СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc167823969)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР 3](#_Toc167823970)

[1.1 Описание предметной области 3](#_Toc167823971)

[1.2 Требования к программному обеспечению 4](#_Toc167823972)

[1.3 Обоснование выбора средств разработки 5](#_Toc167823973)

[2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 8](#_Toc167823974)

[2.1 Проектирование базы данных 8](#_Toc167823975)

[2.2 Разработка программного функционала мобильного приложения 9](#_Toc167823976)

[2.3 Тестирование 34](#_Toc167823977)

[2.4 Руководство пользователя 34](#_Toc167823978)

[3 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ 40](#_Toc167823979)

[4 ОБЕСПЕЧЕНИЕ БЕЗОПАСНЫХ УСЛОВИЙ ТРУДА ПРОГРАММИСТА ПРИ РАЗРАБОТКЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ БРОНИРОВАНИЯ 43](#_Toc167823980)

[ЗАКЛЮЧЕНИЕ 48](#_Toc167823981)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49](#_Toc167823982)

Введение

Часть людей сталкивается с желанием читать иностранные книги для изучения языка или культурного погружения. Однако процесс чтения часто прерывается необходимостью искать переводы слов, разбирать сложные конструкции или адаптировать текст под свой уровень владения языком. Традиционные методы, такие как бумажные словари или отдельные онлайн-переводчики, усложняют восприятие и снижают вовлечённость. Современные цифровые решения частично решают эти проблемы, но большинство приложений не объединяют функции перевода и адаптации текста в едином интерфейсе, что снижает их эффективность для языковой практики. Кроме того, отсутствие интерактивных элементов в существующих инструментах затрудняет формирование устойчивых навыков восприятия иностранных текстов. Это создаёт потребность в решении, которое не только упростит перевод, но и превратит чтение в комплексный образовательный процесс.

Возникла идея создать мобильное приложение, которое сочетает чтение иностранных текстов с мгновенным переводом на базе искусственного интеллекта. Такое решение упростит языковую практику, сделает её более эффективной и доступной для широкой аудитории.

Целью дипломного проекта является разработка мобильного приложения для чтения иностранных книг.

Задачи:

* описать предметную область;
* собрать требования к программному обеспечению;
* обосновать выбор средств разработки;
* спроектировать базу данных;
* описать графический интерфейс;
* разработать программный функционал мобильного приложения;
* провести тестирование разработанного мобильного приложения;
* написать руководство пользователя.

1. Аналитический обзор

1.1 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.

1.2 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.

1.3 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.

1. Разработка программного продукта
   1. Проектирование базы данных

В соответствии с требованиями 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.

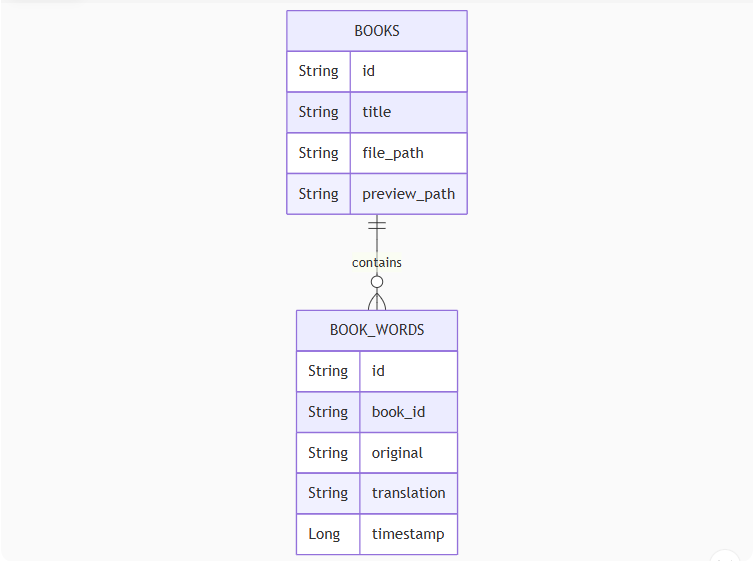


Рисунок 1 – ER-диаграмма сущностей в базе данных

База данных соответствует требованиям нормализации. Первая нормальная форма соблюдена за счёт атомарности всех атрибутов и отсутствия повторяющихся групп данных. Вторая нормальная форма выполнена, так как неключевые атрибуты полностью зависят от первичных ключей. Например, в сущности book\_words поля original, translation и timestamp зависят от составного ключа (id, book\_id). Третья нормальная форма обеспечена отсутствием транзитивных зависимостей между неключевыми атрибутами: все данные зависят исключительно от первичных ключей своих сущностей.

