СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем имеет целью создание эффективного инструмента для управления проектами и повышения прозрачности рабочих процессов. В условиях современного бизнеса, где информационные технологии играют ключевую роль, необходимость в надежных и удобных решениях для управления проектами становится все более актуальной. Данное приложение призвано упростить процесс контроля и координации работы сотрудников, обеспечить доступ к актуальной информации о проектах и повысить общую эффективность работы отдела.

Практика разработки данного приложения направлена на комплексное освоение студентами всех видов профессиональной деятельности по специальности, формирование общих и профессиональных компетенций, а также приобретение необходимых умений и опыта практической работы. Учебная практика реализуется в рамках модулей по основным видам профессиональной деятельности и направлена на последующее освоение общих и профессиональных компетенций по специальности.

В настоящей работе представлен отчет о разработке кроссплатформенного мобильного приложения на языке Kotlin в среде Android Studio с использованием MySQL Workbench. Основными видами деятельности являлись: разработка спецификаций компонентов программных систем, написание кода программного продукта на основе готовых спецификаций на уровне модуля, выполнение отладки программных модулей с использованием специализированных программных средств, тестирование и оптимизация программных модулей, а также разработка компонентов проектной и технической документации с использованием графических языков спецификаций.

Задачи практики включают:

1. Выполнение разработки спецификаций отдельных компонентов.
2. Осуществление разработки кода программного продукта на основе готовых спецификаций на уровне модуля.

3. Выполнение отладки программных модулей с использованием специализированных программных средств.
4. Выполнение тестирования программных модулей.
5. Осуществление оптимизации программного кода модуля.
6. Разработка компонентов проектной и технической документации с использованием графических языков спецификаций.
7. Понимание сущности и социальной значимости своей будущей профессии, проявление к ней устойчивого интереса.
8. Организация собственной деятельности, выбор типовых методов и способов выполнения профессиональных задач, оценка их эффективности и качества.
9. Принятие решений в стандартных и нестандартных ситуациях и несение за них ответственности.
10. Осуществление поиска и использования информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.
11. Использование информационно-коммуникационных технологий в профессиональной деятельности.
12. Работа в коллективе и в команде, эффективное общение с коллегами, руководством, потребителями.
13. Принятие ответственности за работу членов команды (подчиненных), за результат выполнения заданий.
14. Самостоятельное определение задач профессионального и личностного развития, самообразование, осознанное планирование повышения квалификации.
15. Ориентация в условиях частой смены технологий в профессиональной деятельности.

Приложение предусматривает создание дерева проектов с полями для хранения информации о названии проекта, аналитиках, разработчиках, датах и других важных аспектах. Система ролей включает три категории пользователей: аналитики, программисты и начальники, каждый из которых имеет свои уникальные права доступа и функционал. При нажатии на раздел с аналитиками,

программистами или начальниками выдается информация обо всех сотрудниках данной категории. При выборе конкретного сотрудника отображаются сроки проектов, в которых он принимал участие, их названия и другие детали.

Кроме того, приложение позволяет формировать отчеты по работе отдела, сводки и информацию о премиях в формате электронных документов (PDF, Word, Excel), что значительно упрощает процесс отчетности и анализа эффективности работы.

Таким образом, данное мобильное приложение не только удовлетворяет потребности отдела разработки и внедрения корпоративных информационных систем, но и способствует профессиональному росту студентов, готовя их к реальным задачам в области программирования и управления проектами. Оно обеспечивает прозрачность и удобство в управлении проектами, повышает эффективность работы отдела и способствует более качественному выполнению поставленных задач.

ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Функциональное моделирование в нотации IDEF0

Предметная область мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем включает управление проектами, пользователями и отчетами. Основные сущности системы: проекты, пользователи и роли. Проекты характеризуются названием, датами начала и окончания, участниками (аналитиками, программистами, начальниками) и другими атрибутами. Пользователи имеют роли, которые определяют их права доступа и функционал в системе. Отчеты генерируются на основе данных о проектах и пользователях и экспортируются в формате PDF, Word или Excel.

Система предусматривает три роли: аналитик, программист и начальник. Аналитики вводят и обновляют данные о проектах, программисты разрабатывают и тестируют функционал приложения, а начальники контролируют выполнение проектов и генерируют отчеты. Приложение позволяет просматривать информацию о сотрудниках и их участии в проектах, а также формировать сводки и отчеты по работе отдела. Для эффективного управления данными о деятельности компании необходима разработка информационной системы, которая будет включать в себя следующие основные сущности:

Общий модуль (shared/)

src/commonMain/

- DatabaseService.kt: Интерфейс для работы с базой данных, определяющий методы для выполнения CRUD-операций. Включает методы для добавления, обновления, удаления и получения данных о проектах, пользователях и отчетах.
- DatabaseHelper.kt: Вспомогательные функции для работы с базой данных, такие как подключение, выполнение запросов и управление транзакциями. Обеспечивает надежное взаимодействие с базой данных и обработку ошибок.

- Project.kt: Модель проекта, включающая поля для названия, дат начала и окончания, списка участников, статуса и других атрибутов. Используется для хранения и манипуляции данными о проектах.
- User.kt: Модель пользователя, включающая поля для ФИО, роли, контактных данных и других атрибутов. Используется для хранения и манипуляции данными о пользователях.
- Role.kt: Модель роли, определяющая права доступа и функционал пользователей. Включает поля для названия роли и описания прав доступа.
- ProjectRepository.kt: Репозиторий для работы с проектами, реализующий методы интерфейса DatabaseService. Обеспечивает доступ к данным о проектах и их манипуляцию.
- UserRepository.kt: Репозиторий для работы с пользователями, реализующий методы интерфейса DatabaseService. Обеспечивает доступ к данным о пользователях и их манипуляцию.
- ReportRepository.kt: Репозиторий для работы с отчетами, включающий методы для генерации и экспорта отчетов. Обеспечивает доступ к данным о отчетах и их манипуляцию.
- GetProjectsUseCase.kt: Use Case для получения списка проектов с возможностью фильтрации и сортировки. Включает методы для получения проектов по различным критериям.
- GetUsersByRoleUseCase.kt: Use Case для получения списка пользователей по роли. Включает методы для получения пользователей по заданной роли.
- GenerateReportUseCase.kt: Use Case для генерации отчетов на основе данных о проектах и пользователях. Включает методы для создания отчетов в различных форматах.
- DateUtils.kt: Утилиты для работы с датами, включая форматирование, преобразование и вычисление разницы между датами. Используется для обработки дат в проектах и отчетах.
- ReportUtils.kt: Утилиты для генерации отчетов в формате PDF, Word или Excel. Включает методы для создания и сохранения отчетов.

src/androidMain/ и src/iosMain/

- AndroidDatabaseService.kt: Реализация DatabaseService для Android, включающая специфичные для платформы методы работы с базой данных. Обеспечивает взаимодействие с базой данных на уровне Android-приложения.

