МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных технологий и прикладной математики

УТВЕРЖДАЮ:
Проректор по учебной работе,
качеству образования – первый
проректор
_____ Хагуров Т.А.
подпись
« 29 » августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1. О.34 Разработка мобильных приложений

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Профиль Искусственный интеллект и аналитика данных

Форма обучения очная

Квалификация бакалавр

Краснодар 2025

Рабочая программа дисциплины «Разработка мобильных приложений» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем.

Программу составил(и):

Подколзин В.В. доцент, канд. физ.-мат. наук, доцент _

Рабочая программа дисциплины «Разработка мобильных приложений» утверждена на заседании кафедры информационных технологий протокол №1 от «26» августа 2025г.
Заведующий кафедрой (разработчика)
В. В. Подколзин

подпись

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики протокол № 1 от «28» августа 2025г.
Председатель УМК факультета
А. В. Коваленко

подпись

Рецензенты:
Мостовой Евгений Викторович, генеральный директор ООО «Портал-Юг», e-mail: mostovoy@portal-yug.ru

Луценко Евгений Вениаминович, доктор экономических наук, кандидат технических наук, профессор кафедры компьютерных технологий и систем Федерального государственного бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина», e-mail: prof.lutsenko@gmail.com

1 Цели и задачи изучения дисциплины (модуля)

1.1 Цель освоения дисциплины

Основной целью дисциплины является изучение методов и технологий создания приложений для мобильных устройств, закрепить навыки объектно-ориентированного программирования, работы с базами данных и сетевого взаимодействия с поддержкой ИИ. Важным является формирование у студентов компетенций в области проектирования, разработки, тестирования и публикации современных нативных Android-приложений на языке Kotlin с активным использованием инструментов искусственного интеллекта (ИИ) для оптимизации процесса разработки, повышения качества кода и создания интеллектуальных функций приложений.

1.2 Задачи дисциплины

Основные задачи курса на основе системного подхода:

- ознакомление с приемами разработки приложений для мобильных устройств;
- приобретение навыков работы в среде Android Studio;
- совершенствование навыков разработки программного кода с использованием ИИ;
- совершенствование навыков тестирования программного кода, в т.ч. с использованием ИИ;
- совершенствование навыков доступа и манипулирования данными в СУБД SQLite;
- совершенствование навыков работы в компьютерных сетях по протоколу HTTP в формате JSON;
- совершенствование навыков объектно-ориентированного программирования на языке Java/Kotlin;
- приобретение навыков практической разработки мобильных приложений в среде Android Studio.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Разработка мобильных приложений» относится к «Обязательная часть» Блока 1 «Дисциплины (модули)» учебного плана.

Входными знаниями для освоения данной дисциплины являются знания, умения и опыт, накопленный студентами в процессе изучения дисциплины «Программирование», «Алгоритмы и структуры данных», «Базы данных», «Web-разработка», «Компьютерные сети», «Объектно-ориентированное программирование и шаблоны проектирования», «Параллельное и низкоуровневое программирование».

1.4 Профессиональные роли в структуре образовательной программы

Роль 1: **AI PM (Менеджер проектов ИИ)**

Продуктовый менеджер управляющий жизненным циклом ИИ-продуктов и координирующий междисциплинарные команды:

- Управление ИИ-проектами от идеи до внедрения
- Анализ бизнес-требований и постановка задач
- Координация работы технических и бизнес-команд
- Планирование ресурсов и контроль сроков

– Оценка эффективности и ROI ИИ-решений

Роль 2: **Data Analyst (Аналитик данных)**

Специалист по анализу данных извлечению инсайтов и построению аналитических моделей для поддержки бизнес-решений:

– Исследовательский анализ данных (EDA)
– Построение отчетов и дашбордов
– Статистический анализ и тестирование гипотез
– Создание прогнозных моделей
– Визуализация результатов для стейкхолдеров

Роль 3: **MLOps (Специалист по эксплуатации ИИ)**
DevOps-инженер специализирующийся на автоматизации и операционном управлении жизненным циклом ML-моделей:

– Автоматизация процессов обучения и развертывания моделей
– Мониторинг производительности ML-систем
– Управление версиями моделей и данных
– Обеспечение CI/CD для ML-проектов
– Оптимизация вычислительных ресурсов

Роль	**Определение**	**Трудовые действия**	**Фокус компетенций**	**Функция в ИИ-проекте**
AI PM (Менеджер проектов ИИ)	Продуктовый менеджер, управляющий жизненным циклом ИИ-продуктов и координирующий междисциплинарные команды	• Управление ИИ-проектами от идеи до внедрения • Анализ бизнес-требований и постановка задач • Координация работы технических и бизнес-команд • Планирование ресурсов и контроль сроков • Оценка эффективности и ROI ИИ-решений	Управление проектами, бизнес-анализ, координация команд	Управление процессами создания ИИ-решений, включая координацию команды разработки
Data Analyst (Аналитик данных)	Специалист по анализу данных извлечению инсайтов и построению аналитических моделей для поддержки бизнес-решений	• Исследовательский анализ данных (EDA) • Построение отчетов и дашбордов • Статистический анализ и тестирование	Статистический анализ, визуализация данных, предварительная обработка	Извлечение знаний из данных, построение аналитических моделей, использующих МО и ИИ

		гипотез • Создание прогнозных моделей • Визуализация результатов для стейкхолдеров		
MLOps (Специалист по эксплуатации ИИ)	DevOps-инженер, специализирующийся на автоматизации и операционном управлении жизненным циклом ML-моделей	• Автоматизация процессов обучения и развертывания моделей • Мониторинг производительности ML-систем • Управление версиями моделей и данных • Обеспечение CI/CD для ML-проектов • Оптимизация вычислительных ресурсов	DevOps для ML, автоматизация, мониторинг систем	Автоматизация и операционное управление жизненным циклом МО-моделей

1.5 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций:

ОПК-4 **Способен участвовать в разработке технической документации программных продуктов и программных комплексов**

ОПК-4.2 ***Способен применять стандарты, нормы и правила при оформлении технической документации на различных стадиях проектирования и поддержки жизненного цикла программных продуктов и программных комплексов***

Знать *Методы создания программного кода с использованием ИИ*
Стандарты оформления кода для используемых языков программирования Java / Kotlin

Современные решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio

Уметь *Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ*

Использовать существующие типовые решения и шаблоны проектирования программного обеспечения, в т.ч. с использованием ИИ

Владеть

Анализ возможностей реализации требований к программному обеспечению
Разработка структуры программного кода, в т.ч. с использованием ИИ
Деятельность, направленная на решение задач аналитического характера, предполагающих выбор и многообразие актуальных способов решения задач с использованием ИИ
Проектирование программных интерфейсов при разработке мобильных приложений, в т.ч. с использованием ИИ
Проектирование структур данных при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ

PL-2 **Способен применять JVM-совместимые языки программирования для решения задач в области ИИ**

PL-2.1 ***Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений***

Знать

Методы и средства проектирования программного обеспечения, в т.ч. с использованием ИИ
Стандарты оформления кода для используемых языков программирования Java / Kotlin
Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio с использованием ИИ
Методы тестирования и оценки качества программного кода с использованием ИИ
Модель памяти Java
Алгоритмы сборки мусора

Уметь

Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ
Осуществлять выбор инструментов разработки на JVM-совместимых языках, приемлемых для создания прикладной системы
Проводить оценку качества программного кода с использованием ИИ
Применять типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio с использованием ИИ
Методы тестирования и оценки качества программного кода с использованием ИИ
Оптимизировать сборку мусора

Владеть *Проектирование программных интерфейсов при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ*
Разработка, изменение архитектуры компьютерного программного обеспечения и ее согласование с системным аналитиком и архитектором программного обеспечения при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ
Использование ИИ инструментов при тестировании и оценки качества программного кода
Применяет основные библиотеки для решения рутинных задач в серверном программировании: ввод-вывод, применение простейших примитивов многопоточного программирования, интеграция с базами данных
Способен поддерживать приложения с высоким параллелизмом и конкуренцией