Для целостности данных настроены внешние ключи, например, поле book\_id в book\_words ссылается на books.id, что гарантирует корректность связей. Использование искусственных первичных ключей (id) упрощает управление уникальностью записей. В качестве примера тестовых данных можно привести запись в таблице books с id «B1», названием «Example Book», путями к файлу и превью, а также связанную запись в book\_words с id «W1», содержащую слово «Hello», его перевод «Привет» и временную метку.

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

* 1. Разработка программного функционала мобильного приложения

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.

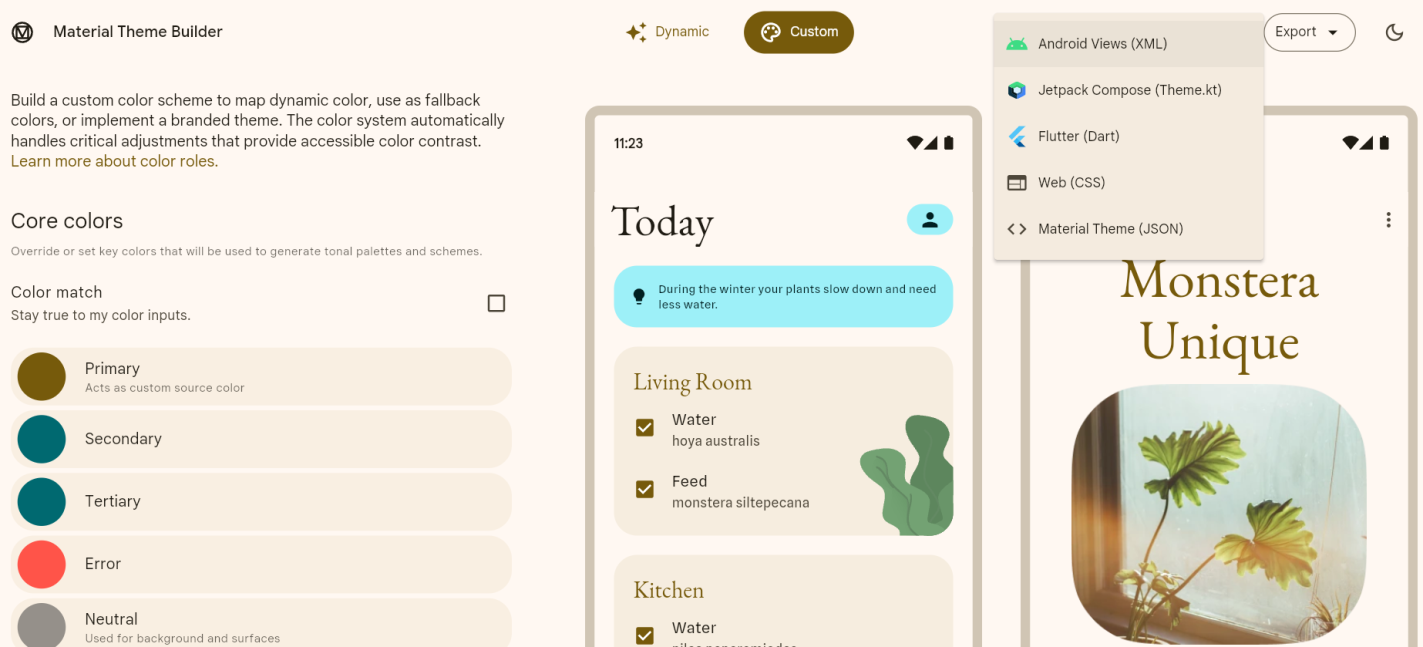


Рисунок 2 – Конструктор цветовых тем Material Theme Builder

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

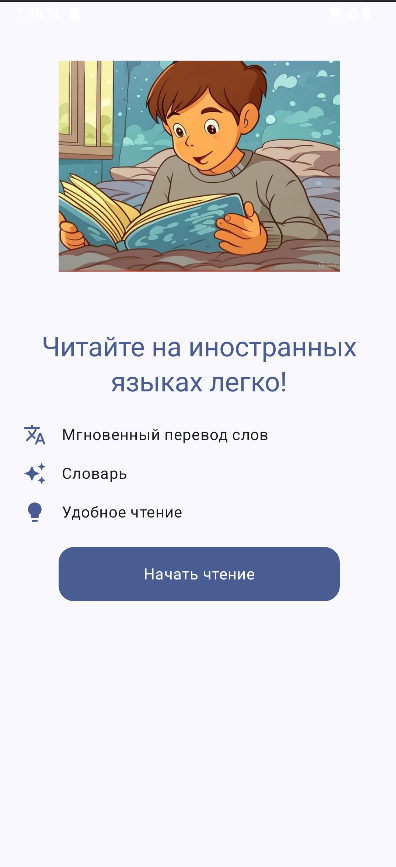


Рисунок 3 – Экран приветствия

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.