Android-приложение (androidApp/)

src/main/

- LoginActivity.kt: Экран авторизации, позволяющий пользователям входить в систему. Включает поля для ввода логина и пароля, а также кнопку для входа. Обрабатывает процесс авторизации и перенаправляет пользователя на главный экран при успешном входе.
- MainActivity.kt: Главный экран приложения, отображающий основные функции и навигацию. Включает меню для перехода к списку проектов, пользователям и отчетам. Обеспечивает централизованный доступ ко всем функциям приложения.
- ProjectListFragment.kt: Фрагмент для отображения списка проектов с возможностью фильтрации и сортировки. Включает элементы интерфейса для отображения информации о проектах и взаимодействия с ними. Позволяет пользователям просматривать и управлять проектами.
- UserDetailsFragment.kt: Фрагмент для отображения деталей пользователя, включая информацию о его участии в проектах. Включает элементы интерфейса для отображения данных о пользователе и его проектах. Позволяет пользователям просматривать информацию о сотрудниках.
- ReportFragment.kt: Фрагмент для генерации отчетов, позволяющий выбрать формат экспорта (PDF, Word, Excel). Включает элементы интерфейса для выбора параметров отчета и его генерации. Позволяет пользователям создавать и экспортировать отчеты.
- ReportActivity.kt: Активность для отображения отчетов и их экспорта. Включает элементы интерфейса для просмотра отчетов и их сохранения. Обеспечивает удобный доступ к сгенерированным отчетам.
- LoginViewModel.kt: ViewModel для управления состоянием экрана авторизации. Включает методы для обработки ввода данных и выполнения

процесса авторизации. Обеспечивает связь между интерфейсом и логикой авторизации.

- MainViewModel.kt: ViewModel для управления состоянием главного экрана. Включает методы для обработки навигации и взаимодействия с основными функциями приложения. Обеспечивает связь между интерфейсом и логикой главного экрана.
- ReportViewModel.kt: ViewModel для управления состоянием экрана отчетов. Включает методы для обработки выбора параметров отчета и его генерации. Обеспечивает связь между интерфейсом и логикой отчетов.
- LoginViewModelFactory.kt: Фабрика для создания экземпляра LoginViewModel. Обеспечивает инициализацию ViewModel с необходимыми зависимостями.
- ProjectAdapter.kt: Адаптер для отображения списка проектов в ProjectListFragment. Включает методы для привязки данных о проектах к элементам интерфейса. Обеспечивает динамическое отображение списка проектов.

res/

- activity_login.xml: Layout для экрана авторизации, включающий поля для ввода логина и пароля, а также кнопку для входа. Обеспечивает интерфейс для процесса авторизации.
- activity_main.xml: Layout для главного экрана, включающий элементы навигации и основные функции. Обеспечивает централизованный доступ ко всем функциям приложения.
- fragment_project_list.xml: Layout для списка проектов, включающий элементы для отображения информации о проектах. Обеспечивает интерфейс для просмотра и управления проектами.
- fragment_user_details.xml: Layout для деталей пользователя, включающий элементы для отображения информации о пользователе и его участии в проектах. Обеспечивает интерфейс для просмотра данных о сотрудниках.

- fragment_report.xml: Layout для экрана отчетов, включающий элементы для выбора формата экспорта. Обеспечивает интерфейс для генерации и экспорта отчетов.
- activity_report.xml: Layout для активности отчетов, включающий элементы для отображения отчетов. Обеспечивает интерфейс для просмотра и сохранения отчетов.
- item_project.xml: Layout для элемента списка проектов, включающий элементы для отображения информации о проекте. Обеспечивает интерфейс для отображения данных о проекте в списке.
- strings.xml: Строковые ресурсы, включающие текстовые элементы интерфейса. Обеспечивает локализацию и управление текстом в приложении.
- colors.xml: Цветовые ресурсы, определяющие палитру приложения. Обеспечивает управление цветами интерфейса.
- styles.xml: Стили приложения, определяющие внешний вид элементов интерфейса. Обеспечивает единообразие и привлекательность интерфейса.
- main_menu.xml: Меню приложения, включающее элементы навигации. Обеспечивает доступ к основным функциям приложения через меню.

Настройки проекта (settings.gradle.kts)

- settings.gradle.kts: Настройки проекта, включающие конфигурацию модулей и зависимостей. Обеспечивает управление структурой проекта и его зависимостями.

Конечными пользователями информационной системы будут сотрудники «Авиационный комплекс им. С.В. Ильюшина», которые будут иметь доступ к необходимой информации в соответствии со своими ролями и полномочиями. Система будет предусматривать авторизацию, регистрацию и разделение пользователей по ролям для обеспечения безопасности данных.

Таким образом, разработка информационной системы является актуальной задачей, решение которой позволит повысить эффективность работы компании и качество обслуживания пользователей.