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 2 зач. ед. (72 часов), их распределение по видам работ представлено в таблице

Вид учебной работы	Всего часов	Семестры (часы)
		7
Контактная работа, в том числе:	**52,2**	**52,2**
Аудиторные занятия (всего):	**50**	**50**
Занятия лекционного типа	**16**	16
Лабораторные занятия	**34**	34
Иная контактная работа:		
Контроль самостоятельной работы (КСР)	**2**	2
Промежуточная аттестация (ИКР)	**0,2**	0,2
Самостоятельная работа, в том числе:	**19,8**	**19,8**
Проработка учебного (теоретического) материала	**10**	10
Выполнение индивидуальных заданий (подготовка сообщений, презентаций)	**9,8**	9,8
Общая трудоемкость **час.**	**72**	**72**
Общая трудоемкость **в том числе контактная работа**	**52,2**	**52,2**
Общая трудоемкость **зач. ед**	**2**	**2**

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.
Разделы (темы) дисциплины, изучаемые в 7 семестре

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	Введение в Android-разработку	10	2		2	2
2.	Kotlin для Android	10	2		2	2
3.	Основы UI/UX в Android	14	2		6	2
4.	Архитектура приложения.	12	2		4	2

5.	Работа с данными	16	2		6	3
6.	Сетевое взаимодействие	14	2		4	3
7.	Фоновые задачи	14	2		4	3
8.	Тестирование (Unit, UI), отладка, профилирование. ИИ-инструменты QA.	15,8	2		6	2,8
ИТОГО по разделам дисциплины		**69,8**	**16**		**34**	**19,8**
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,2				
Общая трудоемкость по дисциплине		**72**				

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины
2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1.	Введение в Android-разработку	Архитектура Android: компоненты ОС, жизненный цикл приложения. Android Studio, SDK, эмулятор. Gradle. Версионирование (Git). Обзор AI-ассистентов для генерации шаблонов проектов.	К, Т
2.	Kotlin для Android	Kotlin vs Java: преимущества (Null safety, extension functions, data classes, sealed classes, delegated properties. Лямбды, higher-order functions, inline functions. DSL). Использование AI-ассистентов для: генерации шаблонов кода (классы, функции);	К, Т

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
		автодополнения и предсказания кода; рефакторинга (упрощение, оптимизация); объяснения сложного кода или ошибок; генерации документации (KDoc).	

3.	Основы UI/UX в Android	Верстка XML: View, ViewGroup, ConstraintLayout. Ресурсы приложения: строки, стили, темы. Material Design: компоненты (Button, TextView, CardView). Генерация XML-разметки через текстовые описания. View System vs. Jetpack Compose. Модификаторы. State в Compose. Навигация (Navigation Component). Адаптивные макеты для разных экранов. Доступность. Генерация прототипов UI/макетов по описанию (Figma AI, Uizard, Galileo AI). Конвертация дизайн-макетов (Figma, Sketch) в приближенный код Compose.	К, Т
4.	Архитектура приложения.	Проблемы монолитной архитектуры. Принципы SOLID, CLEAN. ViewModel, LiveData/StateFlow. Паттерны MVVM, MVI (обзор). Жизненный цикл компонентов (Lifecycle-Aware Components). DI: внедрение зависимостей (Hilt/Koin). Рефакторинг кода под MVVM. Intents: явные и неявные. Навигация: Navigation Component. Автогенерация обработчиков событий.	К, Т
5.	Работа с данными	SharedPreferences: хранение ключе-значение. Локальное хранилище: Room (ORM), DAO, миграции. Асинхронность: Kotlin Coroutines, Flow/StateFlow/SharedFlow. Генерация моделей Room Entity/DAO из схемы БД с помощью ИИ-ассистентов. Генерация шаблонного кода для асинхронных операций (Coroutines/Flow). Корректировка SQL-запросов с помощью ИИ.	К, Т
6.	Сетевое взаимодействие	REST API: принципы работы. Retrofit, JSON (Moshi/Gson). Обработка ошибок. Генерация моделей данных (Data Classes) и Retrofit интерфейсов по JSON-схеме/ответу API.	К, Т

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4

7.	Фоновые задачи	Сервисы. WorkManager. Широковещательные сообщения. Доступ к сенсорам, камере, геолокации, уведомлениям. Разрешения (Permissions). Использование ИИ для генерации boilerplate кода работы с API камеры/геолокации или настройки WorkManager.	К, Т
8.	Тестирование (Unit, UI), отладка, профилирование. ИИ-инструменты QA.	Юнит-тесты (JUnit, MockK), Инструментальные (UI) тесты (Espresso, Compose Testing). Отладка (Android Studio Debugger, Logcat). Профилирование (CPU, Memory, Energy). Генерация юнит-тестов (Copilot, JetBrains AI Assistant, Testim). Генерация сценариев UI-тестов (Appium, Testim, Applitools - с ИИ-элементами). ИИ-анализ логов (Logcat) для выявления закономерностей сбоев.	К, Т

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.2 Занятия семинарского типа

Не предусмотрено

2.3.3 Лабораторные занятия

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
1.	Введение в Android-разработку	Изучение среды Android Studio. Подключение и эмуляция мобильных устройств. Создание простого приложения. Активировать Copilot/JetBrains AI Assistant в IDE. Создать проект "HelloWorld" с Empty Activity (Compose). С помощью ИИ сгенерировать простое приложения с элементами управления.	Т, РЗ
2.	Kotlin для Android	Приложение содержащее несколько активностей. Переход между активностями. Обмен данными между активностями. Рассмотрение кейса взаимодействия активностей в Android Studio. Рассмотрение кейса взаимодействий приложений в Android Studio. Создать Data class, Sealed class.	Т, РЗ

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4

		Генерации extension-функций с помощью ИИ.	
3.	Основы UI/UX в Android	Создание приложения Calculator в Android Studio. Сгенерировать в AI-ассистенте (Figma+Galileo) Макет по текстовому описанию. Экспортировать дизайн в XML. Использовать плагин Figma-to-Compose для генерации кода. Доработать UI с помощью ИИ-подсказок. Генерация прототипов UI/макетов по описанию (Figma AI, Uizard, Galileo AI). Конвертация дизайн-макетов (Figma, Sketch) в приближенный код Compose (экспериментальные плагины/инструменты).	Т, РЗ
4.	Архитектура приложения.	Построить архитектуру приложения с использованием ИИ-шаблонов. Data class NewsItem. Базовый ViewModel . Генерация StateFlow/LiveData бойлерплейта и шаблон Observer для Compose с помощью ИИ.	Т, РЗ
5.	Работа с данными	Организация списков. Преставление элементов списка. Описать схему БД. Генерировать Entity/DAO по описанию, миграция при изменении схемы с помощью ИИ. Анализ сгенерированного кода. Генерация шаблонного кода для асинхронных операций (Coroutines/Flow).	Т, РЗ
6.	Сетевое взаимодействие	Интегрировать REST API с автоматизацией ИИ. Генерация data-классов по JSON-ответу. Создания Retrofit-интерфейса. Загрузка данных. Обработка ошибок через ИИ-подсказки.	Т, РЗ
7.	Фоновые задачи	Реализовать доступ к аппаратным функциям с ИИ. Разрешения. Использование ИИ для генерации boilerplate кода работы с API камеры/геолокации или настройки WorkManager. Сохранение фото. Геолокация. Интеграция TensorFlow Lite.	Т, РЗ
8.	Тестирование (Unit, UI), отладка, профилирование. ИИ-инструменты QA.	Генерация тест-кейсов (Testim). Создание юнит-тестов (Copilot, JetBrains AI Assistant, Testim). Генерация сценариев UI-тестов (Appium, Testim, Applitools - с ИИ-элементами). Запуск тестов. ИИ-анализ логов (Logcat) для выявления	Т, РЗ

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
		закономерностей сбоев. Полнота тестирования.	

