федеральное государственное бюджетное образовательное учреждение высшего образования

«**Вологодский государственный университет**»

университетский колледж

Информационные системы и программирование

(наименование специальности)

**ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Код работы отчета 09.02.07.13.00.25.2025\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Код специальности, код УК, регистрационный номер по журналу, год

Руководитель Смирнова Валентина Александровна

(уч. степень, звание, должность. Ф.И.О)

Выполнил (а) обучающийся Ширкунов Артём Евгеньевич

(Ф.И.О)

Группа, курс ИСП 42–21, 4 курс

Дата сдачи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка по защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись преподавателя)

Вологда

2025 г

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc166530332)

[1 ЗНАКОМСТВО С ПРЕДПРИЯТИЕМ. ЗАКРЕПЛЕНИЕ РАБОЧЕГО МЕСТА 4](#_Toc166530333)

[1.1 Инструктаж по охране труда 4](#_Toc166530334)

[1.2 Описание предприятия и рабочего места 5](#_Toc166530335)

[2 ИЗУЧЕНИЕ ТЕХНИКО-ЭКОНОМИЧЕСКОЙ ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ 6](#_Toc166530336)

[3 АНАЛИЗ ИСПОЛЬЗУЕМЫХ СРЕДСТВ ОБРАБОТКИ ИНФОРМАЦИИ 9](#_Toc166530337)

[3.1 Описание предметной области 9](#_Toc166530334)

[3.2 Требования к программному обеспечению 10](#_Toc166530334)

[3.3 Обоснование выбора средств разработки 11](#_Toc166530334)

[4 АДМИНИСТРИРОВАНИЕ БАЗ ДАННЫХ, СУЩЕСТВУЮЩИХ НА ПРЕДПРИЯТИИ 14](#_Toc166530338)

[5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 15](#_Toc166530339)

[6 ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ 37](#_Toc166530340)

7 [СБОР И ОБОБЩЕНИЕ МАТЕРИАЛА ДЛЯ ДИПЛОМНОГО ПРОЕКТИРОВАНИЯ 39](#_Toc166530341)

ЗАКЛЮЧЕНИЕ 44

ВВЕДЕНИЕ

Производственная практика является неотъемлемым этапом в учебном процессе. Она помогает студентам применить учебные знания на производстве, развить профессиональные навыки и ознакомиться с реальными рабочими процессами.

В период с 21 апреля по 17 мая 2025 года была пройдена производственная практика в ООО «Макси инновации» на должности программиста в отделе программного обеспечения.

Целью преддипломной производственной практики является закрепление навыков разработки и поддержания компьютерных систем.

Задачи, которые необходимо решить для достижения поставленной цели:

* ознакомиться со структурой предприятия и пройти инструктаж по охране труда;
* изучить технико-экономическую характеристику предприятия;
* проанализировать используемые средства обработки информации;
* провести администрирование базы данных предприятия;
* разработать программные модули программного обеспечения;
* выполнить индивидуальное задание;
* собрать и обобщить материалы для дипломного проектирования.

1 ЗНАКОМСТВО С ПРЕДПРИЯТИЕМ. ЗАКРЕПЛЕНИЕ РАБОЧЕГО МЕСТА

1.1 Инструктаж по охране труда

В первую очередь в ходе прохождения производственной практики основное внимание было уделено изучению инструкций по охране труда и технике безопасности. В данных инструкциях подробно излагались предписания, регламентирующие разрешённые и запрещённые действия в рабочей зоне. Ключевые и наиболее значимые положения включали:

* требовалось поддерживать чистоту в рабочей зоне, избегая размещения посторонних объектов на поверхности стола;
* обязательной процедурой являлась проверка исправности электрооборудования перед его подключением к сети;
* категорически воспрещалось начинать трудовую деятельность при наличии признаков алкогольного или наркотического опьянения;
* курение в пределах рабочего пространства было строго запрещено;
* дисплей следовало располагать на 5 градусов ниже линии взгляда, соблюдая прямую плоскость расположения;
* оптимальное расстояние между глазами пользователя и монитором определялось диапазоном 60-80 см;
* требовалось активировать блокировку учётной записи при каждом временном отсутствии на рабочем месте;
* использование оборудования коллег без официального разрешения находилось под запретом;
* самостоятельный разбор технических устройств строго воспрещался;
* недопустимым считалось воздействие жидкостей на электротехнические устройства;
* запрещено подключать любые USB носители.

1.2 Описание предприятия и рабочего места

ООО «Макси инновации» осуществляет разработку программного обеспечения для поддержки деятельности розничной сети «Макси», которая включает более 70 магазинов в 18 городах России. Сеть занимается реализацией продукции, закупаемой у сторонних поставщиков, а также товаров собственного производства. Помимо розничной торговли, компания «Макси» ведёт деятельность в сфере строительства коммерческой и жилой недвижимости, а также занимается оптовыми поставками.

ООО «Макси инновации» создаёт разнообразные программные решения для автоматизации бизнес-процессов, включая торговую систему управления операциями, корпоративную социальную сеть с информационными материалами для сотрудников, клиентские мобильные приложения и сайт с функциями просмотра ассортимента, оформления заказов и применения персональных скидок, а также мобильную торговую платформу для терминалов сбора данных, используемых персоналом.

Практика проходила в бизнес-центре города Вологды по адресу улица Молодёжная, дом 25, кабинет 225. Для выполнения задач был предоставлен персональный компьютер с операционной системой Windows 10, доступ к корпоративной Wi-Fi-сети и учётная запись в системе GitLab.

2 ИЗУЧЕНИЕ ТЕХНИКО-ЭКОНОМИЧЕСКОЙ ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ

Исходные данные для дальнейших расчетов приведены в соответствии с таблицей.

Таблица 1 – Исходные данные

|  |  |  |
| --- | --- | --- |
| №п/п | Показатель | Значение |
| 1 | Стоимость ПК, руб. | 110000 |
| 2 | Срок полезного использования, лет | 3 |
| 3 | Число рабочих дней в году, дн. | 247 |
| 4 | Количество часов работы ПК в день, час | 6 |
| 5 | Заработная плата, в час | 305 |
| 6 | Начисления на заработную плату, % | 30,2 |
| 7 | Коэффициент накладных расходов | 0,1 |
| 8 | Плановая рентабельность, % | 25 |
| 9 | Затраты машинного времени на отладку программы, % | 10 |
| 10 | Расход электроэнергии в час, кВт | 0,5 |
| 11 | Стоимость кВт часа электроэнергии, руб. | 6,37 |

Определение затрат времени по стадиям разработки проекта представлено в соответствии с таблицей.

Таблица 2 – Определение затрат времени по стадиям разработки проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Стадия разработки проекта | Затраты времени, чел./часы | Поправочный коэффициент | Затраты времени с учетом коэффициента |
| 1. Разработка технического задания | 40 |  |  |
| 1.1. Затраты времени разработчика постановки задачи | 10 | 0,65 | 6,5 |
| 1.2. Затраты времени разработчика программного обеспечения | 30 | 0,35 | 10,5 |
| 2. Разработка эскизного проекта | 50 |  |  |
| 2.1. Затраты времени разработчика постановки задачи | 5 | 0,7 | 3,5 |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| 2.2. Затраты времени разработчика программного обеспечения | 40 | 0,3 | 12 |
| 3. Разработка технорабочего проекта | 410 |  |  |
| 3.1. Подготовка контента программного обеспечения | 70 | 0,2 | 14 |
| 3.2. Создание | 280 | 0,4 | 112 |
| 3.3. Оптимизация | 60 | 0,4 | 24 |
| 4. Внедрение | 25 |  |  |
| 4.1. Затраты времени разработчика постановки задачи | 15 | 0,5 | 7,5 |
| 4.2. Затраты времени разработчика программного обеспечения | 10 | 0,5 | 5 |
| Итого: | 525 |  | 195 |

Произведем расчет затрат.