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

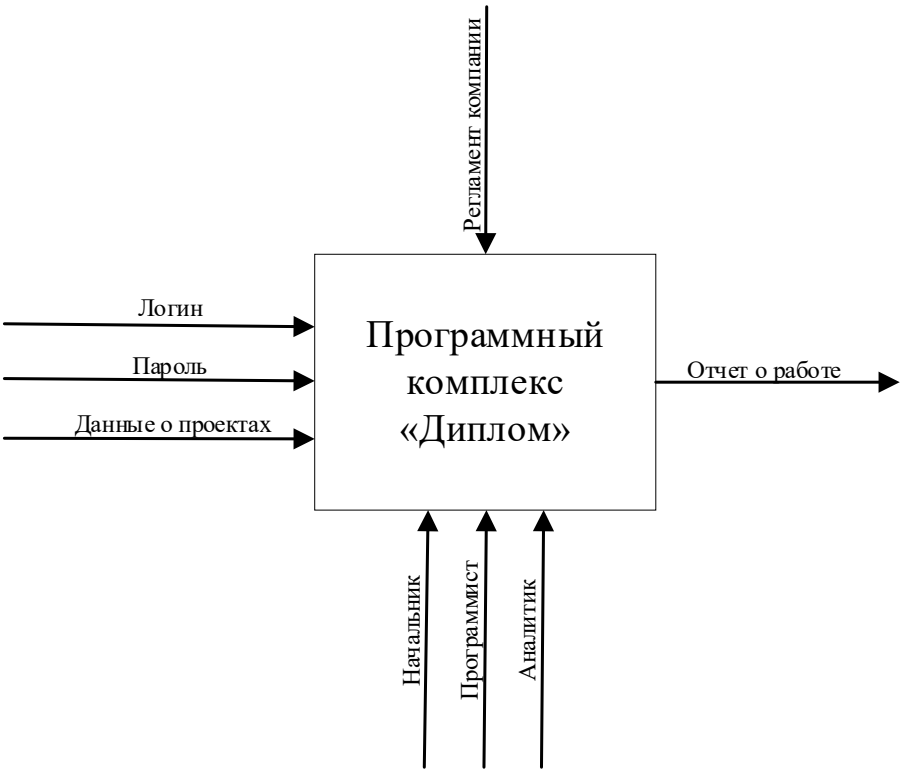

Рисунок 1.1 – Контекстная диаграмма IDEF0

1.2 Сравнительный анализ программ аналогов

Android Studio – это официальная интегрированная среда разработки (IDE) для создания приложений под Android. Она предоставляет широкий набор инструментов для разработки, отладки и тестирования мобильных приложений. Android Studio поддерживает языки программирования Kotlin и Java, что позволяет разработчикам выбирать наиболее подходящий язык для их проекта. Визуальный редактор макетов позволяет создавать и редактировать пользовательские интерфейсы с помощью перетаскивания элементов, что упрощает процесс разработки. Встроенные инструменты для отладки включают возможность установки точек останова и просмотра значений переменных, что делает процесс отладки более удобным. Эмуляторы устройств позволяют запускать и тестировать приложения на виртуальных устройствах с различными конфигурациями, обеспечивая тестирование в разных условиях. Интеграция с системой контроля версий, такой как Git, позволяет управлять кодом и отслеживать изменения. Широкий выбор плагинов и расширений позволяет расширять функциональность IDE в зависимости от потребностей разработчика.

IntelliJ IDEA от JetBrains является мощной IDE для Java и других языков программирования, которая также поддерживает разработку Android-приложений через плагин. IntelliJ IDEA предлагает улучшенные возможности рефакторинга кода, анализа зависимостей и интеграции с различными инструментами разработки. Однако, для работы с Android-приложениями требуется установка дополнительных плагинов, что может быть менее удобно по сравнению с Android Studio, где все необходимые инструменты доступны «из коробки».

Eclipse – это еще одна популярная IDE, которая поддерживает разработку Android-приложений через плагин ADT (Android Development Tools). Eclipse предоставляет гибкую и расширяемую платформу для разработки, но может быть менее удобной для новичков по сравнению с Android Studio. Eclipse требует установки и настройки плагинов для работы с Android, что может занять дополнительное время.

Visual Studio Code от Microsoft – это легкая и быстрая IDE, которая поддерживает разработку Android-приложений через расширения. Visual Studio Code предлагает мощные возможности редактирования кода, интеграцию с Git и широкий выбор плагинов. Однако, для полноценной разработки Android-приложений потребуется установка дополнительных расширений, что может быть менее удобно по сравнению с Android Studio.

MySQL Workbench – это унифицированное визуальное средство для разработчиков баз данных, администраторов и архитекторов. Оно предоставляет инструменты для проектирования, разработки, администрирования и управления базами данных MySQL. MySQL Workbench позволяет создавать и редактировать ER-диаграммы (диаграммы «сущность-связь»), что упрощает процесс проектирования базы данных. SQL-редактор с поддержкой автодополнения и подсветки синтаксиса позволяет эффективно писать и выполнять SQL-запросы. Инструменты для администрирования баз данных включают управление пользователями, правами доступа, резервным копированием и восстановлением данных. Мониторинг производительности позволяет отслеживать и анализировать производительность базы данных, что помогает в оптимизации работы системы. Инструменты для миграции данных позволяют переносить данные из других систем управления базами данных в MySQL, что упрощает процесс перехода на новую систему.

phpMyAdmin – это веб-интерфейс для управления базами данных MySQL. PhpMyAdmin предоставляет удобный способ выполнения SQL-запросов, управления таблицами и данными через веб-браузер. Он поддерживает импорт и экспорт данных, а также управление пользователями и правами доступа. PhpMyAdmin доступен из любого браузера, что делает его удобным для удаленного управления базами данных. Однако, для сложных задач проектирования и администрирования баз данных phpMyAdmin может быть менее удобен по сравнению с MySQL Workbench.

DBeaver — это универсальная база данных клиент, поддерживающий работу с различными системами управления базами данных, включая MySQL. DBeaver предлагает мощные возможности для визуализации данных, редактирования SQL-запросов и администрирования баз данных. DBeaver поддерживает широкий спектр

функций, включая ER-диаграммы, мониторинг производительности и миграцию данных. Однако, интерфейс DBeaver может быть сложнее для освоения по сравнению с MySQL Workbench.

HeidiSQL — это легкий и быстрый клиент для управления базами данных MySQL, MariaDB и другими. HeidiSQL предоставляет удобный интерфейс для выполнения SQL-запросов, управления таблицами и данными, а также импорта и экспорта данных. HeidiSQL прост в использовании и подходит для выполнения базовых задач администрирования баз данных. Однако, для сложных задач проектирования и мониторинга производительности HeidiSQL может быть менее функционален по сравнению с MySQL Workbench.

Android Studio специально разработана для создания Android-приложений и предоставляет все необходимые инструменты «из коробки». IntelliJ IDEA требует установки дополнительных плагинов для поддержки разработки Android, но предлагает более мощные возможности рефакторинга и анализа кода. Обе IDE могут быть ресурсоемкими, но Android Studio оптимизирована для работы с Android-проектами, что может обеспечить лучшую производительность в некоторых случаях. IntelliJ IDEA имеет более широкий выбор плагинов, что позволяет расширить функциональность IDE для различных задач разработки.

MySQL Workbench предоставляет настольное приложение с богатым набором инструментов для проектирования и администрирования баз данных. PhpMyAdmin является веб-интерфейсом, что делает его доступным из любого браузера, но может быть менее удобным для сложных задач. MySQL Workbench предлагает более широкий набор инструментов для визуализации данных, мониторинга производительности и миграции данных. PhpMyAdmin ограничен в этих аспектах, но предоставляет удобные возможности для управления таблицами и данными. PhpMyAdmin может быть более простым в освоении для новичков благодаря своему интуитивно понятному веб-интерфейсу. MySQL Workbench требует установки и настройки на локальном компьютере, но предлагает более мощные возможности для опытных пользователей.

Android Studio и MySQL Workbench предоставляют специализированные инструменты для разработки Android-приложений и управления базами данных MySQL соответственно, что делает их отличным выбором для большинства задач.

Однако, если требуется более гибкая и расширяемая среда разработки или веб-доступ к базе данных, стоит рассмотреть альтернативы, такие как IntelliJ IDEA и phpMyAdmin.

ГЛАВА 2 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данной главе представлены результаты практической работы в должности программиста. Была выполнена реализация всех заданий практики.