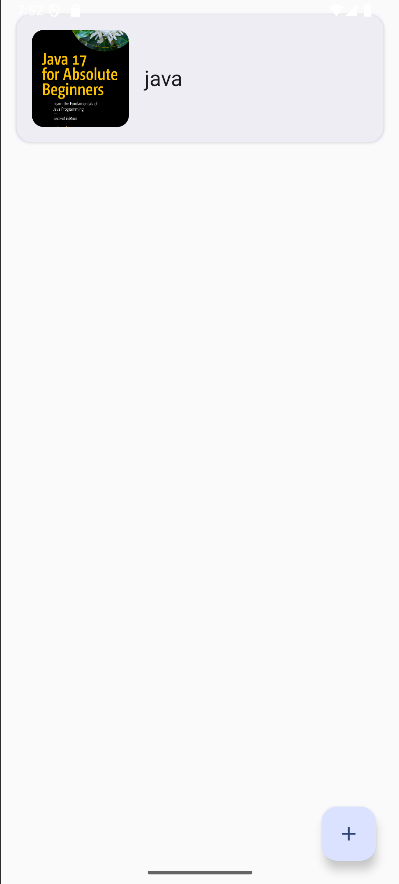


Рисунок 4 – Экран со списком книг

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.

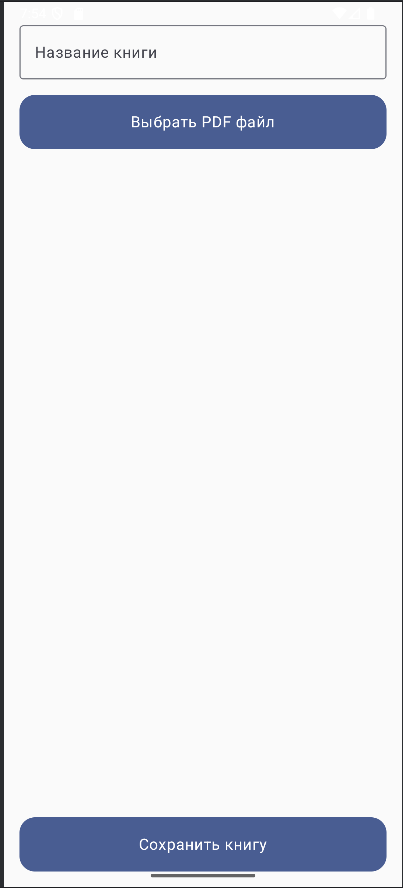


Рисунок 5 – Экран добавления книги

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.

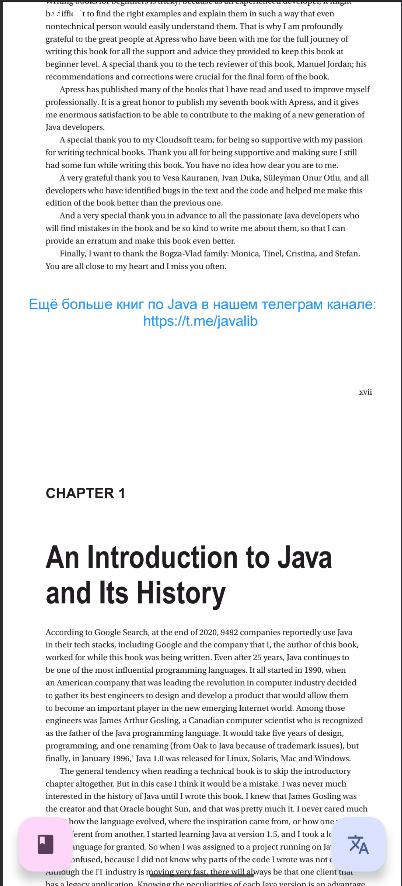


Рисунок 6 – Экран чтения книги

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.