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.4 Примерная тематика курсовых работ (проектов)

Не предусмотрено.

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	2	3
1	Изучение теоретического материала	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019
2	Решение задач	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа,

– в форме аудиофайла,

– в печатной форме на языке Брайля.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа,

– в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии

В соответствии с требованиями ФГОС в программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; метод малых групп, разбор практических задач и кейсов.

При обучении используются следующие образовательные технологии:

- Технология коммуникативного обучения – направлена на формирование

коммуникативной компетентности студентов, которая является базовой, необходимой для адаптации к современным условиям межкультурной коммуникации.

– Технология разноуровневого (дифференцированного) обучения – предполагает осуществление познавательной деятельности студентов с учётом их индивидуальных способностей, возможностей и интересов, поощряя их реализовывать свой творческий потенциал. Создание и использование диагностических тестов является неотъемлемой частью данной технологии.

– Технология модульного обучения – предусматривает деление содержания дисциплины на достаточно автономные разделы (модули), интегрированные в общий курс.

– Информационно-коммуникационные технологии (ИКТ) - расширяют рамки образовательного процесса, повышая его практическую направленность, способствуют интенсификации самостоятельной работы учащихся и повышению познавательной активности. В рамках ИКТ выделяются 2 вида технологий:

– Технология использования компьютерных программ – позволяет эффективно дополнить процесс обучения языку на всех уровнях.

– Интернет-технологии – предоставляют широкие возможности для поиска информации, разработки научных проектов, ведения научных исследований.

– Технология индивидуализации обучения – помогает реализовывать личностно-ориентированный подход, учитывая индивидуальные особенности и потребности учащихся.

– Проектная технология – ориентирована на моделирование социального взаимодействия учащихся с целью решения задачи, которая определяется в рамках профессиональной подготовки, выделяя ту или иную предметную область.

– Технология обучения в сотрудничестве – реализует идею взаимного обучения, осуществляя как индивидуальную, так и коллективную ответственность за решение учебных задач.

– Игровая технология – позволяет развивать навыки рассмотрения ряда возможных способов решения проблем, активизируя мышление студентов и раскрывая личностный потенциал каждого учащегося.

– Технология развития критического мышления – способствует формированию разносторонней личности, способной критически относиться к информации, умению отбирать информацию для решения поставленной задачи.

Комплексное использование в учебном процессе всех вышеназванных технологий стимулируют личностную, интеллектуальную активность, развивают познавательные процессы, способствуют формированию компетенций, которыми должен обладать будущий специалист.

Основные виды интерактивных образовательных технологий включают в себя:

– работа в малых группах (команде) - совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путём творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности;

– проектная технология - индивидуальная или коллективная деятельность по отбору, распределению и систематизации материала по определенной теме, в результате которой составляется проект;

– анализ конкретных ситуаций - анализ реальных проблемных ситуаций, имевших место в соответствующей области профессиональной деятельности, и поиск вариантов лучших решений;

– развитие критического мышления – образовательная деятельность, направленная на развитие у студентов разумного, рефлексивного мышления, способного выдвинуть новые идеи и увидеть новые возможности.

Подход разбора конкретных задач и ситуаций широко используется как преподавателем, так и студентами во время лекций, лабораторных занятий и анализа результатов самостоятельной работы. Это обусловлено тем, что при исследовании и решении каждой конкретной задачи имеется, как правило, несколько методов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

Семестр	Вид занятия	Используемые интерактивные образовательные технологии	количество интерактивных часов
7	Л, ЛР	Практические занятия в режимах взаимодействия «преподаватель – студент» и «студент – студент»	16
Итого			**16**

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

Темы, задания и вопросы для самостоятельной работы призваны сформировать навыки поиска информации, умения самостоятельно расширять и углублять знания, полученные в ходе лекционных и практических занятий.

Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «название дисциплины».

Оценочные средства включает контрольные материалы для проведения текущего контроля в форме тестовых заданий, заданий по темам и промежуточной аттестации в форме вопросов и заданий к зачету.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	Введение в Android-разработку	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 1-8 Типовые тестовые задания 1 Типовые контрольные задания 1-5	Теоретический вопрос 1-2 Задание для самостоятельной работы
2	Kotlin для Android	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 8-20 Типовые тестовые задания 2 Типовые контрольные задания 6-16	Теоретический вопрос 2-3 Задание для самостоятельной работы
3	Основы UI/UX в Android	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 21-30 Типовые тестовые задания 3	Теоретический вопрос 5-7 Задание для самостоятельной работы

			Типовые контрольные задания 17-26	
4	Архитектура приложения.	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 31-40 Типовые тестовые задания 4-6 Типовые контрольные задания 27-35	Теоретический вопрос 8-10 Задание для самостоятельной работы
5	Работа с данными	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 41-50 Типовые тестовые задания 7-8 Типовые контрольные задания 36-44	Теоретический вопрос 11-14 Задание для самостоятельной работы
6	Сетевое взаимодействие	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 51-60 Типовые тестовые задания 9-10 Типовые контрольные задания 45-53	Теоретический вопрос 15-16 Задание для самостоятельной работы
7	Фоновые задачи	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 61-70 Типовые тестовые задания 11 Типовые контрольные задания 54-62	Теоретический вопрос 17-18 Задание для самостоятельной работы

8	Тестирование (Unit, UI), отладка, профилирование. ИИ-инструменты QA.	ОПК-4.1; PL-2.1	Типовые контрольные вопросы 71-80 Типовые тестовые задания 8 Типовые контрольные задания 63-73	Теоретический вопрос 19-20 Задание для самостоятельной работы

Показатели, критерии и шкала оценки сформированных компетенций

Соответствие освоения компетенций планируемым результатам обучения и критериям их оценивания (уровень: **продвинутый**; оценка: **зачтено**):

ОПК-4 **Способен участвовать в разработке технической документации программных продуктов и программных комплексов**

ОПК-4.2 ***Способен применять стандарты, нормы и правила при оформлении технической документации на различных стадиях проектирования и поддержки жизненного цикла программных продуктов и программных комплексов***

Знать

Методы создания программного кода с использованием ИИ

Стандарты оформления кода для используемых языков программирования Java / Kotlin

Современные решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio

Уметь

Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ

Использовать существующие типовые решения и шаблоны проектирования программного обеспечения, в т.ч. с использованием ИИ

Владеть

Анализ возможностей реализации требований к программному обеспечению

Разработка структуры программного кода, в т.ч. с использованием ИИ

Деятельность, направленная на решение задач аналитического характера, предполагающих выбор и многообразие актуальных способов решения задач с использованием ИИ

Проектирование программных интерфейсов при разработке мобильных приложений, в т.ч. с использованием ИИ

Проектирование структур данных при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ

PL-2 **Способен применять JVM-совместимые языки программирования для решения задач в области ИИ**

PL-2.1 ***Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с***

использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений

Знать

Методы и средства проектирования программного обеспечения, в т.ч. с использованием ИИ
Стандарты оформления кода для используемых языков программирования Java / Kotlin
Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio с использованием ИИ
Методы тестирования и оценки качества программного кода с использованием ИИ
Модель памяти Java
Алгоритмы сборки мусора

Уметь

Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ
Осуществлять выбор инструментов разработки на JVM-совместимых языках, приемлемых для создания прикладной системы
Проводить оценку качества программного кода с использованием ИИ
Применять типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения при разработке мобильных приложений в среде Android Studio с использованием ИИ
Методы тестирования и оценки качества программного кода с использованием ИИ
Оптимизировать сборку мусора