2.1 Разработка приложений обработки данных на ассемблере (задание №1)

Задание выполнено в среде SASM. Ниже приведен программный код:

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В.
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Ассемблер
Задание: Дан двумерный массив слов. Сформировать одномерный массив, каждый элемент которого равен сумме элементов соответствующей строки.

Использованные секции:
.data – секция данных, содержащая инициализированные переменные;
.text– секция кода, содержащая исполняемые инструкции.

Переменные:
A – двумерный массив слов;
b – одномерный массив, инициализированный нулями, для хранения сумм элементов строк массива A.
ROW – константа, определяющая количество строк в массиве A.
COL – константа, определяющая количество столбцов в массиве A.

Функции:

Main – основная функция программы, выполняющая вычисление сумм элементов строк массива A и сохранение результатов в массив B.

```
.intel_syntax noprefix
.data
   A: .word 1, 2, 3, 4
      .word 8, 7, 6, 5
      .word 9, 10, 11, 12
   .equ ROW, 3
   .equ COL, 4
   # инициализирован нулями, 3 элемента по 2 байта каждый
   B: .zero 2*ROW
.text
.global main
main:
   mov %rbp, %rsp #for correct debugging
   mov ecx, ROW
   # вычисление смещения строк в массиве A
   mov ebx, 0
   # индекс для массива B
   mov edi, 0

L1: mov edx, ecx
   mov ecx, COL
   # индекс для доступа к элементам строки в массиве A
   mov esi, 0
   # хранение текущей суммы элементов строки
   mov ax, 0

L2:
```

```
# добавляет значение элемента массива A по
# индексу esi и смещению ebx к регистру ax
add ax, A[esi][ebx]
# сохраняет текущую сумму в регистре
mov B[edi],ax
add esi, 2
loop L2

# переход к следующему элементу массива B
add edi, 2
# переход к следующему элементу массива A
add ebx, 2*COL
mov ecx, edx
loop L1

xorq  %rax, %rax
```

2.2 Разработка приложений обработки данных на языке программирования высокого уровня (технология ООП) (задание №2)

2.2.1 Листинг файла Clock

Задание выполнено в среде Eclipce. Ниже приведен программный код:

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
Clock – класс, представляющий объект часов с полями для времени и даты.

Переменные:

Id – уникальный идентификатор объекта часов;

Time – время, связанное с объектом часов;

Date – дата, связанная с объектом часов.

Функции:

Clock() – Конструктор по умолчанию;

Clock(LocalTime time, LocalDate date) – конструктор для инициализации объекта часов с заданными значениями времени и даты;

Long getId() – метод для получения уникального идентификатора объекта часов;

void setId(Long id) – метод для установки уникального идентификатора объекта часов;

LocalTime getTime() – метод для получения времени объекта часов;

void setTime(LocalTime time) – метод для установки времени объекта часов;

LocalDate getDate() – метод для получения даты объекта часов;

void setDate(LocalDate date) – метод для установки даты объекта часов.

```
package com.example.laba4;

import java.time.LocalDate;
import java.time.LocalTime;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Clock {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
private Long id;
private LocalTime time;
private LocalDate date;

public Clock() {}

public Clock(LocalTime time, LocalDate date) {
    this.time = time;
    this.date = date;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public LocalTime getTime() {
    return time;
}

public void setTime(LocalTime time) {
    this.time = time;
}

public LocalDate getDate() {
    return date;
}

public void setDate(LocalDate date) {
```

```
        this.date = date;
    }
}
```

2.2.2 Листинг файла ClockController

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
ClockController – класс, управляющий обработкой HTTP-запросов для работы с объектами часов.

Переменные:
clockRepository – репозиторий для работы с объектами часов.

Функции:

showClockList(Model model) – метод для отображения списка всех объектов часов;
showAddClockForm(Clock clock) – метод для отображения формы добавления нового объекта часов;
addClock(Clock clock, BindingResult result, Model model) – метод для добавления нового объекта часов;
deleteClock(@PathVariable Long id, Model model) – метод для удаления объекта часов по его идентификатору.

```
package com.example.laba4;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
public class ClockController {

    private final ClockRepository clockRepository;

    @Autowired
    public ClockController(ClockRepository clockRepository) {
        this.clockRepository = clockRepository;
    }

    @GetMapping("/clocks")
    public String showClockList(Model model) {
        model.addAttribute("clocks", clockRepository.findAll());
        return "clocks";
```

```
    }

    @GetMapping("/addClock")
    public String showAddClockForm(Clock clock) {
        return "add-clock";
    }

    @PostMapping("/addClock")
    public String addClock(Clock clock, BindingResult result, Model model) {
        if (result.hasErrors()) {
            return "add-clock";
        }
        clockRepository.save(clock);
        return "redirect:/clocks";
    }

    @GetMapping("/deleteClock/{id}")
    public String deleteClock(@PathVariable Long id, Model model) {
        clockRepository.deleteById(id);
        return "redirect:/clocks";
    }
}
```

2.2.3 Листинг файла ClockPage

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
ClockPage – класс для работы с веб-страницей, отображающей объекты часов.

Переменные:
webDriver – драйвер для управления браузером;
header – элемент заголовка страницы.

Функции:

getHeaderText() – метод для получения текста заголовка страницы;
getClockEntries() – метод для получения списка всех записей о часах на странице.

```
package com.example.laba4;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

import java.util.List;
import java.util.stream.Collectors;

public class ClockPage {
    private WebDriver webDriver;

    @FindBy(tagName = "h2")
    private WebElement header;

    public ClockPage(WebDriver webDriver) {
        this.webDriver = webDriver;
        PageFactory.initElements(webDriver, this);
    }

    public String getHeaderText() {
        return header.getText();
    }

    public List<String> getClockEntries() {
        return webDriver.findElements(By.cssSelector("tbody tr"))
                .stream()
```

```
            .map(row -> row.findElements(By.tagName("td")).stream()
                .map(WebElement::getText)
                .collect(Collectors.joining(" ")))
            .collect(Collectors.toList());
    }
}
```

2.2.4 Листинг файла ClockRepository

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
ClockRepository – интерфейс репозитория для работы с объектами часов.

```
package com.example.laba4;

import org.springframework.data.repository.CrudRepository;

public interface ClockRepository extends CrudRepository<Clock, Long> {
```

```
}
```

2.2.5 Листинг файла Demo4Application

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
Demo4Application – основной класс приложения, содержащий точку входа.