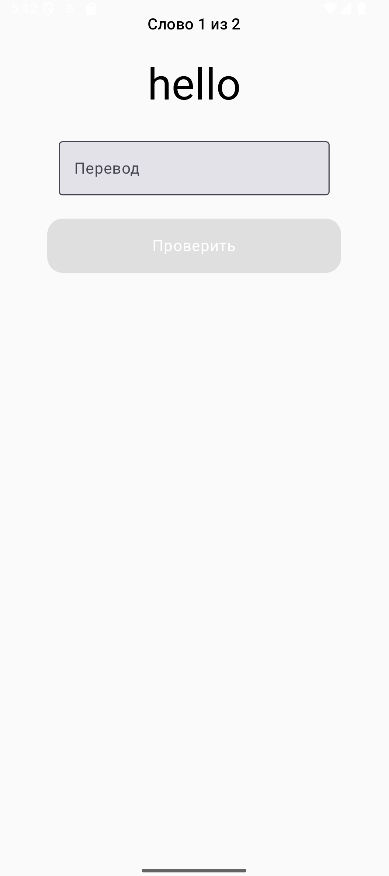


Рисунок 8 – Экран изучения слов

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()

)

}

}

}

}

}

Следующим этапом по разработке приложения была реализация бизнес-логики. В первую очередь, необходима была поддержка чтения PDF файлов. Для чтения книг был создан собственный класс PdfHandler. Код класса PdfHandler:

/\*\*

\* Handles PDF file operations including preview generation and storage management.

\* Manages internal file storage structure for books and previews.

\*/

class PdfHandler @Inject constructor(

@ApplicationContext val context: Context

) {

/\*\*

\* Generates a bitmap preview of the first page of a PDF document.

\* Scales the page to double its original dimensions for better visibility.

\*

\* @param uri Content URI of the PDF file to preview

\* @return [Bitmap] of the first page, or `null` if:

\* - File cannot be opened

\* - PDF rendering fails

\* - Insufficient memory

\* @throws SecurityException If URI access permission isn't granted

\*/

fun generatePdfPreview(uri: Uri): Bitmap? {

return try {

val parcelFileDescriptor = context.contentResolver.openFileDescriptor(uri, "r")

val pdfRenderer = PdfRenderer(parcelFileDescriptor!!)

val page = pdfRenderer.openPage(0)

val bitmap = createBitmap(page.width \* 2, page.height \* 2)

page.render(bitmap, null, null, PdfRenderer.Page.RENDER\_MODE\_FOR\_DISPLAY)

page.close()

pdfRenderer.close()

bitmap

} catch (\_: Exception) {

null

}

}

/\*\*

\* Saves PDF file to internal storage and generates its preview image.

\* Creates following directory structure:

\* - `filesDir/books/` for PDFs

\* - `filesDir/previews/` for preview images

\*

\* @param uri Content URI of source PDF file

\* @return Pair containing:

\* - First: Absolute path to saved PDF file

\* - Second: Absolute path to generated preview image

\* @throws SecurityException If URI access permission isn't granted

\* @throws IllegalArgumentException For invalid URI

\*/

fun savePdfWithPreview(uri: Uri): Pair<String, String> {

val pdfPath = saveToInternalStorage(uri)

val previewPath = saveBitmapToFile(generatePdfPreview(uri)!!)

return pdfPath to previewPath

}

private fun saveToInternalStorage(

uri: Uri

): String {

val file = File(context.filesDir, "books/${generateUUID()}.pdf")

file.parentFile?.mkdirs()

context.contentResolver.openInputStream(uri)?.use { input ->

FileOutputStream(file).use { output ->

input.copyTo(output)

}

}

return file.absolutePath

}

private fun saveBitmapToFile(bitmap: Bitmap): String {

val file = File(context.filesDir, "previews/${generateUUID()}.png")

file.parentFile?.mkdirs()

FileOutputStream(file).use {

bitmap.compress(Bitmap.CompressFormat.PNG, 90, it)

}

return file.absolutePath

}

}

В приложении необходимо было встроить возможность перевода иностранных слов на русский язык. Для этого функционала был реализован сетевой класс LocalRepository, который используется для работы с API искусственного интеллекта от сбербанка. Код класса LocalRepository:

/\*\*

\* Implementation of [LocalRepository]

\*/

class LocalRepositoryImpl @Inject constructor(

private val dao: Dao

) : LocalRepository {

override fun getBooks(): Flow<List<Book>> {

return dao.getBooks().map { it.toBooks() }

}

override suspend fun getBook(id: String): Book {

return dao.getBook(id).toBook()

}

override suspend fun saveBook(book: Book) {

dao.saveBook(book.toBookDatabase())

}

override suspend fun deleteBook(id: String) {

dao.deleteBook(id)

}

override fun getWordsForBook(bookId: String): Flow<List<BookWord>> {

return dao.getWordsForBook(bookId).map { it.toBookWords() }

}

override suspend fun insert(word: BookWord) {

dao.insert(word.toBookWordDatabase())

}

override suspend fun deleteBookWord(id: String) {

dao.deleteBookWord(id)

}

}

Работа с сетью происходила через всплывающий диалог, который вызывается по кнопке с иконкой переводчика на экране чтения книги в соответствии с рисунком 8.