С – стоимость электроэнергии рассчитывается по формуле (1):

|  |  |
| --- | --- |
| , руб., | (1) |

где – расход электроэнергии в час (по паспортным данным ПК), кВт;

– стоимость кВт часа электроэнергии.

руб.

Стоимость одного машинного часа рассчитывается по формуле (2):

|  |  |
| --- | --- |
| , руб., | (2) |

где – балансовая стоимость ПК;

– срок полезного использования;

– число рабочих дней в году;

– число часов работы ПК в день;

– стоимость электроэнергии.

= 27,93 руб.

Затраты, связанные с созданием данного программного продукта, рассчитываются по формуле (3):

|  |  |
| --- | --- |
| , руб., | (3) |

где затраты на создание программного продукта;

– время на составление программного продукта (чел/час);

– средняя заработная плата программиста (часовая);

– начисления на заработную плату;

– коэффициент накладных расходов;

– стоимость одного часа машинного времени;

– затраты машинного времени на отладку программы.

руб.

Договорная цена определяется по формуле (4):

|  |  |
| --- | --- |
| , руб., | (4) |

где R *–* плановая рентабельность.

104910,74 руб.

Расчеты, которые были проведены в ходе выполнения дипломного проекта, указывают на то, что проект является экономически обоснованным, так как договорная цена выше, чем затраты проекта.

3 АНАЛИЗ ИСПОЛЬЗУЕМЫХ СРЕДСТВ ОБРАБОТКИ ИНФОРМАЦИИ

3.1 Описание предметной области

Конечные пользователи разрабатываемого приложения — люди, изучающие иностранные языки или интересующиеся культурным погружением через чтение. Они сталкиваются с необходимостью преодолевать языковой барьер при работе с текстами: искать переводы незнакомых слов, анализировать сложные грамматические конструкции или адаптировать материал под свой уровень владения языком. Книги и тексты в приложении могут быть представлены в разных форматах и иметь метаданные, такие как уровень сложности, жанр или культурный контекст.

Каждый текст имеет структуру, которая может включать главы, разделы или отдельные предложения, требующие пояснений. Пользователь может выбирать фрагменты для мгновенного перевода, просматривать альтернативные значения слов или получать грамматические комментарии. Уровень сложности текста может динамически адаптироваться: например, заменять редкие слова на более распространённые или упрощать предложения, сохраняя смысл.

Существующие решения, такие как LingQ или Readlang, предоставляют базовые функции перевода и словарного запаса, но не объединяют их в единую систему с интерактивной адаптацией. Например, приложения вроде Beelinguapp предлагают параллельные тексты на двух языках, но не учитывают индивидуальный уровень пользователя. Другие платформы, например, Fluency Reader, фокусируются на скорочтении, игнорируя культурный контекст и гибкость настройки. Это создаёт пробел в нише приложений, где чтение становится комплексным инструментом обучения: сочетает мгновенную помощь, адаптацию контента и интерактивные упражнения для закрепления навыков.

Таким образом, ключевой задачей приложения будет интеграция искусственного интеллекта для анализа текстов, их динамической адаптации под пользователя и предоставления контекстных подсказок в реальном времени, сохраняя при этом погружение в иноязычную среду.

3.2 Требования к программному обеспечению

Приложение должно быть разработано для платформы Android с поддержкой версий начиная с API 24 (Android 7.0) и выше. Язык программирования — Kotlin, среда разработки — Android Studio. Приложение обязано корректно работать на устройствах с разрешением экрана от 480x800 (WVGA) до 1440x3088 (QHD+). Для хранения данных необходимо использовать SQLite. Необходима интеграция с искусственным интеллектом DeepSeek.

В приложении определена одна основная роль — пользователь, который может:

* Загружать свои книги файлом в формате PDF;
* Выделять слова и фразы в тексте для мгновенного перевода через DeepSeek. При выделении отображается перевод, грамматическая информация и примеры использования;
* Адаптировать текст перевода под свой уровень сложности: для A1 предложения сокращаются до 10 слов с заменой сложных терминов, для C2 добавляются культурные комментарии;
* Сохранять прогресс чтения;
* Создавать списки избранных слов и книг;
* Проходить упражнение для повторения добавленных слов;
* Настраивать интерфейс: выбирать темную или светлую тему, регулировать размер шрифта.

Интерфейс должен быть локализован на русский язык, поддерживать адаптивную верстку для всех указанных разрешений экрана. Обязательно наличие интуитивного меню с разделами «Библиотека», «Словарь», «Настройки».

Время отклика при переводе или адаптации текста не должно превышать 2 секунд. Приложение должно сохранять работоспособность при низком заряде батареи. Все пользовательские данные шифруются.

Приложение не поддерживает офлайн-перевод через сторонние API. Адаптация текстов доступна только для языков, указанных в метаданных приложения. Максимальный размер загружаемой книги — 100 МБ.

Приложение отправляет уведомления, если книга не открывалась более 3 дней. Кэширование переводов и комментариев обеспечивает офлайн-доступ. Руководство пользователя должно включать инструкции по всем функциям.

3.3 Обоснование выбора средств разработки

Для хранения данных будущей системы необходима база данных. Среди популярных решений для локального хранения данных в мобильных приложениях можно выделить SQLite, Realm, Room и Firebase Realtime Database.

SQLite – это встроенная, легковесная СУБД, которая не требует отдельного сервера и идеально подходит для мобильных приложений. Она поддерживает стандартные запросы, имеет низкое потребление ресурсов и высокую надежность. Однако, SQLite может быть недостаточно производительной при работе с очень большими объёмами данных и сложными запросами.

Realm – это объектно-ориентированная СУБД, разработанная специально для мобильных платформ. Она обеспечивает высокую скорость работы и удобный API для взаимодействия с данными. Realm поддерживает синхронизацию в реальном времени и шифрование. К недостаткам можно отнести ограниченную гибкость в сравнении с SQL-базами и необходимость соблюдения лицензионных условий для коммерческого использования.

ORMLite — это сторонняя библиотека, которая служит ORM-прослойкой для SQLite и упрощает работу с ним в Android-приложениях. ORMLite автоматически генерирует базовый CRUD-код, поддерживает аннотации для моделей и интеграцию с разными типами баз данных. Однако, ORMLite требует ручной настройки сложных SQL-запросов и не имеет встроенной поддержки LiveData или Android Architecture Components, что может увеличить время на разработку, особенно для тех, кто не работал с ORM-подходами ранее.

Firebase Realtime Database – это облачное решение для хранения данных, которое обеспечивает синхронизацию в реальном времени. Она удобна для приложений, требующих мгновенного обновления информации на всех устройствах. Недостатками являются зависимость от интернет-соединения, ограничения по размеру данных в бесплатном тарифе и сложность локального тестирования.