Функции:
main(String[] args) – метод для запуска приложения.

```
package com.example.laba4;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication
@EnableJpaRepositories(basePackages = "com.example.laba4")
public class Demo4Application {

    public static void main(String[] args) {
        SpringApplication.run(Demo4Application.class, args);
    }
}
```

2.2.6 Листинг файла main.css

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
main.css – файл стилей для оформления веб-страниц.

```
h2 {
   font-family: sans-serif;
   font-size: 3.5em;
   text-transform: uppercase;
}
```

```
p {
    font-family: sans-serif;
}

table {
    border: 1px solid #eee;
    table-layout: fixed;
    width: 100%;
    margin-bottom: 20px;
}

table th {
    font-weight: bold;
    padding: 5px;
    background: #efefef;
    border: 1px solid #dddddd;
}

table td {
    padding: 5px 10px;
    border: 1px solid #eee;
    text-align: left;
}

table tbody tr:nth-child(odd) {
    background: #fff;
}

table tbody tr:nth-child(even) {
    background: #F7F7F7;
}
```

```
.button-style {
    display: inline-block;
    padding: 10px 15px;
    background: #f2f2f2;
    border: 1px solid #000;
    color: #333;
    text-decoration: none;
    text-align: center;
}

input[type=submit] {
    padding: 10px 15px;
    background: #f2f2f2;
    border: 1px solid #000;
    color: #333;
    text-decoration: none;
    text-align: center;
}

.button-style:hover, .button-style:focus {
    background: #e2e2f2;
}
```

2.2.7 Листинг файла add-clock.html

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
add-clock.html – HTML-файл для формы добавления нового объекта часов.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <title>Добавить часы</title>
```

```
    <link th:href="@{/main.css}" rel="stylesheet" />
</head>
<body>
<form method="post" th:action="@{/addClock}" th:object="${clock}">
    <div>
        <label for="time">Время</label>
        <input type="time" th:field="*{time}">
    </div>
    <div>
        <label for="date">Дата</label>
        <input type="date" th:field="*{date}">
    </div>
    <div>
        <input type="submit" value="Сохранить">
    </div>
</form>
</body>
</html>
```

2.2.8 Листинг файла clocks.html

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
clocks.html – HTML-файл для отображения списка всех объектов часов.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <title>Clock List</title>
```

```
    <link th:href="@{/main.css}" rel="stylesheet" />
</head>
<body>
<h2>Часы</h2>
<table>
    <thead>
    <tr>
        <th>Время</th>
        <th>Дата</th>
        <th>Удалить</th>
    </tr>
    </thead>
    <tbody>
    <tr th:each="clock : ${clocks}">
        <td th:text="${clock.time}"></td>
        <td th:text="${clock.date}"></td>
        <td><a th:href="@{/deleteClock/{id}(id=${clock.id})}" class="button-
style">Удалить</a></td>
    </tr>
    </tbody>
</table>
<div>
    <a href="/addClock" class="button-style">Добавить</a>
</div>
</body>
</html>
```

2.2.9 Листинг файла ClockPageTest

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
ClockPageTest – класс для тестирования веб-страницы с объектами часов.

Переменные:
driver – драйвер для управления браузером;
clockPage – объект страницы с часами;
clockRepository – репозиторий для работы с объектами часов.

Функции:

setUp() – метод для подготовки тестовых данных;

tearDown() – метод для очистки тестовых данных;

testHeaderText() – метод для проверки текста заголовка страницы;

testClockEntries() – метод для проверки наличия записей о часах на странице.

```
package com.example.laba4;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit.jupiter.SpringJUnitConfig;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import java.time.LocalDate;
import java.time.LocalTime;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;

import static org.assertj.core.api.Assertions.assertThat;

@SpringBootTest
@SpringJUnitConfig
@Transactional(propagation = Propagation.NOT_SUPPORTED)
public class ClockPageTest {
```

```
    private WebDriver driver;
    private ClockPage clockPage;

    @Autowired
    private ClockRepository clockRepository;

    @BeforeEach
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "F:\\chromedriver-win64\\chromedriver.exe");

        ChromeOptions options = new ChromeOptions();
        options.addArguments("--headless");
        driver = new ChromeDriver(options);

        driver.get("http://localhost:8080/clocks");
        clockPage = new ClockPage(driver);

        Clock testClock = new Clock(LocalTime.now(), LocalDate.now());
        clockRepository.save(testClock);

        List<Clock> allClocks = StreamSupport.stream(clockRepository.findAll().spliterator(), false)
                .collect(Collectors.toList());

        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```

```
    @AfterEach
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
        clockRepository.deleteAll();
    }

    @Test
    public void testHeaderText() {
        assertThat(clockPage.getHeaderText()).contains("ЧАСЫ");
    }

    @Test
    public void testClockEntries() {
        List<String> clockEntries = clockPage.getClockEntries();
        assertThat(clockEntries).isNotEmpty();
    }
}
```

2.2.10 Листинг файла pom.xml

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Java

Задание:
Создать класс «Часы». Список полей класса:

- ключ;
- время;
- дата.

Выполнить объектно-реляционное отображение с помощью JPA. Создать классы «Spring Repository». Создать Web-приложение и выполнить:

- добавление объектов;
- удаление объектов;
- обновление объектов;
- вывод всех объектов.

Использованные классы:
pom.xml – файл конфигурации Maven для управления зависимостями проекта.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.2</version>
    <relativePath/>
</parent>
<groupId>com.example</groupId>
<artifactId>laba4</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo-4</name>
<description>Demo project for Spring Boot</description>
<properties>
    <java.version>17</java.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
```

```
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.28.1</version>
    </dependency>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-chrome-driver</artifactId>
        <version>4.28.1</version>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.9.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.assertj</groupId>
        <artifactId>assertj-core</artifactId>
        <version>3.24.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
```

```
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>
```

Ниже приведена диаграмма классов, как компонент проектной документации:

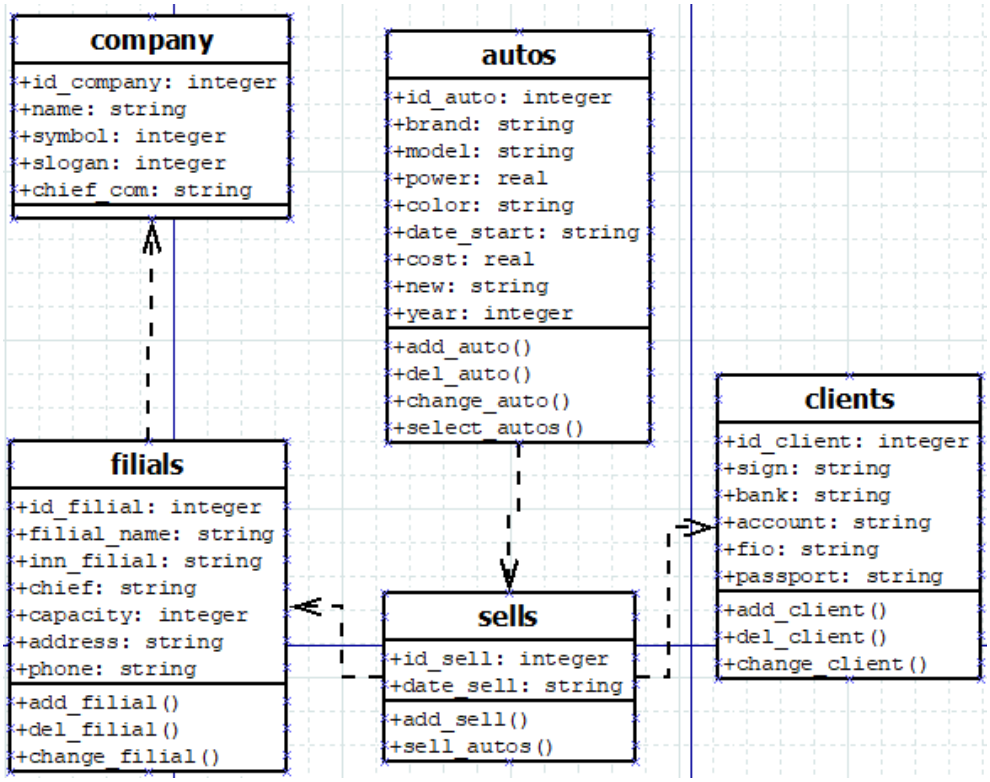

Рисунок 2.1 – Диаграмма классов

2.3 Проектирование, разработка и отладка программ (задание №3)

Задание выполнено в среде Android Studio. Ниже приведен программный код и изображение визуального приложения:

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

Название: Диплом

Разработал: Дюрягина И. В

Группа: ТИП-82

Дата и номер версии: 20.03.2025 v1.0

Язык: Kotlin

Задание:

Создать кроссплатформенное мобильное приложение на Kotlin в Android Studio с использованием MySQL Workbench для учета проектов отдела разработки и внедрения корпоративных информационных систем.

Создать дерево проектов с полями:

- Название;
- Аналитики;
- Разработчики;
- Даты;
- ФИО.

Приложение должно предусматривать три роли, их авторизацию, регистрацию и хранение логина и пароля в базе данных:

- Аналитик;
- Программист;
- Начальник.

При нажатии на аналитиков – выдается информация обо всех аналитиках, аналогично с начальником и программистами.

При нажатии на конкретного аналитика выдаются сроки проектов, где он принимал участие, их названия и т.д., аналогично с начальником и программистами.

Приложение должно предусматривать отчет по работе отдела, сводки, премии в формате электронного документа (PDF, Word, Excel).

Использованные классы:
LoginActivity – класс активности для экрана авторизации;
LoginViewModel – класс для управления логикой авторизации;
LoginViewModelFactory – фабрика для создания экземпляра LoginViewModel;
UserRepository – класс для работы с данными пользователей;
ActivityLoginBinding – класс для привязки элементов интерфейса.

Переменные:
binding – объект для доступа к элементам интерфейса активности;
viewModel – объект ViewModel для управления логикой авторизации.

Функции:
onCreate(Bundle) – метод инициализации активности и обработки UI-элементов;
setOnClickListener для loginButton – обработчик нажатия кнопки входа;
login(String, String, Callback) – метод ViewModel для выполнения авторизации.

```
package com.example.diplom.android.ui

import android.os.Bundle
import android.view.View
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider
import com.example.diplom.databinding.ActivityLoginBinding
```

```
import com.example.diplom.repos.UserRepository
import com.example.diplom.android.viewmodel.LoginViewModel
import com.example.diplom.android.viewmodel.LoginViewModelFactory

class LoginActivity : AppCompatActivity() {

    private lateinit var binding: ActivityLoginBinding

    // Создаем ViewModel вручную
    private lateinit var viewModel: LoginViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Создаем ViewModel с помощью ViewModelProvider
        viewModel = ViewModelProvider(
            this,
            LoginViewModelFactory(UserRepository())
        ).get(LoginViewModel::class.java) // Указываем тип ViewModel явно

        // Обработка нажатия на кнопку входа
        binding.loginButton.setOnClickListener {
            val username = binding.usernameEditText.text.toString()
            val password = binding.passwordEditText.text.toString()

            // Показать ProgressBar
            binding.progressBar.visibility = View.VISIBLE

            // Вызов ViewModel для авторизации
            viewModel.login(username, password) { success, errorMessage ->
```

```
            // Скрыть ProgressBar
            binding.progressBar.visibility = View.GONE

            if (success) {
                // Успешная авторизация
                binding.errorTextView.visibility = View.GONE
                // Переход на следующий экран
            } else {
                // Показать ошибку
                binding.errorTextView.text = errorMessage
                binding.errorTextView.visibility = View.VISIBLE
            }
        }
    }
}
```

2.4 Разработка кода программного продукта на основе готовых спецификаций на уровне модуля (задание №4)

Задание выполнено в среде Android Studio. Ниже приведен программный код:

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

Название: Диплом.

Разработал: Дюрягина И. В.

Группа: ТИП-82.

Дата и номер версии: 20.03.2025 v1.0.

Язык: Kotlin.

Задание:

Создать кроссплатформенное мобильное приложение на Kotlin в Android Studio с использованием MySQL Workbench для учета проектов отдела разработки и внедрения корпоративных информационных систем.

Создать дерево проектов с полями:

- Название;
- Аналитики;
- Разработчики;
- Даты;
- ФИО.

Приложение должно предусматривать три роли, их авторизацию, регистрацию и хранение логина и пароля в базе данных:

- Аналитик;
- Программист;
- Начальник.

При нажатии на аналитиков – выдается информация обо всех аналитиках, аналогично с начальником и программистами.

При нажатии на конкретного аналитика выдаются сроки проектов, где он принимал участие, их названия и т.д., аналогично с начальником и программистами.

Приложение должно предусматривать отчет по работе отдела, сводки, премии в формате электронного документа (PDF, Word, Excel).

Использованные классы:
MainActivity – класс активности для главного экрана приложения;
MainViewModel – класс для управления логикой загрузки проектов;
MainViewModelFactory – фабрика для создания экземпляра MainViewModel;
ProjectRepository – класс для работы с данными проектов;
ActivityMainBinding – класс для привязки элементов интерфейса.

Переменные:
binding – объект для доступа к элементам интерфейса активности;
viewModel – объект ViewModel для управления логикой загрузки проектов.