Владеть

Проектирование программных интерфейсов при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ
Разработка, изменение архитектуры компьютерного программного обеспечения и ее согласование с системным аналитиком и архитектором программного обеспечения при разработке мобильных приложений в среде Android Studio, в т.ч. с использованием ИИ
Использование ИИ инструментов при тестировании и оценки качества программного кода
Применяет основные библиотеки для решения рутинных задач в серверном программировании: ввод-вывод, применение простейших примитивов многопоточного программирования, интеграция с базами данных
Способен поддерживать приложения с высоким параллелизмом и конкуренцией

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Типовые тестовые задания

1. Укажите, что относится к системным ресурсам Android-приложения:
   - grandle.propertities
   - activity_main.xml

- AndroidManifest.xml
- java
- colors.xml
- MainActivity
- mipmap
- styles.xml
- build.grandle

2. Укажите порядок вызова обработчиков событий в порядке жизненного цикла активности :
   1) onResume()
   2) onCreate()
   3) onStop()
   4) onStart()
   5) onDestroy()
   6) onPause()

3. Укажите, что относится к визуальным элементам пользовательского интерфейса:
   - ChipGroup
   - Intent
   - Drawable
   - CheckBox
   - Manifest
   - TextView
   - Activity
   - TabItem
   - Button
   - Plain Text

4. Укажите какие атрибуты и методы относятся только к меню (основному):
   - inflate()
   - android:title
   - onOptionsItemSelected()
   - setShowAsAction
   - android:id
   - registerForContextMenu()
   - android:showAsAction

5. Укажите какие из пруведенных видов навигации относятся к навигации на основе списков и сеток:
   - Carousel
   - Simple Buttons
   - Swipe Views
   - Dashboard
   - List
   - Tabs
   - Grid

6. Укажите из обработчиков событий относятся **только** к жизненному циклу фрагмента:

- onActivityCreated()
- onResume()
- onDestroy()
- onStart()
- onPause()
- onAttach()
- onCreate()
- onStop()
- onDestroyView()

7. Укажите какие из характеристик относятся к характеристике внешней памяти:
   - может быть в собственной памяти устройства или на внешнем носителе
   - энергозависимая
   - пользователь может явно разрешить другим приложениям доступ к файлам
   - доступна не всегда
   - сохраненные данные в памяти позволяют читать и записывать файлы
   - для доступа требуется разрешение, устанавливаемое в файле манифеста приложения
   - файлы могут быть доступны только данному приложению
   - хранилища доступны для чтения везде
   - хранятся apk-файлы, данные приложений, медиафайлы, документы и пр.
   - доступность памяти должна проверяться

8. Укажите порядок действий при работе с базой данных SQLite:
   1) вызывается метод onUpgrade()
   2) проверить, существует ли база данных
   3) определить класс контракта
   4) создать базу данных, создать таблицы
   5) Создать наследника класса SQLiteOpenHelper
   6) проверить версию
   7) Создать наследника класса SQLiteOpenHelper
   8) открыть базу данных
   9) Если файл базы данных не существует, то он создается

9. Укажите какие из методов AsyncTask не взаимодействуют с основным потоком приложения :
   - onPreExecute()
   - doInBackground()
   - onPostExecute()
   - onProgressUpdate()

10. Укажите какие из утверждений справедливы для фоновых служб:
    - не имеют пользовательского интерфейса
    - работают в фоновом режиме
    - предназначены для выполнения разовых операций
    - со службой могут связываться только приложение, его создавшее
    - более высокий приоритет, чем бездействующим активностям
    - могут контролироваться из других сервисов

11. Укажите какие из утверждений справедливы для автономного приемника:
    - объявлен в манифесте
    - используют асинхронные API
    - не могут запускать активности
    - не могут запускать службы
    - регистрируется в коде приложения
    - является объектом BroadcastReceiver

Типовые контрольные вопросы:

1. Какие основные компоненты проекта на Android с использованием Jetpack Compose перечислены в build.gradle.kts? (Укажите 3)
2. Какую команду в Android Studio используют для активации Copilot/JetBrains AI Assistant?
3. Какой шаблон Activity в мастере создания проекта следует выбрать для работы с Compose?
4. Укажите последовательность действий для запуска приложения на эмуляторе.
5. Какие ключевые слова наиболее эффективны для промпта ИИ при генерации кнопки "Submit" в Compose?
6. Опишите преимущества Jetpack Compose перед традиционным XML-подходом.
7. Перечислите основные этапы настройки эмулятора Android.
8. Объясните назначение функции @Composable.
9. Опишите, как ИИ-ассистент (Copilot/JetBrains AI) может ускорить создание базового UI.
10. Какие основные различия между Empty Activity (View) и Empty Activity (Compose)?
11. Какой модификатор (val/var) следует использовать для объявления параметра data class User?
12. Укажите ключевое слово Kotlin для создания Sealed class, представляющей состояния загрузки данных (Loading, Success, Error).
13. Какой метод Activity используется для явного запуска другой активности с передачей данных?
14. Для чего предназначен метод putExtra() объекта Intent?
15. Какой тип промпта для ИИ эффективен для генерации extension-функции String.isValidEmail()?
16. Опишите назначение и преимущества data class в Kotlin для Android-разработки.
17. Объясните концепцию Sealed class (sealed hierarchy) и приведите пример использования для представления состояния UI.
18. Опишите механизм передачи данных между активностями с использованием Intent и Bundle.
19. Какие преимущества дают extension-функции? Приведите пример.
20. Как ИИ может помочь в создании ViewModel для новой активности?
21. Какой инструмент ИИ (Figma AI / Uizard / Galileo AI) используется для генерации прототипа UI по текстовому описанию?
22. Какой плагин позволяет конвертировать Figma-макет в приближенный код Jetpack Compose?
23. Укажите основной Composable-компонент для создания вертикального списка в Compose.
24. Какой модификатор (Modifier) чаще всего используется для задания отступов (padding) и выравнивания (align) элемента?
25. Какое свойство TextField в Compose отвечает за обработку изменения введенного текста?
26. Опишите процесс создания UI в Compose: от текстового описания функционала до

готового кода (с использованием ИИ-инструментов).
27. Перечислите основные преимущества декларативного подхода Compose.
28. Объясните назначение и использование модификаторов (Modifier) в Compose.
29. Как управлять состоянием UI (например, текстом в TextField) в Compose?
30. Какие ограничения есть у текущих ИИ-инструментов для генерации кода Compose из дизайн-макетов?
31. Какой компонент Jetpack (ViewModel / LiveData / Flow) рекомендуется использовать для хранения UI-состояния в приложениях с Compose?
32. Укажите функцию Compose для наблюдения за изменением StateFlow/LiveData (collectAsState() / observeAsState()).
33. Какой шаблон проектирования реализует ViewModel (MVC / MVP / MVVM / MVI)?
34. Для чего нужна аннотация @HiltViewModel?
35. Какой запрос к ИИ эффективен для генерации шаблона ViewModel с StateFlow для экрана списка новостей?
36. Опишите роль ViewModel в архитектуре приложения с использованием Compose.
37. Объясните разницу между MutableStateFlow и StateFlow. Как они связаны?
38. Как реализуется паттерн "Наблюдатель" (Observer) между ViewModel (источник состояния) и Composable-функцией (наблюдатель)?
39. Опишите преимущества использования StateFlow/SharedFlow + Compose перед LiveData в новых проектах.
40. Как ИИ может помочь в создании базовой структуры ViewModel и обработки состояний (Loading, Success, Error)?
41. Какая аннотация Room отмечает класс как сущность БД (@Entity / @Dao / @Database)?
42. Укажите аннотацию Room для метода DAO, выполняющего вставку (@Insert / @Update / @Query).
43. Какой диспетчер Kotlin Coroutines (Dispatchers.IO / Dispatchers.Main / Dispatchers.Default) следует использовать для операций с БД Room?
44. Какой тип SQL-запроса генерируется для DAO-метода с аннотацией @Query("SELECT * FROM items")?
45. Какой промпт ИИ использовать для генерации @Entity и @Dao для сущности Task (id, title, isCompleted)?
46. Опишите основные компоненты библиотеки Room (Entity, DAO, Database).
47. Объясните, почему операции с БД Room необходимо выполнять вне основного потока. Как Kotlin Coroutines решают эту проблему?
48. Как реализуется миграция схемы БД в Room? Как ИИ может помочь в этом процессе?
49. Опишите разницу между suspend функциями в DAO и возвратом Flow<List<T>>.
50. Какие риски связаны с использованием ИИ для генерации кода доступа к данным? Как их минимизировать?
51. Какая библиотека является стандартом де-факто для REST-запросов в Android (Retrofit / Volley / HttpURLConnection)?
52. Какой конвертер Retrofit используется для работы с JSON (Gson / Moshi / Jackson)?
53. Укажите аннотацию Retrofit для GET-запроса (@GET / @POST / @PUT).
54. Какой тип промпта ИИ эффективен для генерации data-класса по JSON-ответу вида { "id": 1, "name": "Alice" }?
55. Как обрабатываются сетевые ошибки в Retrofit с использованием Kotlin Coroutines (try/catch / Response обертка / Callback)?
56. Опишите процесс настройки Retrofit в проекте Android (зависимости, создание экземпляра, определение интерфейса API).