Realm и Firebase не подходят для разрабатываемой системы из-за лицензионных ограничений и требований к сетевому подключению. ORMLite, хотя и упрощает работу с SQLite, добавляет сложность в настройке. Поэтому в качестве СУБД было решено использовать SQLite, который обеспечивает баланс между производительностью, простотой и низкими требованиями к ресурсам.

Клиентское мобильное приложение должно быть разработано под операционную систему Android. Разработка мобильных приложений под эту платформу производится с помощью таких языков программирования, как Java, Kotlin, JavaScript, C#, Dart.

Язык Java раньше активно использовался для разработки под Android, но с появлением Kotlin потерял популярность. Java имеет обширную документацию, но её синтаксис считается более многословным, а поддержка современных функций требует дополнительных усилий.

Kotlin является современным языком программирования, официально поддерживаемый Google для разработки Android-приложений. Он сочетает лаконичность синтаксиса с мощными возможностями, такими как корутины для асинхронных операций, расширения для существующих классов и полную совместимость с Java. Kotlin снижает количество плохого кода и упрощает интеграцию с новыми архитектурными подходами.

Приложения под Android также разрабатываются с помощью Dart и фреймворка Flutter. Flutter позволяет создавать кроссплатформенные приложения с высокой производительностью, но имеет ограничения при работе с нативными API и увеличенный размер итогового APK.

JavaScript с React Native также используется для кроссплатформенной разработки, но может уступать в производительности на сложных задачах и требует дополнительных настроек для интеграции с нативными модулями.

C# с Xamarin подходит для создания приложений под Android, но менее популярен в сравнении с Kotlin, а взаимодействие с Java-библиотеками иногда вызывает сложности.

В качестве языка программирования был выбран Kotlin, так как он обеспечивает высокую продуктивность разработки, тесную интеграцию с экосистемой Android и поддержку современных парадигм программирования.

Таким образом, в качестве СУБД оптимальным выбором является SQLite, а языка программирования для разработки мобильного приложения под Android – язык Kotlin.

4 АДМИНИСТРИРОВАНИЕ БАЗ ДАННЫХ, СУЩЕСТВУЮЩИХ НА ПРЕДПРИЯТИИ

Была создана база данных со следующими сущностями:

* books (книги);
* book\_words (слова из книг).

Сущность books содержит информацию о книгах: уникальный идентификатор (id), название книги (title), путь к файлу книги (file\_path) и путь к превью обложки (preview\_path). Каждая книга представлена отдельной записью, что позволяет однозначно идентифицировать её в системе.

Сущность book\_words предназначена для хранения словарных единиц, связанных с конкретными книгами. Она включает уникальный идентификатор (id), внешний ключ на книгу (book\_id), оригинальное слово (original), его перевод (translation) и временную метку создания записи (timestamp). Связь между сущностями books и book\_words реализована через поле book\_id, обеспечивая отношение «один ко многим» — одна книга может содержать множество слов, но каждое слово относится только к одной книге в соответствии с рисунком 1.

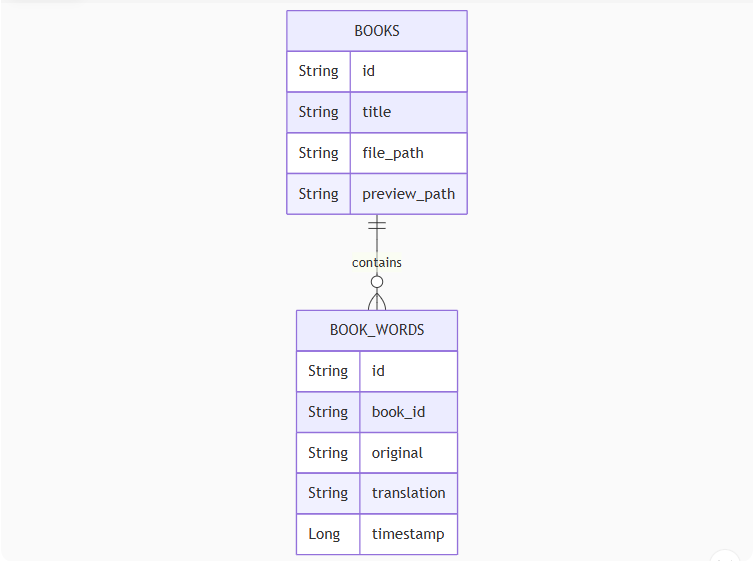


Рисунок 1 – ER-диаграмма сущностей в базе данных

Таким образом, база данных готова к интеграции с приложением, обеспечивая надёжное хранение и обработку данных о книгах и словарных единицах.

5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Во время прохождения преддипломной производственной практики было необходимо разработать прототип мобильного приложения для чтения иностранных книг.

Идея приложения заключается в том, что пользователи могут читать иностранные книги с встроенным переводчиком на основании искусственного интеллекта, а также изучать новые добавленные словосочетания. Это позволит упростить процесс изучения новых языков с помощью объединения переводчика, чтения книги и словаря в едином приложении.

Приложение разрабатывалось с помощью языка программирования Kotlin. За основу был взят архитектурный шаблон Model-View-ViewModel, обеспечивающий независимость логики приложения от интерфейса и гибкость разработки.

Разработка мобильного приложения началась с создания интерфейса экранов приложения. Для создания цветовой темы приложения использовался конструктор тем Material Theme Builder, позволяющий импортировать цветовую тему приложения для Android в соответствии с рисунком 2.

