всех значимых событий.

3) Резервное копирование:
- автоматизированный процесс создания бэкапов;
- хранение нескольких поколений резервных копий;
- процедуры аварийного восстановления.

4) Производительность:
- оптимизация запросов к базе данных;
- кэширование часто используемых данных;
- балансировка нагрузки в пиковые периоды.

1.4.6 Требования к техническим средствам

1) Серверная часть:
   - аппаратные требования:
     - процессор: 4+ ядер, 2.5+ GHz;
     - оперативная память: 16+ GB;
     - дисковое пространство: 500+ GB SSD;
     - сетевая карта: 1+ Gbit.
   - программное окружение:
     - ОС: Windows 10+;
     - СУБД: MySQL 8.0+;
     - процессор: 2+ ядер, 2+ GHz;
     - оперативная память: 8+ GB.

2) Клиентская часть:

- стационарные рабочие места:
  - ОС: Windows 10+;
  - процессор: 2+ ядер, 2+ GHz;
  - оперативная память: 8+ GB.
- мобильные устройства:
  - ОС: Android 8.0+/iOS 13+;
  - оперативная память: 2+ GB;
  - свободного места: 100+ MB.

1.4.7 Требования к программной совместимости

1) Серверные технологии:
- язык программирования: Kotlin;
- система сборки: Gradle.

2) Клиентские технологии:
- мобильные платформы: Kotlin;
- нативные компоненты: Android SDK.

3) Форматы данных:
- обменные: JSON, XML;
- отчетные: XLSX, DOCX.

4) Протоколы взаимодействия:
- HTTP/HTTPS.

1.4.8 Техническая документация

1) Архитектурные решения:
- диаграммы компонентов системы;
- схемы взаимодействия модулей;
- описание API.
2) Руководство администратора:
- процедуры установки и настройки;
- методики обслуживания;
- рекомендации по масштабированию.
3) Спецификации баз данных:
- диаграммы IDEF0, ER;
- описание таблиц и отношений.

1.4.9 Пользовательская документация

1) Руководство пользователя:
- пошаговые инструкции по основным сценариям (Руководство пользователя);
- описание интерфейсов;
- примеры работы с системой.
2) Справочная система:
- контекстная помощь;
3) Обучающие материалы:
- тестовые сценарии.

1.4.10 Эксплуатационная документация

1) Регламенты работы:
- графики технического обслуживания;
- процедуры обновления;
- политики безопасности.

2) Методики тестирования:
- наборы тестовых случаев;
- чеклисты проверок;
- сценарии нагрузочного тестирования.

3) Планы развития:
- дорожная карта;
- календарь обновлений;
- система учета пожеланий пользователей.

1.5 Обоснование выбора средств реализации программной системы

1.5.1 Анализ требований к клиентской части

1) Для мобильного приложения учета проектов требовалось решение, обеспечивающее:

- широкий охват пользовательской аудитории;
- стабильную работу на различных устройствах;
- возможности для дальнейшего масштабирования;
- поддержку современных стандартов разработки.

2) Android была выбрана по следующим причинам:

- доминирующая доля рынка мобильных ОС (более 70%);
- гибкость в настройке и кастомизации интерфейса;
- открытая экосистема с широкими возможностями интеграции;
- поддержка современных технологий (Kotlin, Jetpack Components).

3) Язык программирования Kotlin обеспечивает:

- полную совместимость с существующей Java-инфраструктурой;
- современные языковые конструкции (корутины, null-safety);
- снижение количества шаблонного кода;
- официальную поддержку Google для Android-разработки;
- улучшенную производительность по сравнению с Java.

1.5.2 Серверная архитектура системы

1) Серверная часть должна была удовлетворять следующим условиям:
- простота разработки и поддержки;
- высокая производительность при обработке запросов;
- гибкость в масштабировании;
- надежная работа с базами данных;
- безопасность передаваемых данных.

2) Микрофреймворк Flask был выбран благодаря:
- минималистичному и понятному API;
- легкости в настройке и развертывании;
- возможности создания RESTful API;
- богатой экосистеме расширений;
- поддержке асинхронных операций.

3) использование Python обеспечивает:
- быструю разработку благодаря простому синтаксису;
- доступ к мощным библиотекам для работы с данными;
- кроссплатформенную совместимость;
- хорошую документацию и сообщество;
- интеграцию с различными СУБД.

1.5.3 Система хранения данных

1) Критерии выбора СУБД

При выборе системы управления базами данных учитывались:

- надежность и отказоустойчивость;
- производительность при работе с проектами;
- поддержка транзакций;
- возможности резервного копирования;
- совместимость с выбранными технологиями.

2) Обоснование выбора MySQL

Реляционная СУБД MySQL была выбрана потому, что:

- доказанная надежность в production-средах;
- высокая производительность для OLTP-нагрузок;
- поддержка ACID-транзакций;
- широкие возможности настройки и оптимизации;
- хорошая интеграция с Python.

1.5.4 Взаимодействие компонентов системы

1) Схема обмена данными

Архитектура системы построена по принципу:

- клиентское приложение формирует HTTP-запросы;
- сервер обрабатывает запросы и взаимодействует с БД;
- данные передаются в формате JSON;
- реализована система кэширования на клиенте.

2) безопасность передачи данных

Для защиты информации реализованы:

- шифрование передаваемых данных (HTTPS);
- аутентификация по токенам;
- валидация входных параметров;
- защита от SQL-инъекций;
- разграничение прав доступа.

1.5.5 Дополнительные технологии

1) Инструменты разработки

В процессе разработки используются:

- Android Studio – для клиентской части;
- PyCharm – для серверной разработки;
- MySQL – для хранения базы данных.

2) Вспомогательные библиотеки

Для расширения функциональности применяются:

- Retrofit – для сетевых запросов в Android;
- MySQL-Connector – для работы с БД в Python.

1.6 Вывод по разделу