57. Объясните преимущества использования data-классов (Kotlin) для парсинга JSON.
58. Как ИИ может ускорить создание интерфейса API для Retrofit и соответствующих data-классов?
59. Опишите стратегии обработки сетевых ошибок (HTTP коды, исключения сети/таймаута) в ViewModel.
60. Почему важно выполнять сетевые запросы в фоновом потоке? Как это обеспечивается с помощью Coroutines?
61. Какой компонент WorkManager используется для определения периодической работы (PeriodicWorkRequest / OneTimeWorkRequest)?
62. Укажите разрешение (uses-permission) в манифесте, необходимое для доступа к точной геолокации (ACCESS_FINE_LOCATION / CAMERA / INTERNET).
63. Какой API используется для запроса разрешений во время выполнения (Runtime Permissions) в Compose (rememberLauncherForActivityResult() / ActivityResultContracts.RequestPermission())?
64. Какой промпт ИИ использовать для генерации шаблона кода открытия камеры через Intent?
65. Для чего нужен TensorFlow Lite в мобильной разработке?
66. Опишите назначение и преимущества библиотеки WorkManager для выполнения фоновых задач.
67. Объясните модель разрешений в Android (install-time / runtime). Как запросить опасные разрешения (dangerous permissions)?
68. Как ИИ может помочь в генерации шаблонного кода для работы с камерой или геолокацией?
69. Опишите процесс сохранения фото, полученного от камеры, во внутреннее или внешнее хранилище.
70. Какие основные сценарии использования TensorFlow Lite на мобильных устройствах?
71. Какой инструмент ИИ (Testim / Copilot / Appium) генерирует сценарии UI-тестов?
72. Укажите библиотеку для модульного тестирования в Android (JUnit / Espresso / UI Automator).
73. Какой инструмент профилирования в Android Studio показывает использование ЦП, памяти, сети и батареи (Profiler / Logcat / Layout Inspector)?
74. Какой фильтр в Logcat используется для поиска сообщений об ошибках (tag:MyTag / level:ERROR / package:my.app.package)?
75. Какой тип промпта ИИ эффективен для генерации юнит-теста функции String.isValidEmail()?
76. Опишите различия между модульным (Unit), интеграционным и UI-тестированием.
77. Как ИИ-инструменты (Testim, Applitools) помогают в создании и поддержке UI-тестов?
78. Объясните, как использовать Android Studio Profiler для выявления утечек памяти.
79. Как анализировать логи (Logcat) для поиска причины сбоя приложения? Как ИИ может помочь в этом анализе?
80. Что такое "полнота тестирования" (test coverage) и как ее измерить? Как ИИ может предложить сценарии для увеличения покрытия?

Типовые контрольные задания:

1. Сравните возможности Copilot и JetBrains AI Assistant. Составьте таблицу: поддерживаемые языки (Kotlin/Java), генерация кода Compose, объяснение кода, поиск ошибок, рефакторинг, интеграция с IDE. Выберите предпочтительный инструмент и обоснуйте.
2. Используя ИИ-ассистент, сгенерируйте код Composable-функции для экрана "О

себе". Экран должен содержать: ваше фото (заглушка Image), имя (Text), краткую биографию (Text), список навыков (LazyColumn с Chip). Доработайте сгенерированный код (цвета, шрифты, отступы).

3. Установите и настройте эмулятор для устройства с нестандартными характеристиками (например, складной экран, разрешение 1080x1920, API 34). Сделайте скриншоты работающего на нем сгенерированного в ЛР1 приложения "HelloWorld".
4. Напишите 5 разных промптов для ИИ с целью сгенерировать кнопку "Отправить" в Compose. Проанализируйте различия в сгенерированном коде (внешний вид, модификаторы, обработка клика). Определите наиболее эффективный промпт.
5. Исследуйте файл build.gradle.kts проекта Compose. Объясните назначение зависимостей androidx.compose.*, org.jetbrains.kotlinx:kotlinx-coroutines-android, androidx.lifecycle:lifecycle-runtime-ktx. Как ИИ может помочь в добавлении новых зависимостей?
6. Создайте намеренно "сломанную" версию приложения "HelloWorld" (например, удалите setContent, используйте устаревший компонент). Попросите ИИ-ассистент диагностировать и исправить ошибку. Задокументируйте процесс.
7. Найдите небольшой пример кода UI на XML (например, кнопка с текстом). Используя ИИ, сгенерируйте эквивалентный код на Jetpack Compose. Сравните подходы.
8. Запустите базовое приложение Compose в Android Studio Profiler. Зафиксируйте показатели использования CPU, памяти и энергии при запуске. С помощью ИИ предложите 1-2 способа потенциальной оптимизации (даже если они не нужны для простого приложения).
9. С помощью ИИ сгенерируйте data class для представления "Товара" (id, название, описание, цена, рейтинг, список URL изображений). Добавьте в класс метод formattedPrice(): String, форматирующий цену (например, "1 999 ₽").
10. Создайте sealed class Result<T> с состояниями Loading, Success(data: T), Error(message: String, code: Int). Используя ИИ, сгенерируйте функцию-расширение Result<T>.handle(onLoading: () -> Unit, onSuccess: (T) -> Unit, onError: (String, Int) -> Unit).
11. Реализуйте приложение с 3 экранами (активностями/fragments): Список -> Детали -> Редактирование. Передавайте объект data class Product между всеми экранами (в обе стороны для Редактирования). Используйте Parcelable или Serializable (генерацию Parcelable можно запросить у ИИ).
12. Создайте приложение с кнопкой "Открыть ссылку". По клику сформируйте корректный Intent для открытия URL в браузере. Используйте ИИ для подсказки по обработке случая, если браузер не установлен.
13. Исследуйте флаги Intent.FLAG_ACTIVITY_* (NEW_TASK, CLEAR_TOP, SINGLE_TOP). С помощью ИИ сгенерируйте примеры кода, демонстрирующие разницу в поведении навигации при использовании каждого флага. Создайте простые экраны для тестирования.
14. Опишите ИИ функционал экрана "Профиль пользователя" (загрузка данных, отображение имени, email, аватара, кнопка "Редактировать"). Попросите сгенерировать каркас ProfileViewModel с использованием StateFlow и sealed class для состояния.
15. Создайте extension-функцию для String (или Context), которая будет возвращать строку из ресурсов по ее имени (String resource ID). Попросите ИИ помочь с реализацией: fun Context.getStringResourceByName(resName: String): String?.
16. Создайте проект "HelloAI" с Empty Activity (Compose). С помощью ИИ-ассистента сгенерируйте код экрана, содержащего: заголовок `Text`, поле ввода (`TextField`) для

имени и кнопку (Button) "Приветствовать". При нажатии на кнопку должно появляться приветствие (Text) вида "Привет, [Имя]!".