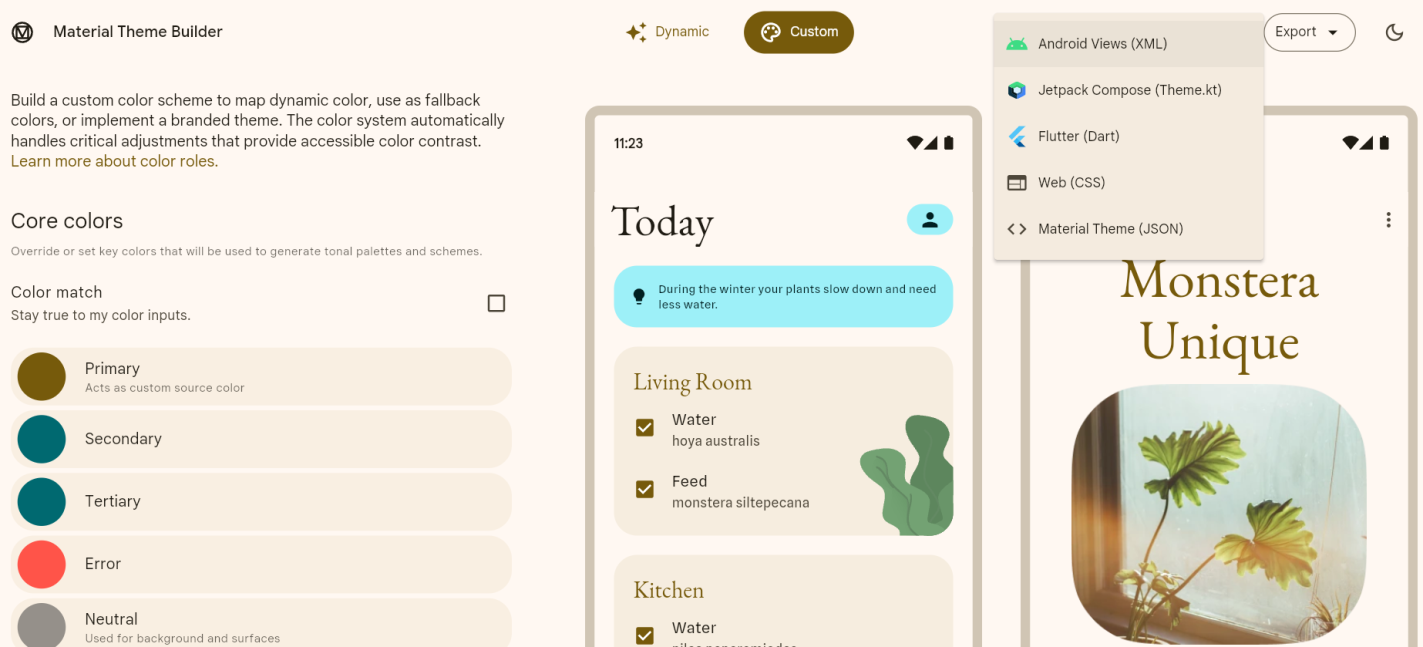


Рисунок 2 – Конструктор цветовых тем Material Theme Builder

В первую очередь был создан экран приветствия, содержащий информацию о приложении. Экран приветствия в соответствии с рисунком 3.

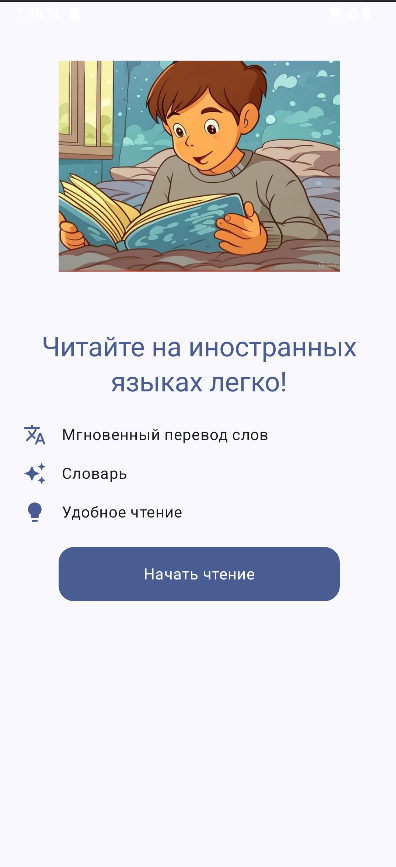


Рисунок 3 – Экран приветствия

Код экрана приветствия:

@Composable

fun WelcomeScreen(

onStartReading: () -> Unit

) {

val scrollState = rememberScrollState()

Column(

modifier = Modifier

.fillMaxSize()

.verticalScroll(scrollState)

.background(MaterialTheme.colorScheme.background)

.padding(24.dp),

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.spacedBy(24.dp)

) {

AsyncImage(

modifier = Modifier

.fillMaxWidth(0.8f)

.aspectRatio(1f),

model = R.drawable.ic\_reading\_illustration,

contentDescription = "Чтение с переводом",

contentScale = ContentScale.Fit

)

Text(

text = "Читайте на иностранных языках легко!",

style = MaterialTheme.typography.headlineMedium,

textAlign = TextAlign.Center,

color = MaterialTheme.colorScheme.primary

)

Column(

verticalArrangement = Arrangement.spacedBy(16.dp),

modifier = Modifier.fillMaxWidth()

) {

FeatureItem(icon = Icons.Default.Translate, text = "Мгновенный перевод слов")

FeatureItem(icon = Icons.Default.AutoAwesome, text = "Словарь")

FeatureItem(icon = Icons.Default.Lightbulb, text = "Удобное чтение")

}

Button(

onClick = onStartReading,

text = "Начать чтение"

)

}

}

@Composable

private fun FeatureItem(icon: ImageVector, text: String) {

Row(

verticalAlignment = Alignment.CenterVertically,

horizontalArrangement = Arrangement.spacedBy(16.dp)

) {

Icon(

imageVector = icon,

contentDescription = null,

tint = MaterialTheme.colorScheme.primary

)

Text(

text = text,

style = MaterialTheme.typography.bodyLarge,

color = MaterialTheme.colorScheme.onBackground

)

}

}

Далее был разработан список книг для чтения, которые пользователь добавляет из системных файлов сам в соответствии с рисунком 4.

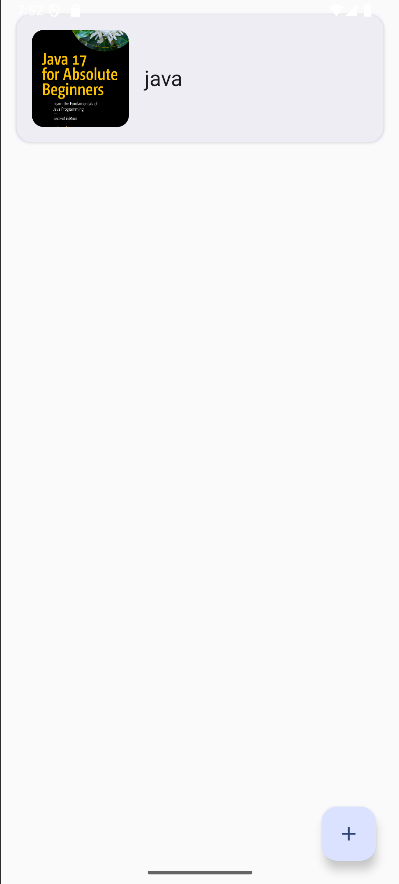


Рисунок 4 – Экран со списком книг

Код экрана со списком книг:

@Composable

fun BookListScreen(

viewModel: BookListViewModel = hiltViewModel(),

onClickedAddBook: () -> Unit,

onClicked: (id: String) -> Unit

) {

BookList(

state = viewModel.state,

onClicked = onClicked,

onDelete = { id -> viewModel.deleteBook(id) },

errorMessageState = viewModel.errorMessageState,

onClearError = { viewModel.clearError() },

loadingState = viewModel.loadingState,

onClickedAddBook = onClickedAddBook

)

}

@OptIn(ExperimentalFoundationApi::class)

@Composable

fun BookList(

modifier: Modifier = Modifier,

state: BookListContract.State,

onClicked: (id: String) -> Unit,

onDelete: (id: String) -> Unit,

errorMessageState: String?,

onClearError: () -> Unit,

loadingState: Boolean,

onClickedAddBook: () -> Unit

) {

val bookList: List<Book> = state.items

val scrollState = rememberLazyListState()

Box(modifier = modifier.fillMaxSize()) {

if (bookList.isEmpty()) {

NoDataPlug(

modifier = Modifier.align(Alignment.Center),

title = "Библиотека пуста",

subtitle = "Добавьте первую книгу для чтения"

)

} else {

LazyColumn(

modifier = Modifier.fillMaxSize(),

state = scrollState,

verticalArrangement = Arrangement.spacedBy(16.dp),

contentPadding = PaddingValues(16.dp)

) {

itemsIndexed(bookList) { index, book ->

SwipeToDeleteContainer(

onDelete = { onDelete(book.id) },

modifier = Modifier.animateItemPlacement()

) {

BookCard(

book = book,

onClick = { onClicked(book.id) }

)

}

}

}

}

FloatingActionButton(

modifier = Modifier

.padding(24.dp)