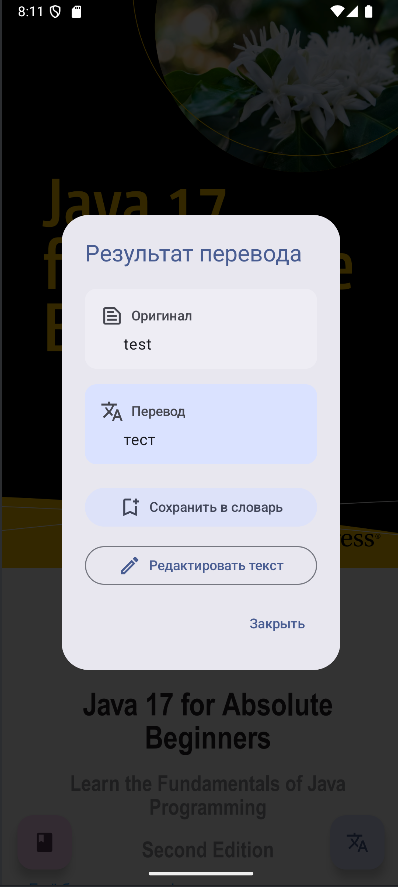


Рисунок 9 – Диалог для перевода текста

Приложение было разработано с помощью шаблона проектирования архитектуры Model-View-ViewModel (MVVM), благодаря чему интерфейс приложения имеет минимальную зависимость от реализации логики работы с данными. Данные приложение получало через эмулятор сервера Mockoon, благодаря чему можно было приступить к тестированию.

* 1. Тестирование

Тестирование приложения проводилось с помощью физического устройства Realme 11 Pro с версией операционной системы Android 14 и с помощью эмулятора c операционной системой 8.1. Тестирование приведено в приложении 3. В ходе тестирования было проверено соответствие поведения программы техническому заданию. В ходе проверки тестов были найдены упущенные из виду недочёты, которые в последствие были исправлены. В итоге, требования были выполнены, а работа приложения оказалась корректной и надёжной. Таким образом приложение готово для дальнейшего использования конечным пользователем.

* 1. Руководство пользователя

Приложение предоставляет возможность людям читать иностранные книги с встроенным переводчиком слов на основании искусственного интеллекта.

Приложение доступно для использования на устройствах с операционной системой Android не ниже версии 8.1. Для работы с приложением необходим доступ в сеть Internet. Также, чтобы получать уведомления о предстоящих забронированных событиях, нужно предоставить соответствующие разрешения приложению.

При первом запуске приложения открывается экран приветствия, представленная в соответствии с рисунком 9.

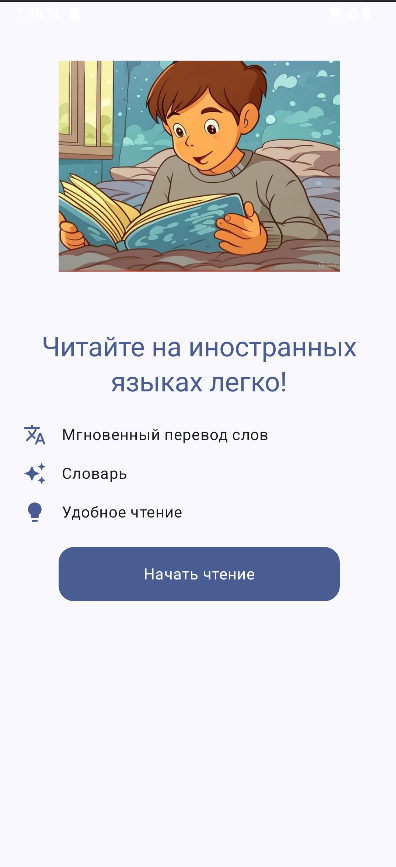


Рисунок 10 – Экран приветствия

На экране приветствия пользователь видит краткую информацию о приложении, для работы необходимо нажать кнопку «Начать чтение», которая в последствии откроет экран со списком книг в соответствии с рисунком 10.