Функции:
onCreate(Bundle) – метод инициализации активности и обработки UI-элементов;
loadProjects(Callback) – метод ViewModel для загрузки списка проектов.

```
package com.example.diplom.android.ui

import android.os.Bundle
import android.view.View
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider
import com.example.diplom.databinding.ActivityMainBinding
import com.example.diplom.repos.ProjectRepository
```

```
import com.example.diplom.android.viewmodel.MainViewModel
import com.example.diplom.android.viewmodel.MainViewModelFactory

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private lateinit var viewModel: MainViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        viewModel = ViewModelProvider(
            this,
            MainViewModelFactory(ProjectRepository())
        ).get(MainViewModel::class.java)

        binding.progressBar.visibility = View.VISIBLE

        viewModel.loadProjects { projects ->
            binding.progressBar.visibility = View.GONE
        }
    }
}
```

2.5 Тестирование и оптимизация программных модулей (задание №5)

Оптимизация программного кода – это модификация программы с целью улучшения отдельных характеристик без изменения функциональности. Способы: алгоритмические (ручной способ): математические методы, логические методы; машинно-зависимые (автоматический способ): ассемблерные вставки, модификация алгоритма кодогенератора компилятора

Приемы оптимизация по быстродействию:

1. Замена алгоритмов на более быстрые (часто простой алгоритм показывает низкую производительность по сравнению с более сложным).
2. Пример: замена пузырьковой сортировки массива на быструю сортировку.
3. Хранение данных в «быстрой» памяти (нужно, чтобы интенсивно обрабатываемые блоки данных целиком умещались в КЭШ-память).
4. Разгрузка участков итераций (вынос инвариантных по результату выполнения выражений из тела цикла).
5. Анализ циклов:
   - цикл с предусловием выполняется дольше (на 20–30 %);
   - при программировании вложенных циклов по следует делать цикл с наибольшим числом повторений самым внутренним, а цикл с наименьшим числом повторений – самым внешним;
   - объединение циклов (если это возможно);
   - разворот цикла.

Пример оптимизации кода с применением способа разворота цикла:

1. Хранение данных в быстрой памяти

Неоптимизированный код:

```
private lateinit var viewModel: LoginViewModel
override fun onCreate(...) {
   ...
   viewModel = ViewModelProvider(...).get(...)
}
```

Оптимизированный код:

```
private val viewModel: LoginViewModel by lazy {
   ViewModelProvider(...)[...]
}
```

2. Разгрузка участков итераций

Неоптимизированный код:

```
binding.loginButton.setOnClickListener {
   val username = ...
   val password = ...
   // 15 строк кода внутри
}
```

Оптимизированный код:

```
private fun setupLoginButton() {
   binding.loginButton.setOnClickListener {
      val (u, p) = getCredentials()
      if (!validate(u, p)) return@setOnClickListener
      // ...остальное
   }
}
```

3. Использование быстрых структур данных

Неоптимизированный код:

```
val username = binding.usernameEditText.text.toString()
val password = binding.passwordEditText.text.toString()
```

Оптимизированный код:

```
private fun getCredentials() = Pair(
   binding.usernameEditText.text.toString(),
   binding.passwordEditText.text.toString()
)
```

2.6 Разработка компонентов проектной и технической документации (задание №6)

Тема: «Разработка мобильного приложения для учета проектов отдела разработки и внедрения корпоративных информационных систем»
Название: Диплом
Разработал: Дюрягина И. В
Группа: ТИП-82
Дата и номер версии: 20.03.2025 v1.0
Язык: Kotlin

Задание:
Создать кроссплатформенное мобильное приложение на Kotlin в Android Studio с использованием MySQL Workbench для учета проектов отдела разработки и внедрения корпоративных информационных систем.

Создать дерево проектов с полями:

- Название;
- Аналитики;
- Разработчики;
- Даты;
- ФИО.

Приложение должно предусматривать три роли, их авторизацию, регистрацию и хранение логина и пароля в базе данных:

- Аналитик;
- Программист;
- Начальник.

При нажатии на аналитиков – выдается информация обо всех аналитиках, аналогично с начальником и программистами.

При нажатии на конкретного аналитика выдаются сроки проектов, где он принимал участие, их названия и т.д., аналогично с начальником и программистами.

Приложение должно предусматривать отчет по работе отдела, сводки, премии в формате электронного документа (PDF, Word, Excel).

Использованные классы:
ReportActivity – класс активности для экрана генерации отчетов;
ReportViewModel – класс для управления логикой формирования отчетов;
ReportViewModelFactory – фабрика для создания экземпляра ReportViewModel;
ReportRepository – класс для работы с данными отчетов;
ActivityReportBinding – класс для привязки элементов интерфейса.

Переменные:
binding – объект для доступа к элементам интерфейса активности;
viewModel – объект ViewModel для управления логикой генерации отчетов.

Функции:
onCreate(Bundle) – метод инициализации активности и обработки UI-элементов;
generateReport(Int, Callback) – метод ViewModel для формирования отчета по проекту.

```
package com.example.diplom.android.ui

import android.os.Bundle
import android.view.View
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.ViewModelProvider
import com.example.diplom.databinding.ActivityReportBinding
import com.example.diplom.repos.ReportRepository
import com.example.diplom.android.viewmodel.ReportViewModel
```

```
import com.example.diplom.android.viewmodel.ReportViewModelFactory

class ReportActivity : AppCompatActivity() {

    private lateinit var binding: ActivityReportBinding
    private lateinit var viewModel: ReportViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityReportBinding.inflate(layoutInflater)
        setContentView(binding.root)

        viewModel = ViewModelProvider(
            this,
            ReportViewModelFactory(ReportRepository())
        ).get(ReportViewModel::class.java)

        binding.progressBar.visibility = View.VISIBLE
        val projectId = intent.getIntExtra("projectId", 1)

        viewModel.generateReport(projectId) { reportContent ->
            binding.progressBar.visibility = View.GONE
            binding.reportTextView.text = reportContent
        }
    }
}
```

2.7 Разработка мобильного приложения (задание №7)

Тема: «Разработка мобильного приложения формата «Список-детальная страница» для отображения данных о футбольных матчах».

Название: Лабораторная работа 3

Разработал: Дюрягина И. В.

Группа: ТИП-82

Дата и номер версии: 20.03.2025 v1.0

Язык: Kotlin.

Задание:

Необходимо разработать приложение формата Список-детальная страница. Приложение должно состоять из двух экранов. На первом экране располагается список объектов с краткой информацией по каждому. При клике на элемент списка должен происходить переход на экран с детальной информацией об этом объекте. Информацию необходимо брать с удаленного сервиса(публичного апи)

Используемое апи должно быть уникальным для каждого студента. Необходимо указать ссылку на используемое апи в общем файле: https://docs.google.com/spreadsheets/d/1zcukSKsFcQt361tpjPR6pfpJrKV6ME3TICmWUAJZY5g/edit?gid=0#gid=0

Требования

- Минимум 2 экрана
- Минимум один список(RecyclerView)
- В списке должно отображаться минимум 3 поля для каждого объекта
- На втором экране отображается минимум 7 полей
- Используемое апи должно быть уникальным для каждого студента

Использованные классы:

DetailActivity – активность для отображения детальной информации о матче;

Match – модель данных матча;

Goal – модель данных о голе.