17. Дайте подробное текстовое описание экрана "Настройки" (заголовок, переключатель "Темная тема", список "Язык", "Уведомления", кнопка "Сохранить"). Сгенерируйте прототип в Figma AI, Galileo AI или Uizard. Экспортируйте результат (скриншот/файл).
18. Возьмите сгенерированный в ЛР3 или задании 1 макет Figma. Используйте плагин Figma-to-Compose (или аналогичный). Проанализируйте сгенерированный код. Какие части работают хорошо? Что выглядит избыточно или некорректно? Составьте отчет.
19. Реализуйте экран "Настройки" из задания 1 в Compose **вручную**, используя сгенерированный ИИ макет как референс. Сравните результат с кодом, сгенерированным плагином.
20. Возьмите код простого, но "некрасивого" экрана Compose (можно сгенерировать заведомо плохой код ИИ или написать самому). Попросите ИИ-ассистент отрефакторить его: улучшить отступы, выравнивание, использовать рекомендованные Material 3 компоненты, добавить цветовую схему.
21. Имея реализованный экран Compose в светлой теме, попросите ИИ помочь сгенерировать код для поддержки темной темы. Реализуйте переключение темы с помощью MaterialTheme и сохранение состояния в DataStore (шаги по DataStore можно запросить у ИИ).
22. Опишите ИИ анимацию: "Плавное увеличение кнопки при нажатии с 100% до 105% и обратно при отпускании, длительность 300мс". Сгенерируйте код анимации в Compose (animate*AsState) и интегрируйте ее в вашу кнопку.
23. Создайте экран списка (LazyColumn). Используя ИИ-подсказки, реализуйте разное отображение элементов списка в портретной и ландшафтной ориентации (например, Grid в ландшафте). Используйте BoxWithConstraints или ориентацию.
24. Возьмите реализованный экран. Попросите ИИ проанализировать код на предмет доступности (семантические свойства Modifier.semantics, контрастность, размеры текста, лейблы для интерактивных элементов). Реализуйте 3 ключевых улучшения, предложенных ИИ.
25. Создайте приложение с двумя экранами (активностями). На первом экране введите имя и фамилию. По кнопке "Далее" передайте данные во вторую активность и отобразите их там в формате "Фамилия, Имя". Используйте data class User для передачи данных. Сгенерируйте с помощью ИИ extension-функцию User.getFormattedName().
26. Используя текстовое описание ("Экран профиля: Аватар (круглый), имя пользователя (крупный шрифт), email, кнопка 'Редактировать'") сгенерируйте прототип в Figma AI / Galileo AI. Экспортируйте дизайн. Используя плагин Figma-to-Compose (или вручную на основе макета), реализуйте экран в Compose. С помощью ИИ-ассистента в Android Studio доработайте UI (например, добавьте анимацию нажатия кнопки).
27. Создайте data class NewsItem (id, title, summary). С помощью ИИ сгенерируйте шаблон NewsViewModel: используйте StateFlow для хранения состояния списка новостей (List<NewsItem>). Реализуйте Composable-функцию NewsScreen, которая отображает: заголовок экрана, состояние загрузки (Loading), список новостей (при успехе) или сообщение об ошибке. В ViewModel имитируйте загрузку данных (задержка 2 сек, возврат тестового списка).
28. Используя ИИ (Copilot Chat, ChatGPT), составьте сравнительную таблицу паттернов MVI и MVVM. Укажите ключевые компоненты, поток данных, преимущества, недостатки, применимость в Compose. Приведите минимальные примеры состояний (State) и намерений (Intent/Event) для экрана входа.

29. Для экрана "Лента новостей" (фильтрация по категориям, поиск, пагинация) попросите ИИ сгенерировать максимально полный класс состояния (data class NewsScreenState), включая все необходимые поля (isLoading, newsList, errorMessage, selectedCategory, searchQuery, isEndOfList).
30. Создайте два минимальных проекта Compose: один использует StateFlow в ViewModel, другой - LiveData. Сравните код ViewModel и код наблюдения в Composable (collectAsState() vs observeAsState()). Попросите ИИ объяснить ключевые различия и преимущества StateFlow.
31. Опишите ИИ интерфейс экрана (показ данных, ошибка, загрузка). Попросите сгенерировать шаблонную Composable-функцию, которая принимает State из ViewModel и использует when для отрисовки соответствующего состояния (Loading, Success, Error).
32. Исследуйте основы Hilt. Попросите ИИ сгенерировать код: модуль (@Module), предоставляющий экземпляр вашего NewsRepository. Аннотировать NewsViewModel с @HiltViewModel. Показать пример @Inject в ViewModel.
33. Реализуйте экран с вводом текста в TextField. Обеспечьте сохранение введенного текста при повороте экрана, используя rememberSaveable и Saver или ViewModel (объясните выбор). Попросите ИИ помочь с реализацией Saver для сложного объекта.
34. Добавьте в ваш ViewModel логгирование ключевых событий (например, Log.d("VM", "Loading news for category: $category")). Используя ИИ, предложите стратегию для централизованного и конфигурируемого логгирования в приложении.
35. Напишите простой юнит-тест (JUnit) для метода loadData() вашего ViewModel. Проверьте, что состояние корректно меняется на Loading, а затем на Success с данными. Используйте TestCoroutineDispatcher (советы по настройке запросите у ИИ).
36. Дайте ИИ текстовое описание сущности "Транзакция" (id, сумма, тип [доход/расход], категория, дата, описание). Сгенерируйте класс Transaction с аннотациями Room (@Entity, @PrimaryKey).
37. Для сущности Transaction попросите ИИ сгенерировать интерфейс TransactionDao с методами: @Insert, @Delete, @Query("SELECT * FROM transaction ORDER BY date DESC") (возвращает Flow<List<Transaction>>), @Query("SELECT SUM(amount) FROM transaction WHERE type = 'INCOME'") (возвращает Flow<Double>).
38. Добавьте в сущность Transaction новое поле isSynced: Boolean (значение по умолчанию false). Используя ИИ, сгенерируйте код миграции (Migration) для Room, выполняющий ALTER TABLE и устанавливающий isSynced = false для существующих записей.
39. Добавьте в сущность поле tags: List<String> (список тегов). Попросите ИИ помочь реализовать класс Converters с методами @TypeConverter для преобразования List<String> в String (JSON) и обратно.
40. Реализуйте класс TransactionRepository, который инкапсулирует доступ к TransactionDao. Методы репозитория (getAllTransactions(), getTotalIncome(), insertTransaction()) должны быть suspend и вызывать DAO внутри coroutineScope. Используйте ИИ для шаблона.
41. Создайте Composable-функцию, которая использует transactionRepository.getAllTransactions().collectAsState(initial = emptyList()) для отображения списка транзакций в LazyColumn. Добавьте базовый UI.
42. Попросите ИИ сгенерировать запрос DAO для получения суммы доходов и расходов **по месяцам** (@Query("SELECT strftime('%Y-%m', date/1000, 'unixepoch') as month, type, SUM(amount) FROM transaction GROUP BY month, type")).

Обработайте результат Flow<List<SomeResultClass>> (сгенерируйте data class для результата запроса).