.align(Alignment.BottomEnd),

onClick = onClickedAddBook,

containerColor = MaterialTheme.colorScheme.primaryContainer,

contentColor = MaterialTheme.colorScheme.onPrimaryContainer

) {

Icon(

imageVector = Icons.Default.Add,

contentDescription = "Добавить книгу"

)

}

if (loadingState) {

LoadingBar(modifier = Modifier.align(Alignment.Center))

}

errorMessageState?.let { errorMessage ->

ErrorAlertDialog(

text = errorMessage,

onDismiss = onClearError

)

}

}

}

@Composable

private fun BookCard(

book: Book,

onClick: () -> Unit

) {

Card(

modifier = Modifier

.fillMaxWidth()

.clickable(onClick = onClick),

shape = MaterialTheme.shapes.large,

colors = CardDefaults.cardColors(

containerColor = MaterialTheme.colorScheme.surfaceContainer

),

elevation = CardDefaults.cardElevation(defaultElevation = 2.dp)

) {

Row(

modifier = Modifier

.fillMaxWidth()

.padding(16.dp),

verticalAlignment = Alignment.CenterVertically,

horizontalArrangement = Arrangement.spacedBy(16.dp)

) {

BookCover(

previewPath = book.previewPath

)

Column(

modifier = Modifier.weight(1f),

verticalArrangement = Arrangement.spacedBy(4.dp)

) {

Text(

text = book.title,

style = MaterialTheme.typography.titleLarge,

color = MaterialTheme.colorScheme.onSurface,

maxLines = 2,

overflow = TextOverflow.Ellipsis

)

}

}

}

}

@Composable

private fun BookCover(

previewPath: String

) {

Box(

modifier = Modifier

.size(100.dp)

.clip(MaterialTheme.shapes.medium)

) {

AsyncImage(

modifier = Modifier.fillMaxSize(),

model = File(previewPath),

contentDescription = "Обложка книги",

contentScale = ContentScale.Crop

)

}

}

Следующим на разработку стал экран добавления книги, который открывается с помощью кнопки со знаком плюсик в соответствии с рисунком 5.

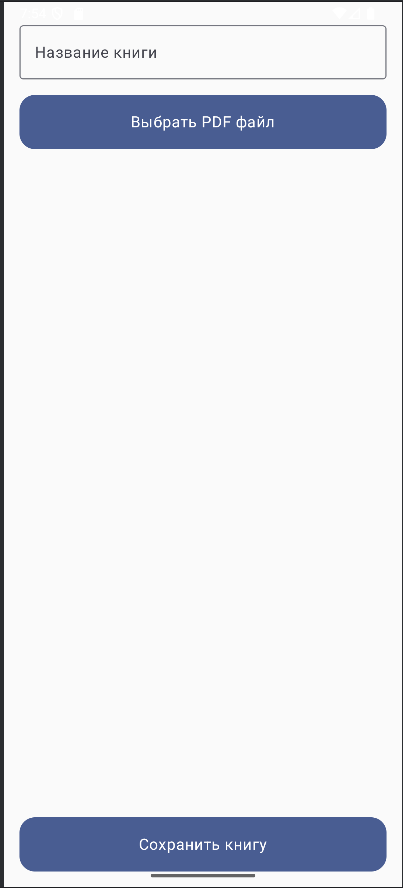


Рисунок 5 – Экран добавления книги

Код экрана добавления книги:

@Composable

fun AddBookScreen(

viewModel: AddBookViewModel = hiltViewModel(),

onBackClicked: () -> Unit

) {

AddBookScreen(

state = viewModel.state,

errorMessage = viewModel.errorMessageState,

setSelectedPdfUri = { uri -> viewModel.setSelectedPdfUri(uri) },

saveBook = { title -> viewModel.saveBook(title) },

onClearError = { viewModel.clearError() },

loadingState = viewModel.loadingState,

onBackClicked = onBackClicked

)

}

@Composable

fun AddBookScreen(

state: AddBookContract.State,

errorMessage: String?,

setSelectedPdfUri: (Uri) -> Unit,

saveBook: (String?) -> Unit,

onClearError: () -> Unit,

loadingState: Boolean,

onBackClicked: () -> Unit

) {

var title by remember { mutableStateOf("") }

val filePicker = rememberLauncherForActivityResult(

contract = ActivityResultContracts.OpenDocument(),

onResult = { uri ->

uri?.let {

setSelectedPdfUri(it)

}

}

)

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.SpaceBetween

) {

Column(

modifier = Modifier

.weight(1f)

.verticalScroll(rememberScrollState()),

verticalArrangement = Arrangement.spacedBy(16.dp),

horizontalAlignment = Alignment.CenterHorizontally,

) {

OutlinedTextField(

value = title,

onValueChange = { title = it },

label = {

Text("Название книги")

},

isError = errorMessage != null,

modifier = Modifier.fillMaxWidth()

)

Button(

text = "Выбрать PDF файл",

onClick = {

filePicker.launch(arrayOf("application/pdf"))

},

modifier = Modifier.fillMaxWidth()

)

state.previewBitmap?.let { bitmap ->

Image(

bitmap = bitmap.asImageBitmap(),

contentDescription = "Превью",

contentScale = ContentScale.Fit,

modifier = Modifier

.fillMaxWidth()

.weight(1f)

.clip(RoundedCornerShape(8.dp))

)

}

}

Button(

text = "Сохранить книгу",

onClick = {

onClearError()

saveBook(title)

onBackClicked()

},

modifier = Modifier.fillMaxWidth()

)

}

if (loadingState) {

LoadingBar()

}

errorMessage?.let {

Text(

text = it,

color = MaterialTheme.colorScheme.error,

modifier = Modifier.padding(horizontal = 16.dp)

)

}

}

Далее был создан экран чтения книги в соответствии с рисунком 6.

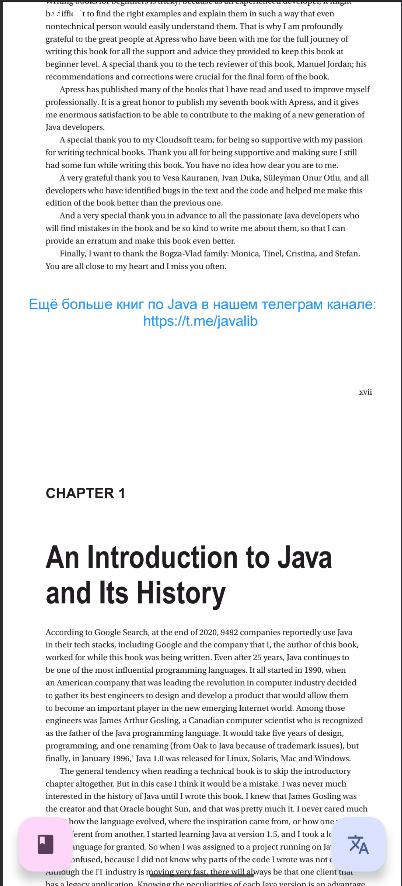


Рисунок 6 – Экран чтения книги

Код экрана чтения книги:

@Composable

fun BookReaderScreen(

viewModel: BookReaderViewModel = hiltViewModel(),

onClickShowDictionary: (String) -> Unit

) {

LaunchedEffect(Unit) {

viewModel.updateState()

}

BookReaderScreen(

mainState = viewModel.mainState,

translationState = viewModel.translationState,

errorMessageState = viewModel.errorMessageState,

onClearError = { viewModel.clearError() },

loadingState = viewModel.loadingState,

handleTranslation = { text ->

viewModel.handleTextSelection(text)