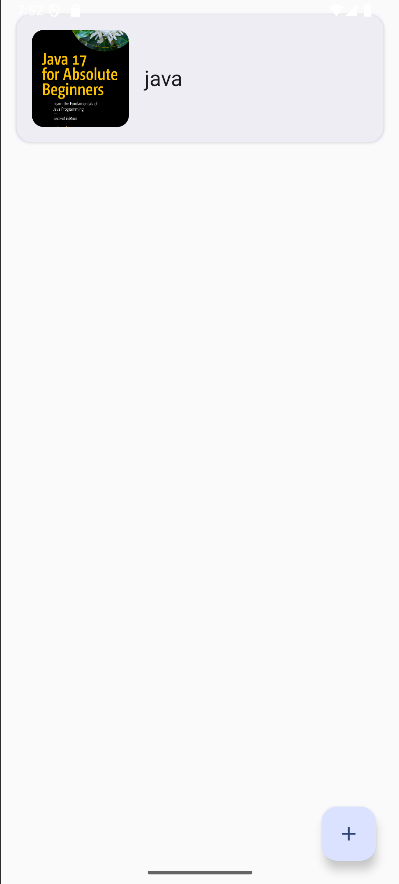


Рисунок 11 – Экран со списком книг

Каждый элемент списка книг отображает в себе название книги, а также его иконку. Для удаления записи нужно свайпнуть влево.

Для добавления книги пользователю необходимо нажать на кнопку с иконкой плюсика. По нажатию откроется экран добавления книги в соответствии с рисунком 11.

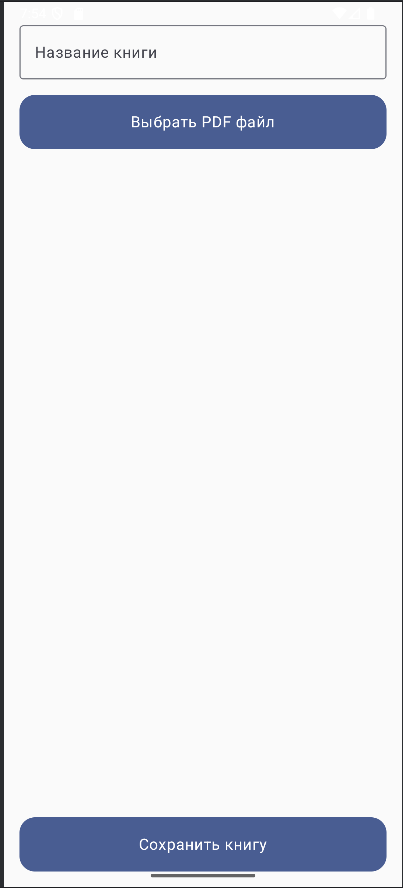


Рисунок 12 – Экран добавления книги

Необходимо вписать название, а также выбрать PDF файл в системе. Для сохранения необходимо нажать на кнопку «Сохранить книгу».

Для чтения книги необходимо выбрать книгу из списка. После выбора откроется режим чтения книги в соответствии с рисунком 12.

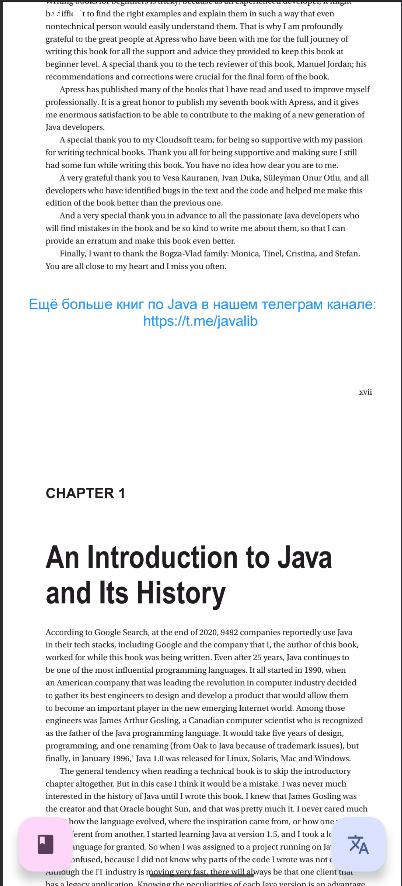


Рисунок 13 – Экран чтения книги

На экране чтения представлены две кнопки. Первая кнопка с иконкой переводчика нужна для того, чтобы переводить иностранные слова на русский язык. По нажатию откроется диалог, где пользователь может ввести иностранное слово или словосочетание и получить перевод в соответствии с рисунком 13.