43. В TransactionRepository создайте метод getMonthlySummary(): Flow<Map<String, Pair<Double, Double>>>, который преобразует поток из задания 7 в удобную для отображения структуру (месяц -> (Доходы, Расходы)). Используйте операторы Flow (map). ИИ поможет с синтаксисом.
44. Описать схему БД для сущности Note (id: автоинкремент, title: String, content: String, createdAt: Long). С помощью ИИ сгенерировать: класс Note с аннотациями Room (@Entity), интерфейс NoteDao с методами @Insert, @Query("SELECT * FROM note ORDER BY createdAt DESC") (возвращающий Flow<List<Note>>), и абстрактный класс AppDatabase. Реализовать репозиторий NoteRepository, использующий DAO и Coroutines для асинхронных операций.
45. Приведите ИИ пример JSON-ответа от любого публичного API (например, https://api.chucknorris.io/jokes/random). Попросите сгенерировать соответствующие data class-ы (включая вложенные классы, если нужно) с аннотациями @SerializedName (Gson/Moshi).
46. На основе документации публичного API (например, PokeAPI, JSONPlaceholder) попросите ИИ сгенерировать интерфейс ApiService с 3-4 методами (@GET, @POST), включая параметры пути/запроса. Укажите базовый URL.
47. Используя ИИ-подсказки, напишите код модуля (Hilt/Koin или вручную в Application-классе), который создает экземпляр Retrofit с базовым URL, конвертером Gson/Moshi и добавляет логгирующий интерсептор (HttpLoggingInterceptor).
48. Реализуйте в Repository метод для сетевого запроса, использующий try/catch. В блоке catch обрабатывайте IOException (сеть) и HttpException (код ответа != 2xx). Преобразуйте ошибку в понятное сообщение для UI. Попросите ИИ помочь с кодом обработки.
49. Добавьте к экрану, отображающему данные из сети, функционал "Повторить" при ошибке и "Обновить" (SwipeRefresh). Сгенерируйте с помощью ИИ пример кода SwipeRefresh в Compose и логику повторного вызова метода репозитория.
50. Реализуйте простую стратегию кеширования в памяти (HashMap или LruCache). При запросе данных сначала проверяйте кеш. Если данных нет/устарели, делайте сетевой запрос и обновляйте кеш. ИИ поможет с шаблоном.
51. Исследуйте, как добавить токен аутентификации в заголовки Retrofit запроса (Interceptor). Попросите ИИ сгенерировать код Interceptor, который добавляет заголовок Authorization: Bearer <token> к каждому запросу. Где безопасно хранить токен? (ИИ подскажет варианты: EncryptedSharedPreferences, Keystore).
52. Выберите API, поддерживающее пагинацию (например, NewsAPI). Попросите ИИ помочь сгенерировать data class для ответа с пагинацией (обычно содержит items: List<T>, totalResults, page) и реализовать базовую логику подгрузки "страниц" в ViewModel (PagingSource можно оставить на потом).
53. Используя публичное API (например, JSONPlaceholder - /posts), с помощью ИИ: сгенерировать data-класс Post по структуре ответа; сгенерировать интерфейс ApiService с методом @GET("posts") suspend fun getPosts(): List<Post>. Реализовать PostsViewModel, который через репозиторий загружает посты с использованием Retrofit и Coroutines, и отображает их в Compose UI. Добавить обработку состояния загрузки и ошибок. Использовать ИИ для подсказок по обработке ошибок.
54. Используя ИИ, сгенерируйте шаблонный код в Composable-функции для запроса разрешений на доступ к камере (CAMERA) и геолокации (ACCESS_FINE_LOCATION) с использованием rememberLauncherForActivityResult и ActivityResultContracts.RequestMultiplePermissions(). Обработайте результаты.

55. Попросите ИИ сгенерировать код для: а) Проверки наличия камеры на устройстве. б) Создания Intent(MediaStore.ACTION_IMAGE_CAPTURE). в) Запуска интента с launcher. г) Обработки результата (получение миниатюры Bitmap из data: Intent?).
56. Модифицируйте код задания 2. Попросите ИИ помочь создать временный файл для сохранения полноразмерного фото (FileProvider!), передать его URI в интент камеры через MediaStore.EXTRA_OUTPUT и сохранить полученное фото в галерею (MediaScannerConnection).
57. Создайте класс UploadWorker(appContext: Context, workerParams: WorkerParameters) : Worker(appContext, workerParams). Попросите ИИ помочь сгенерировать код внутри doWork(), который имитирует загрузку файла (логирование + delay). Запланируйте выполнение этого воркера по нажатию кнопки.
58. Модифицируйте задание 4. Установите ограничения (Constraints) для воркера: выполнять только при подключении к сети Интернет и при достаточном заряде батареи. Используйте ИИ для подсказки по синтаксису.
59. Исследуйте возможности TensorFlow Lite на мобильных устройствах (распознавание изображений, объектов, текста). Используя ИИ (Copilot, ChatGPT), найдите и опишите шаги для интеграции *готовой* TFLite модели (например, MobileNet для классификации изображений) в Android-проект.
60. Реализуйте получение последнего известного местоположения (через FusedLocationProviderClient). Используя ИИ, сгенерируйте код для отображения этого местоположения на карте (Google Maps Compose library). Добавьте простой маркер.
61. Исследуйте особенности запроса фонового доступа к геолокации (ACCESS_BACKGROUND_LOCATION). Чем он отличается от ACCESS_FINE_LOCATION? Когда необходим? Как его запросить? Какие ограничения вводят новые версии Android? Составьте краткий отчет с помощью ИИ.
62. С помощью ИИ сгенерировать шаблон кода для: а) Запроса разрешения на доступ к геолокации во время выполнения в Compose. б) Получения последнего известного местоположения. Реализовать экран, который по нажатию кнопки запрашивает разрешение, получает координаты и отображает их (широта, долгота). Использовать ИИ для подсказок по обработке случаев отказа в разрешении.
63. Используя ИИ для генерации boilerplate, реализовать простую фоновую задачу с WorkManager (например, периодическую синхронизацию с мнимым сервером раз в день, логирующую факт выполнения).
64. Написать функцию Calculator.add(a: Int, b: Int): Int. С помощью ИИ-ассистента (Copilot/JetBrains AI) сгенерировать юнит-тесты (JUnit) для этой функции, покрывающие основные случаи (положительные числа, ноль, отрицательные).
65. Для экрана задания 1 ("HelloAI"): С помощью ИИ-инструмента (Testim) или вручную (при знании Espresso/Compose Testing) описать сценарий UI-теста: Ввод имени в поле, нажатие кнопки, проверка появления приветствия. *ИЛИ* Сгенерировать с помощью ИИ запрос для анализа логов Logcat на предмет поиска всех ERROR за последний запуск приложения.
66. Выберите простой класс утилит (например, StringUtils с методами isEmailValid(email: String): Boolean, capitalizeFirstLetter(str: String): String).

    Попросите ИИ (Copilot, JetBrains AI) сгенерировать набор юнит-тестов (JUnit) для этих методов, покрывающий граничные случаи.
67. Напишите юнит-тест для метода loadData() вашего ViewModel (из темы 4). Используйте TestCoroutineDispatcher для управления корутинами. Проверьте изменения состояния (StateFlow или LiveData). ИИ поможет с моками и

настройкой.