},

clearTranslation = {

viewModel.clearTranslation()

},

onClickShowDictionary = onClickShowDictionary,

onAddToDictionary = {

viewModel.addCurrentTranslationToDictionary()

}

)

}

@Composable

fun BookReaderScreen(

mainState: BookReaderContract.MainState,

translationState: BookReaderContract.TranslationState,

errorMessageState: String?,

onClearError: () -> Unit,

loadingState: Boolean,

handleTranslation: (String) -> Unit,

clearTranslation: () -> Unit,

onClickShowDictionary: (String) -> Unit,

onAddToDictionary: () -> Unit

) {

val book: Book? = mainState.book

val bookPath: String = book?.filePath ?: ""

if (bookPath.isEmpty()) {

LoadingBar()

return

}

val pdfFile = remember { File(bookPath) }

var pdfView by remember { mutableStateOf<PDFView?>(null) }

var showTranslationInput by remember { mutableStateOf(false) }

var selectedText by remember { mutableStateOf("") }

Box(modifier = Modifier.fillMaxSize()) {

AndroidView(

factory = { ctx ->

PDFView(ctx, null).apply {

fromFile(pdfFile)

.enableDoubletap(true)

.enableSwipe(true)

.swipeHorizontal(false)

.load()

pdfView = this

}

},

modifier = Modifier

.fillMaxSize()

)

FloatingActionButton(

onClick = { showTranslationInput = true },

modifier = Modifier

.align(Alignment.BottomEnd)

.padding(16.dp),

containerColor = MaterialTheme.colorScheme.primaryContainer,

contentColor = MaterialTheme.colorScheme.primary

) {

Icon(

imageVector = Icons.Default.Translate,

contentDescription = "Перевести текст"

)

}

FloatingActionButton(

onClick = { onClickShowDictionary(book?.id!!) },

modifier = Modifier

.align(Alignment.BottomStart)

.padding(16.dp),

containerColor = MaterialTheme.colorScheme.tertiaryContainer

) {

Icon(Icons.Default.Book, "Словарь")

}

if (showTranslationInput) {

TranslationInputDialog(

initialText = selectedText,

onDismiss = { showTranslationInput = false },

onConfirm = { text ->

handleTranslation(text)

showTranslationInput = false

}

)

}

if (loadingState) {

LoadingBar()

}

if (errorMessageState != null) {

ErrorAlertDialog(

text = errorMessageState,

onDismiss = onClearError

)

}

if (translationState.translation.isNotEmpty()) {

TranslationResultDialog(

originalText = translationState.text,

translation = translationState.translation,

onDismiss = {

clearTranslation()

},

onRetry = { showTranslationInput = true },

onAddToDictionary = onAddToDictionary

)

}

}

}

@Composable

private fun TranslationInputDialog(

initialText: String,

onDismiss: () -> Unit,

onConfirm: (String) -> Unit

) {

var inputText by remember { mutableStateOf(initialText) }

Dialog(onDismissRequest = onDismiss) {

Surface(

modifier = Modifier

.padding(16.dp)

.widthIn(max = 480.dp)

.heightIn(max = 640.dp),

shape = MaterialTheme.shapes.large,

tonalElevation = 8.dp

) {

Column(modifier = Modifier.padding(16.dp)) {

Text(

text = "Введите текст для перевода",

style = MaterialTheme.typography.headlineSmall

)

Spacer(modifier = Modifier.height(16.dp))

OutlinedTextField(

value = inputText,

onValueChange = { inputText = it },

modifier = Modifier.fillMaxWidth(),

label = { Text("Текст для перевода") },

maxLines = 5

)

Spacer(modifier = Modifier.height(24.dp))

Row(

horizontalArrangement = Arrangement.End,

modifier = Modifier.fillMaxWidth()

) {

TextButton(onClick = onDismiss) {

Text("Отмена")

}

Spacer(modifier = Modifier.width(8.dp))

Button(

onClick = { onConfirm(inputText) },

enabled = inputText.isNotBlank()

) {

Text("Перевести")

}

}

}

}

}

}

@Composable

fun TranslationResultDialog(

originalText: String,

translation: String,

onDismiss: () -> Unit,

onRetry: () -> Unit,

onAddToDictionary: () -> Unit

) {

Dialog(

onDismissRequest = onDismiss

) {

Card(

modifier = Modifier

.padding(16.dp)

.widthIn(max = 480.dp),

shape = MaterialTheme.shapes.extraLarge,

colors = CardDefaults.cardColors(

containerColor = MaterialTheme.colorScheme.surfaceContainerHigh

)

) {

Column(

modifier = Modifier.padding(24.dp),

verticalArrangement = Arrangement.spacedBy(20.dp)

) {

Text(

text = "Результат перевода",

style = MaterialTheme.typography.headlineSmall,

color = MaterialTheme.colorScheme.primary

)

Column(

modifier = Modifier.verticalScroll(rememberScrollState()),

verticalArrangement = Arrangement.spacedBy(16.dp)

) {

TranslationBlock(

title = "Оригинал",

text = originalText,

icon = Icons.Outlined.TextSnippet,

containerColor = MaterialTheme.colorScheme.surfaceContainer

)

TranslationBlock(

title = "Перевод",

text = translation,

icon = Icons.Outlined.Translate,

containerColor = MaterialTheme.colorScheme.primaryContainer

)

}

Column(

verticalArrangement = Arrangement.spacedBy(12.dp),

horizontalAlignment = Alignment.CenterHorizontally

) {

FilledTonalButton(

onClick = {

onAddToDictionary()

onDismiss()

},

modifier = Modifier.fillMaxWidth(),

colors = ButtonDefaults.filledTonalButtonColors(

containerColor = MaterialTheme.colorScheme.secondaryContainer

)

) {

Icon(Icons.Outlined.BookmarkAdd, null)

Spacer(Modifier.width(8.dp))

Text("Сохранить в словарь")

}

OutlinedButton(

onClick = onRetry,

modifier = Modifier.fillMaxWidth(),

border = ButtonDefaults.outlinedButtonBorder.copy(

width = 1.dp

)

) {

Icon(Icons.Outlined.Edit, null)

Spacer(Modifier.width(8.dp))

Text("Редактировать текст")

}

TextButton(

onClick = onDismiss,

modifier = Modifier.align(Alignment.End)

) {

Text("Закрыть")

}

}

}

}

}

}

@Composable

private fun TranslationBlock(

title: String,

text: String,

icon: ImageVector,

containerColor: Color

) {

Column(

modifier = Modifier

.fillMaxWidth()

.clip(MaterialTheme.shapes.medium)

.background(containerColor)

.padding(16.dp),

verticalArrangement = Arrangement.spacedBy(8.dp)

) {

Row(

verticalAlignment = Alignment.CenterVertically,

horizontalArrangement = Arrangement.spacedBy(8.dp)

) {

Icon(

imageVector = icon,

contentDescription = null,

tint = MaterialTheme.colorScheme.onSurfaceVariant

)

Text(

text = title,

style = MaterialTheme.typography.labelLarge,

color = MaterialTheme.colorScheme.onSurfaceVariant

)

}

}

}

Следующим был разработан словарь, содержащий список слов, который пользователь добавляет в процессе чтения книги в соответствии с рисунком 7.