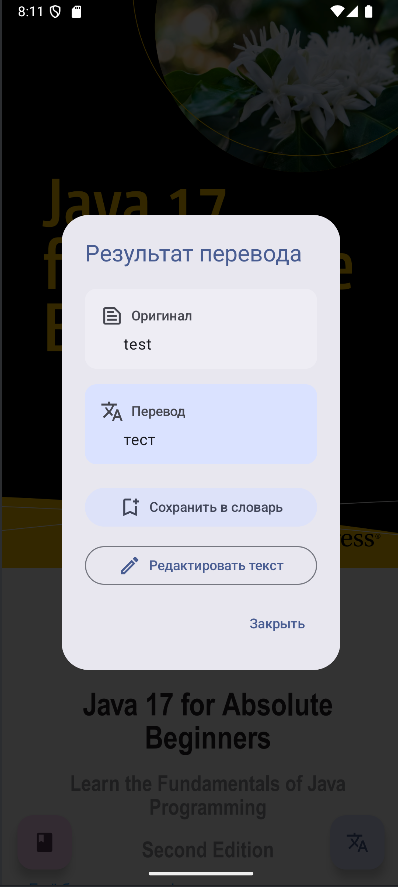


Рисунок 14 – Диалог результата перевода

В окне диалога также есть возможность редактировать текст для перевода и добавить новое словосочетание или слово в словарь. Словарь открывается по второй кнопке с иконкой книги в соответствии с рисунком 14.



Рисунок 15 – Словарь

В словаре хранятся сохраненные переводы по конкретной книге. Есть возможность удалять записи через свайп влево. Каждый элемент отображает оригинальное слово, его перевод, а также время, как давно оно было добавлено в словарь.

Для изучения новых словосочетаний необходимо нажать на кнопку «Изучать». По нажатию откроется экран с игрой, которая позволит закрепить добавленные фразы в соответствии с рисунком 16.

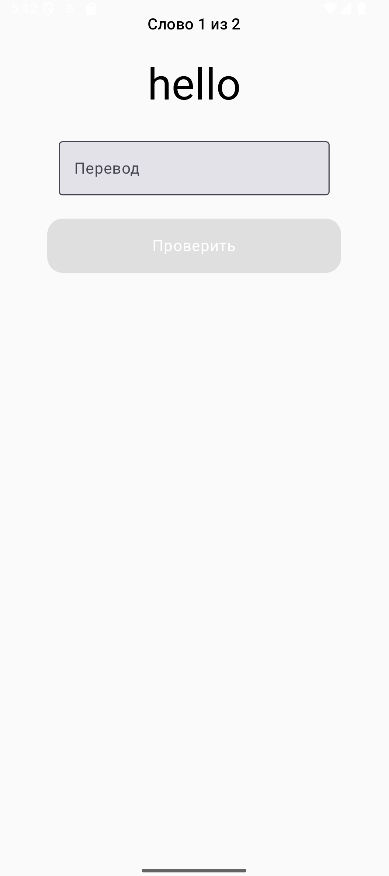


Рисунок 16 – Экран изучения слов

В игре выводится оригинальное слово и нужно написать его перевод на память. Если неправильно будет введен ответ, то система покажет правильный ответ и даст возможность перейти к следующему слову в соответствии с рисунком 17.

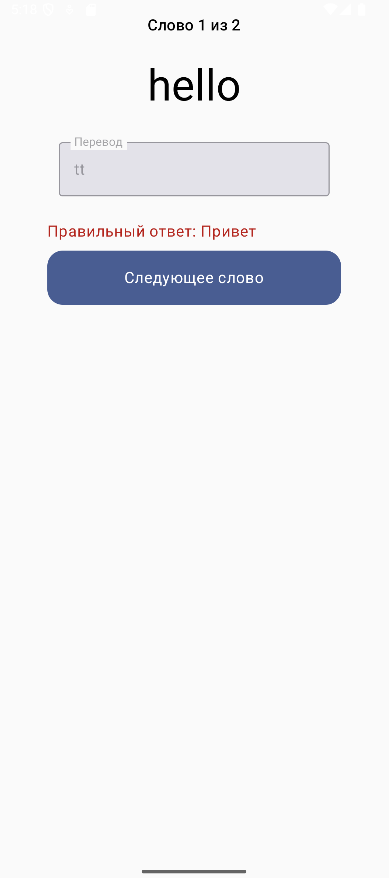


Рисунок 17 – Неправильно введенный ответ в игре

# технико-экономическое обоснование разработки

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.

1. Обеспечение безопасных условий труда программиста при разработке мобильного приложения для чтения иностранных книг

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[5]. Розетки и выключатели не должны иметь повреждений. Следует убедиться в свободных путях эвакуации, наличии исправных огнетушителей[2] 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[1]. После включения компьютера проверяется отсутствие сбоев, четкость изображения, яркость и контрастность[4]. 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[1]. Нельзя качаться на стуле или облокачиваться на технику. Рабочее место должно быть свободным от лишних предметов. Клавиатуру и мышь ежедневно необходимо дезинфицировать спиртовыми средствами[2].

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[4].

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[6].

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[6].

Каждый час работы сопровождается 10-15-минутным перерывом для гимнастики глаз в соответствии с рисунком 18.