Переменные:
match – объект матча, переданный из предыдущего экрана;
dateTextView – текстовое поле для отображения даты матча;
leagueTextView – текстовое поле для отображения названия лиги;
groupNameTextView – текстовое поле для отображения названия группы;
team1Name – текстовое поле для отображения названия первой команды;
team2Name – текстовое поле для отображения названия второй команды;
matchScore – текстовое поле для отображения счета матча;
matchStatus – текстовое поле для отображения статуса матча;
stadiumTextView – текстовое поле для отображения названия стадиона;
team1Logo – изображение логотипа первой команды;
team2Logo – изображение логотипа второй команды;
team1Container – контейнер для списка голов первой команды;
team2Container – контейнер для списка голов второй команды;
currentScoreTeam1 – текущий счет первой команды (временная переменная для отображения голов);
currentScoreTeam2 – текущий счет второй команды (временная переменная для отображения голов);
goal – объект гола (используется в цикле отображения голов);
inflater – инфлейтер для создания элементов интерфейса;
goalView – элемент интерфейса для отображения гола;
ivIcon – иконка гола (гол/пенальти);
tvScorer – текстовое поле с информацией об авторе гола;
iconRes – ресурс иконки в зависимости от типа гола;
finalResult – результат матча (используется для отображения счета).

Функции:
onCreate() – инициализация активности и обработка данных;
displayMatchDetails() – отображение основной информации о матче;
displayGoals() – отображение списка голов;

createGoalView() – создание элемента списка голов.

```
package com.example.laba3_2sem

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.bumptech.glide.Glide

class DetailActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_detail)

        val match = intent.getParcelableExtra<Match>("MATCH_DETAILS")

        if (match != null) {
            displayMatchDetails(match)
            displayGoals(match)
        } else {
            findViewById<TextView>(R.id.dateTextView).text = "Данные о матче
недоступны"
        }
    }

    private fun displayMatchDetails(match: Match) {
        val dateTextView: TextView = findViewById(R.id.dateTextView)
```

```
        val leagueTextView: TextView = findViewById(R.id.leagueTextView)
        val groupNameTextView: TextView = findViewById(R.id.groupNameTextView)
        val team1Name: TextView = findViewById(R.id.team1Name)
        val team2Name: TextView = findViewById(R.id.team2Name)
        val matchScore: TextView = findViewById(R.id.matchScore)
        val matchStatus: TextView = findViewById(R.id.matchStatus)
        val stadiumTextView: TextView = findViewById(R.id.stadiumTextView)
        val team1Logo: ImageView = findViewById(R.id.team1Logo)
        val team2Logo: ImageView = findViewById(R.id.team2Logo)

        dateTextView.text = match.matchDateTime
        leagueTextView.text = match.leagueName
        groupNameTextView.text = match.group.groupName
        team1Name.text = match.team1.shortName
        team2Name.text = match.team2.shortName
        stadiumTextView.text = match.location.locationStadium

        team1Logo.setImageResource(getTeamIconResId(match.team1.teamName))
        team2Logo.setImageResource(getTeamIconResId(match.team2.teamName))

        val finalResult = match.matchResults.find { it.resultName == "Endergebnis" }
        matchScore.text = finalResult?.let { "${it.pointsTeam1} : ${it.pointsTeam2}" } ?:
"N/A"
        matchStatus.text = finalResult?.resultName ?: "Статус неизвестен"
    }

    private fun displayGoals(match: Match) {
        val team1Container: LinearLayout = findViewById(R.id.team1GoalsContainer)
        val team2Container: LinearLayout = findViewById(R.id.team2GoalsContainer)

        team1Container.removeAllViews()
        team2Container.removeAllViews()
```

```
    var currentScoreTeam1 = 0
    var currentScoreTeam2 = 0

    match.goals.sortedBy { it.matchMinute }.forEach { goal ->
        when {
            goal.scoreTeam1 > currentScoreTeam1 -> {
                currentScoreTeam1 = goal.scoreTeam1
                team1Container.addView(createGoalView(goal))
            }
            goal.scoreTeam2 > currentScoreTeam2 -> {
                currentScoreTeam2 = goal.scoreTeam2
                team2Container.addView(createGoalView(goal))
            }
        }
    }
}

private fun createGoalView(goal: Goal): View {
    val inflater = LayoutInflater.from(this)
    val goalView = inflater.inflate(R.layout.item_goal, null)

    val ivIcon: ImageView = goalView.findViewById(R.id.ivGoalIcon)
    val tvScorer: TextView = goalView.findViewById(R.id.tvScorerInfo)

    val iconRes = when {
        goal.isPenalty -> R.drawable.penalty
        else -> R.drawable.goal
    }
    ivIcon.setImageResource(iconRes)

    tvScorer.text = "${goal.goalGetterName} (${goal.matchMinute}')"
```

```
        return goalView
    }
}
```

ЗАКЛЮЧЕНИЕ

По итогам практики получены следующие результаты:

1. Проведён анализ предметной области, а так же сравнительный анализ программ аналогов.
2. Реализована разработка спецификаций компонент программных систем, разработка кода программных модулей.
3. Выполнена отладка программных модулей с использованием специализированных программных средств.
4. Проведено тестирование и оптимизация программных модулей.
5. Освоена методология разработки компонент проектной и технической документации с использованием графических языков спецификаций.

По окончании практики был получен практический опыт написания программ на языках программирования низкого и высокого уровня.

СПИСОК ИСТОЧНИКОВ ИНФОРМАЦИИ

1. Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и защите информации» (с изменениями и дополнениями).
2. Глушаков С.В., Сурядный А.С. Разработка мобильных приложений на Android. — М.: АСТ, 2021.
3. Джошуа Блох. Java. Эффективное программирование. — СПб.: Питер, 2020.
4. Советов Б.Я., Цехановский В.В. Базы данных: теория и практика. — М.: Юрайт, 2022.
5. Иванов А.П. Современные подходы к проектированию кроссплатформенных приложений // Программирование и IT. — 2023. — № 4. — С. 45–52.
6. Петрова Е.К. Использование Kotlin в разработке мобильных приложений // Информационные технологии. — 2022. — № 7. — С. 34–39.
7. Официальная документация Android Studio [Электронный ресурс]. — Режим доступа: https://developer.android.com/studio (дата обращения: 20.03.2025).
8. Руководство по MySQL Workbench 8.0. — MySQL, 2023.
9. Public API для данных о футбольных матчах [Электронный ресурс]. — Режим доступа: https://www.football-data.org (дата обращения: 20.03.2025).
10. Документация Spring Framework [Электронный ресурс]. — Режим доступа: https://spring.io/projects/spring-framework (дата обращения: 20.03.2025).