Рисунок 7 – Экран со списком слов

Для закрепления добавленных слов необходимо в словаре нажать на кнопку «Изучать». По нажатию откроется игра, которая способствует быстрому запоминанию новых фраз в соответствии с рисунком 8.

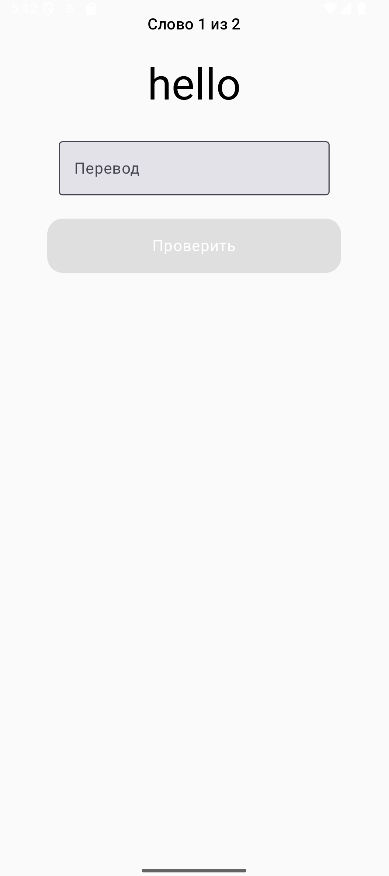


Рисунок 8 – Экран изучения слов

Код экрана изучения слов:

@Composable

fun QuizScreen(

viewModel: QuizViewModel = hiltViewModel(),

onFinish: () -> Unit

) {

QuizScreen(

state = viewModel.state,

onFinish = onFinish,

onAnswerChanged = viewModel::onAnswerChanged,

nextWord = viewModel::nextWord,

checkAnswer = viewModel::checkAnswer

)

}

@Composable

fun QuizScreen(

state: QuizContract.State,

onFinish: () -> Unit,

onAnswerChanged: (String) -> Unit,

nextWord: () -> Unit,

checkAnswer: () -> Unit

) {

if (state.isFinished) {

LaunchedEffect(Unit) {

onFinish()

}

} else {

Box(

modifier = Modifier

.fillMaxSize()

.padding(16.dp)

) {

Column(

modifier = Modifier.fillMaxSize(),

verticalArrangement = Arrangement.spacedBy(24.dp),

horizontalAlignment = Alignment.CenterHorizontally

) {

Text(

text = "Слово ${state.currentIndex + 1} из ${state.totalWords}",

style = MaterialTheme.typography.titleMedium

)

Text(

text = state.currentWord?.original ?: "",

style = MaterialTheme.typography.displayMedium

)

OutlinedTextField(

value = state.userAnswer,

onValueChange = onAnswerChanged,

label = { Text("Перевод") },

enabled = !state.isAnswerChecked,

isError = state.isAnswerWrong,

colors = TextFieldDefaults.colors(

errorContainerColor = MaterialTheme.colorScheme.errorContainer,

errorTextColor = MaterialTheme.colorScheme.onErrorContainer

)

)

if (state.isAnswerChecked) {

Column(verticalArrangement = Arrangement.spacedBy(8.dp)) {

if (state.isAnswerWrong) {

Text(

text = "Правильный ответ: ${state.currentWord?.translation}",

color = MaterialTheme.colorScheme.error

)

}

Button(

text = "Следующее слово",

onClick = nextWord

)

}

} else {

Button(

text = "Проверить",

onClick = checkAnswer,

enabled = state.userAnswer.isNotBlank()

)

}

}

}

}

}

В итоге прототип приложения был успешно разработан. Было проведено тестирование функционала, в ходе которого ошибок не было обнаружено.

6 ВЫПОЛНЕНИЕ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

В качестве индивидуального задания было необходимо выполнить разработку модуля для терминала сбора данных для регистрации нанесённых кодов маркировки.

Основной задачей компании «Макси» стала необходимость маркировки собственной продукции. Для этого через платформу «Честный знак» генерируются уникальные коды маркировки, которые затем печатаются и размещаются на упаковках товаров. После физического нанесения кодов требуется собрать данные о каждом маркировочном идентификаторе и зарегистрировать товары в системе «Честный знак», чтобы обеспечить их легализацию для последующей реализации.

Терминалы сбора данных, функционирующие под управлением ОС Android, требуют специализированного ПО, поэтому разработка ведётся в среде Android Studio с применением языков Java и Kotlin. Созданное приложение включает интерфейс с отображением списка ранее отсканированных кодов, кнопку для добавления новых маркировок и элемент для отправки собранных данных в систему в соответствии с рисунком 9. В списке отображается наименование товара, его уникальный идентификатор в базе данных, а также количество отсканированных экземпляров.

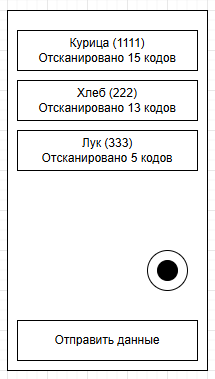


Рисунок 9 – Макет экрана маркировок

При нажатии на кнопку добавления кода маркировки открывается экран сканирования кода в соответствии с рисунком 10, если в базе данных найден такой код, то он добавляется, иначе появляется ошибка «Неизвестный код маркировки».

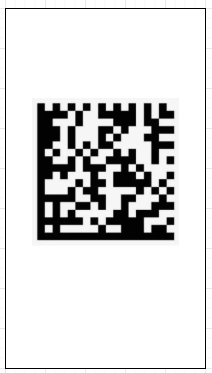


Рисунок 10 – Экран сканирования кода маркировки

7 СБОР И ОБОБЩЕНИЕ МАТЕРИАЛА ДЛЯ ДИПЛОМНОГО ПРОЕКТИРОВАНИЯ

Целью дипломного проекта явилась разработка мобильного приложения для чтения иностранных книг, позволяющего пользователям читать иностранные текста с мгновенным переводом. Данное приложение предназначено для операционной системы Android, используемой в большей части современных смартфонов. Для разработки было решено использовать язык программирования Kotlin, из-за его популярности среди разработчиков. В качестве среды разработки была взята Android Studio, так как содержит эмуляторы смартфонов разных моделей. Для хранения данных была выбрана бесплатная СУБД SQLite, предоставляющая широкий спектр возможностей для разработчиков.

В настоящем разделе рассмотрим вопросы, связанные с обеспечением безопасных условий труда программиста при разработке мобильного приложения для чтения иностранных книг.

Современный этап развития общества характеризуется активным внедрением цифровых технологий во все сферы деятельности. Персональные компьютеры стали неотъемлемой частью рабочего процесса, однако их длительное использование сопряжено с рисками для здоровья. Охрана труда при работе с компьютером направлена на минимизацию воздействия вредных факторов, сохранение работоспособности и профилактику профессиональных заболеваний.

Необходимо обязательно соблюдать требования электро- и пожарной безопасности, правила личной гигиены, знать способы рациональной организации рабочего места, методы защиты от вредных факторов, порядок действий при поломках или возгорании. Важно уметь пользоваться огнетушителями, знать расположение аптечки и соблюдать режим отдыха. Запрещено работать в состоянии алкогольного или наркотического опьянения.