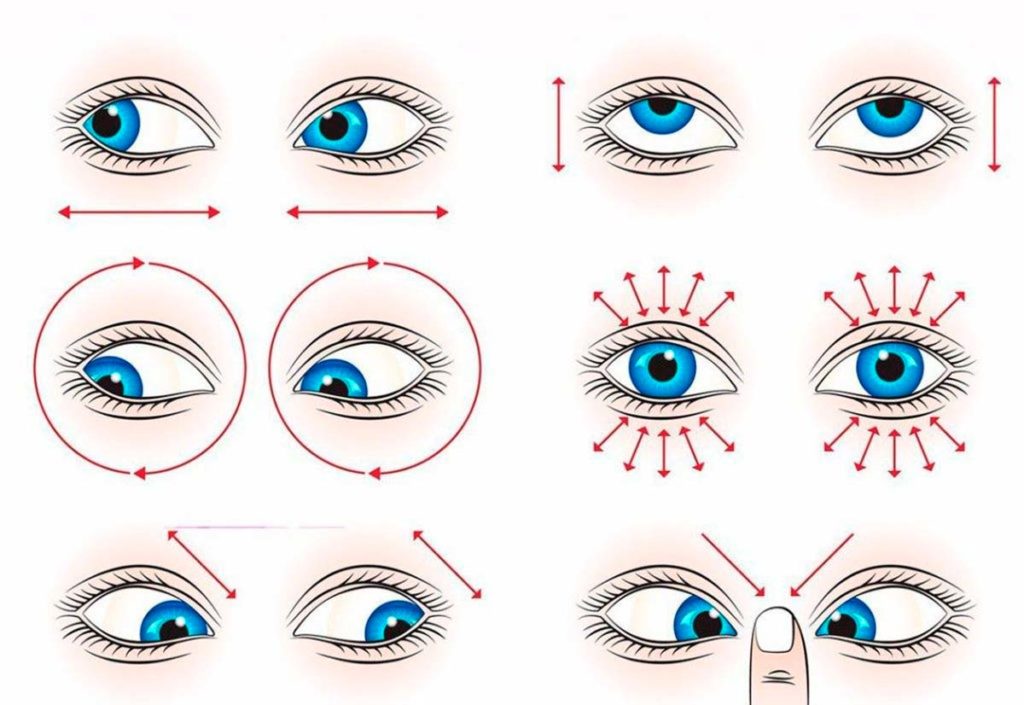


Рисунок 18 – Гимнастика для глаз

В соответствии с рисунком 19 делаются упражнения для профилактики утомления.



Рисунок 19 – 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[3].

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

Заключение

Целью дипломного проекта стала разработка мобильного приложения для интерактивного чтения иностранных текстов с интеграцией перевода на основе искусственного интеллекта. В ходе работы анализировались проблемы языкового погружения, изучались аналогичные сервисы, формулировались требования к адаптации контента и интерактивным функциям. Была спроектирована система хранения пользовательских данных и языковых материалов, а также реализовано приложение под Android с возможностью мгновенного перевода. После тестирования и корректировки интерфейсных недочётов повторная проверка подтвердила стабильность функций. Для удобства освоения создано руководство с примерами работы с текстами и инструментами анализа. Был проведён подсчёт затрат по созданию продукта и стоимости выполненной работы, в результате которого было выяснено, что проект экономически обоснован. Работа выполнялась в соответствии с правилами по обеспечению безопасных условий на рабочем месте программиста, благодаря чему проблем со здоровьем не возникло.

В итоге поставленная цель и задачи были выполнены в полном объёме. Разработанное приложение функционирует и имеет потенциал в будущем обзавестись пользователями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 50923-96. Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения. – Введ. 01.07.1997 – Москва: Стандартинформ, 2008. – 12 с.
2. СП 2.2.3670-20. Санитарно-эпидемиологические требования к условиям труда [Электронный ресурс]: утв. Гл. гос. санитар. врачом РФ 02.12.2020 – Введ. 01.01.2021 // «Кодекс: справ.-правовая система».
3. ГОСТ 12.1.005-88. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны. – Введ. 01.01.1990 – Москва: Издательство стандартов, 1989. – 18 с.
4. ГОСТ Р 50948-2001. Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности. – Введ. 01.07.2002 – Москва: Госстандарт России, 2001. – 284 с.
5. ГОСТ 24940-96. Здания и сооружения. Методы измерения освещенности. – Введ. 01.07.1997 – Москва: Стандартинформ, 1996. – 10 с.
6. GeekBrains кибербезопасность: официальный сайт. – URL: <https://gb.ru/blog/kiberbezopasnost-pravila/> (дата обращения: 04.05.2025). – Текст: электронный.