Перед началом работы необходимо проверить исправность освещения: осветительные приборы должны быть надежно закреплены, без загрязнений, уровень искусственной освещенности – не менее 300 люкс. Розетки и выключатели не должны иметь повреждений. Следует убедиться в свободных путях эвакуации, наличии исправных огнетушителей и аптечки. Помещение необходимо проветрить, зафиксировав окна в открытом положении.

Рабочее место должно иметь площадь не менее 4,5 м². Мебель должна быть устойчивой, кабели – целыми, без перегибов и скручиваний. Системный блок нельзя размещать в закрытых отсеках, а источник бесперебойного питания – на расстоянии. Монитор необходимо установить так, чтобы исключить блики. После включения компьютера проверяется отсутствие сбоев, четкость изображения, яркость и контрастность. При недостатке света используется настольная лампа, не создающая бликов. При обнаружении неисправностей начинать работу до их устранения запрещено.

Во время работы запрещается прикасаться к задней панели системного блока при включенном питании, переключать разъемы кабелей под напряжением, допускать попадание влаги на технику, самостоятельно вскрывать или ремонтировать оборудование, а также отключать устройства, держась за шнур.

Компьютер используется только в исправном состоянии, согласно инструкции производителя. Монитор настраивается под правильную позу: спинка кресла вертикальна, расстояние до экрана – не менее 50 см. Нельзя качаться на стуле или облокачиваться на технику. Рабочее место должно быть свободным от лишних предметов. Клавиатуру и мышь ежедневно необходимо дезинфицировать спиртовыми средствами.

Особое внимание необходимо уделить защите персональных данных и учетных записей. Использование слабых паролей, включающих даты рождения, повторяющиеся символы или простые последовательности, категорически запрещено. Пароль должен содержать не менее 12 символов с комбинацией заглавных и строчных букв, цифр и специальных знаков. Запрещается записывать пароли на бумажных носителях, размещать их на мониторе, рабочем столе или в других видимых местах. При временном отсутствии за рабочим местом необходимо заблокировать компьютер комбинацией клавиш «Win+L» или полностью выключить устройство. В случае подозрения на утечку или компрометацию пароля необходимо немедленно изменить учетные данные.

Антивирусный контроль является неотъемлемой частью безопасности. Все файлы, полученные через электронную почту, облачные сервисы или съемные носители, перед открытием проверяются антивирусным программным обеспечением. Запрещается открывать вложения от неизвестных отправителей, а также использовать зарубежные почтовые сервисы для передачи конфиденциальной информации. Шифрование данных допускается только с применением сертифицированных криптографических средств, одобренных организацией.

Электронная почта требует повышенной осторожности. Перед открытием писем необходимо убедиться в подлинности отправителя, даже если адрес кажется знакомым, так как злоумышленники часто маскируются. Пароли от учетных записей не должны передаваться третьим лицам. В случае получения подозрительного письма с запросом конфиденциальных данных или ссылками на внешние ресурсы следует поместить его в спам.

Важным элементом безопасности является своевременное обновление программного обеспечения и операционных систем, устраняющее потенциальные уязвимости. Резервное копирование данных должно проводиться регулярно с использованием зашифрованных внешних носителей или защищенных облачных сервисов. Запрещается устанавливать непроверенные приложения или игнорировать системные предупреждения, так как это повышает риск кибератак.

Каждый час работы сопровождается 10-15-минутным перерывом для гимнастики глаз в соответствии с рисунком 11.

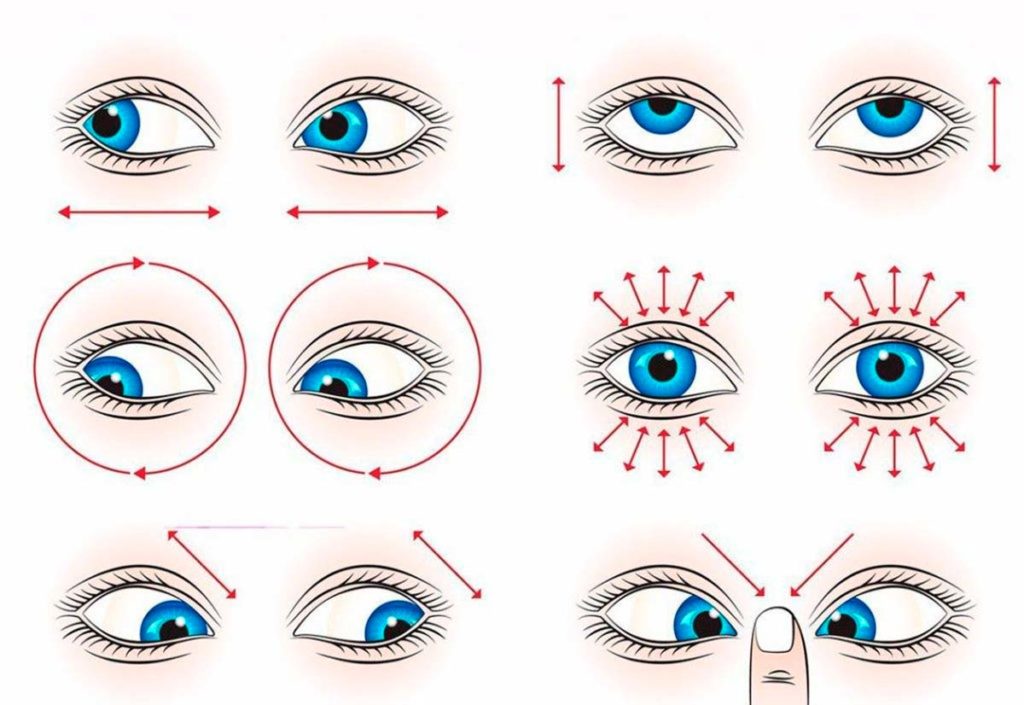


Рисунок 11 – Гимнастика для глаз

В соответствии с рисунком 12 делаются упражнения для профилактики утомления.



Рисунок 12 – Рекомендуемые упражнения для профилактики утомления

Помещение проветривается каждые два часа. При временном прекращении работы компьютер переводится в режим ожидания, при длительном – выключается. Запрещено подключать устройства мокрыми руками, использовать неисправные кабели, размещать на технике посторонние предметы, передвигать включенный компьютер, вскрывать корпус. При обнаружении неисправностей: шум, дым, искры, работа прекращается, оборудование обесточивается. При отключении электричества компьютер выключается. При зрительном дискомфорте увеличивают продолжительность перерывов.

По окончании работы произвести закрытие всех активных задач. выключить питание системного блока, питание всех периферийных устройств. Рабочее место приводится в порядок: техника протирается антистатическими салфетками, экран очищается специальными средствами. Клавиатура и мышь дезинфицируются. Помещение проветривается, окна фиксируются.

Таким образом, изложенные выше предложения обеспечат безопасные условия труда программиста при разработке мобильного приложения для чтения иностранных книг.

ЗАКЛЮЧЕНИЕ

В ходе прохождения производственной практики были закреплены полученные в процессе обучения знания. Была изучена компания, её бизнес-процессы, правила по охране труда, распорядок дня. Были настроены необходимые для работы программы. Была выполнена работа с тестовой базой данных Oracle. Провелась разработка программного модуля дипломного проекта на тему «Разработка мобильного приложения для чтения иностранных книг». Также было выполнено индивидуальное задание по разработке модуля для сканирования кодов маркировки с помощью терминалов сбора данных на операционной системе Android.

Поставленная цель производственной практики достигнута в полном объёме.