68. Опишите ИИ (Testim, Applitools) простой сценарий UI: "Открыть приложение, нажать на кнопку 'Login', ввести email и пароль в текстовые поля, нажать кнопку 'Submit', проверить появление экрана 'Добро пожаловать'". Экспортируйте сгенерированный сценарий (если возможно) или опишите шаги.
69. Создайте намеренную ошибку в приложении (например, NPE). Запустите приложение и получите стектрейс в Logcat. Скопируйте логи и попросите ИИ (Copilot Chat, ChatGPT) проанализировать их: где произошла ошибка (файл, строка), в чем ее причина, как исправить.
70. Запустите ваше приложение в Android Studio Profiler. Создайте сценарий, вызывающий утечку памяти (например, регистрация слушателя без отписки). Зафиксируйте утечку в Memory Profiler. Используйте ИИ, чтобы объяснить, как найти причину утечки по Heap Dump/Allocation Tracking.
71. Настройте JaCoCo или Kover для вашего модуля. Запустите юнит-тесты и сгенерируйте отчет о покрытии. Попросите ИИ проанализировать отчет: какие классы/методы не покрыты? Сгенерируйте тест-кейсы для одного из непокрытых методов.
72. Опишите ИИ основную функциональность одного экрана вашего приложения (например, экран входа). Попросите сгенерировать чек-лист (test checklist) для его ручного тестирования (позитивные/негативные сценарии, проверка UI, граничные значения).
73. Используйте инструменты статического анализа кода (Android Lint через IDE, Detekt, ktlint) на вашем проекте. Попросите ИИ помочь интерпретировать предупреждения/ошибки, особенно связанные с безопасностью (например, HardcodedCredentials), производительностью или стилем кода. Исправьте 5 найденных проблем.

Типовые задания для самостоятельной работы

1. Опишите структуру приложения, назначение и взаимодействие ее компонентов, методы каждого класса: приложение позволяет вводить полную информацию об абитуриенте (предусмотреть возможность подачи документов на несколько направлений)
2. Опишите структуру приложения, назначение и взаимодействие ее компонентов, методы каждого класса: приложение курьера получает информацию о заказах, имеется возможность сообщать серверу о прочтении и доставке заказа (без использования баз данных).
3. Опишите структуру приложения, назначение и взаимодействие ее компонентов, методы каждого класса: приложение «Записная книжка» хранит информацию о списке запланированных дел и списке выполненных дел, с указанием геолокации места выполнения, имеется поддержка аудиозаписей (с использованием базы данных).
4. Опишите структуру программной системы, назначение и взаимодействие ее компонентов, методы каждого класса: программная система сбора информации в виде опроса на мобильном устройстве. Определить WEB сервер, хранящий данные с использованием СУБД. Список опросов и их структура получается от сервера на устройство по протоколу HTTP в формате JSON.

4.2 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Методические рекомендации, определяющие процедуры оценивания тестов:

Тест проводится онлайн в системе Moodle или Google Docs и ограничен по времени. На сдачу теста дается две попытки. Тест считается успешно пройденным если студент правильно ответил на 70% вопросов.

Методические рекомендации, определяющие процедуры оценивания на контрольные вопросы:

Опрос проводится в письменной форме в системе Moodle или Google Docs и ограничен по времени.

Критерии оценки:

оценка «незачет»: непонимание сущности излагаемого вопроса, грубые ошибки в ответе.

оценка «зачтено»: понимает суть вопроса; перечислены основные элементы описываемой сущности; дано частичное описание элементов описываемой сущности

Методические рекомендации, определяющие процедуры оценивания выполнения контрольных заданий:

При оценивании задания учитывается выполнение следующих условий:

- уровень выполнения задания соответствует УГТ 8 - Система завершена и квалифицирована с помощью тестов и демонстраций;
- предоставлен исходный код на Java/Kotlin в среде Android Studio (0-20 баллов);
- продемонстрирована работоспособность приложения на мобильном устройстве или в эмуляторе (0-10 баллов);
- студент понимает исходный код и отвечает на вопросы по его организации (0-25 баллов);
- использование ИИ на разных этапах (дизайн, код, тесты, анализ) (0-35 баллов).

Задание считается выполненным, если набрано 70 баллов.

Методические рекомендации, определяющие процедуры оценивания самостоятельной работы:

Оценивание результатов самостоятельной работы основывается на качестве выполнения студентом индивидуального задания. Описание компонентов, классов и использование ИИ-ассистентов в Google Docs. Уровень выполнения задания соответствует УГТ 8 - Система завершена и квалифицирована с помощью тестов и демонстраций.

Критерии оценки:

оценка «незачет»: не представлена структура приложения и ее компонентов или не описаны свойства и методы основных классов;

оценка «зачтено»: представлена структура приложения и ее компонентов, описаны свойства и методы основных классов, программное приложение реализует полностью/часть необходимого функционала;

Методические рекомендации, определяющие процедуры оценивания на зачете:

Процедура промежуточной аттестации проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

Итоговой формой контроля сформированности компетенций у обучающихся по дисциплине является зачет. Студенты обязаны сдать зачет в соответствии с расписанием и учебным планом

Зачет по дисциплине преследует цель оценить работу студента за курс, получение теоретических знаний, их прочность, развитие творческого мышления, приобретение

навыков самостоятельной работы, умение применять полученные знания для решения практических задач.

Оценивание уровня освоения дисциплины основывается на качестве выполнения студентом заданий текущего контроля и ответов на теоретические вопросы.

Критерии оценки:
оценка «зачтено» в случае выполнения:

- успешно пройдены тесты по всем темам;
- зачтены ответы на контрольные вопросы;
- выполнены все контрольные задания;
- зачтено задание самостоятельной работы.

оценка «незачет» выставляется в случае невозможности поставить оценку «зачтено».

Методические рекомендации, определяющие процедуры оценивания курсовой работы:

Оценка выставляется на основе: выполнения индивидуального плана, индивидуального задания и пояснительной записке. Оценку выставляет комиссия. назначенная кафедрой.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на зачете;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических

средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4.3. Методические указания по организации вычислительной инфраструктуры

Условия применения:

- Курс рассчитан на студентов 7-го семестра (бакалавриат).
- Наличие доступа к вычислительным ресурсам (GitLab, Android Studio, эмуляторы).
- Индивидуальные задания адаптированы под инфраструктуру CI/CD (GitLab, автотесты).
- Автотесты интегрированы с требованиями к спецификации методов, коммитам и использованию ИИ-инструментов.

Цели, задачи и ожидаемые результаты

Цели организации инфраструктуры:

- Формирование навыков работы в IT-инфраструктуре (Git, CI/CD, автотестирование).
- Применение ИИ-инструментов (Copilot, JetBrains AI) в промышленном цикле разработки.
- Обеспечение контроля качества кода через автоматизированное тестирование.

Задачи преподавателя:

1. Создание учетных записей студентов в GitLab.
2. Настройка GitLab Runner для Android-среды (эмуляторы, SDK).
3. Разработка шаблонного репозитория лабораторных работ с поддержкой Kotlin/Android.
4. Реализация автотестов для индивидуальных заданий (Unit, UI-тесты, ИИ-анализ).
5. Визуализация результатов тестирования (HTML-отчеты, интеграция с GitLab).
6. Написание инструкций по работе с ИИ-инструментами и CI/CD.
7. Адаптация заданий по итогам тестирования инфраструктуры.

Ожидаемые результаты студентов:

- Умение работать с Git: именование коммитов, веток, пул-реквесты.
- Навыки запуска и анализа автотестов в CI/CD-конвейере.
- Опыт интеграции ИИ-инструментов (Copilot, JetBrains AI) в разработку.
- Понимание стандартов качества кода (Kotlin/Android, тестовое покрытие).

Порядок реализации

Задача 1: Настройка GitLab

- Создание групповых проектов в GitLab для дисциплины.
- Регистрация студентов с назначением ролей (Developer).

Задача 2: GitLab Runner для Android

- Развертывание Runner на Ubuntu 24.04 с Docker.

- Установка зависимостей:

bash
Copy
Download
```
# Android SDK, эмуляторы, Kotlin-среда
sudo apt-get install android-sdk qemu-kvm kotlin
```

- Регистрация Runner для приватного GitLab-сервера.
- Настройка cron для очистки артефактов.

Задача 3: Шаблон репозитория

Структура шаблона:

text
Copy
Download
```
/android-project
├── .gitlab-ci.yml    # CI-конфигурация
├── Dockerfile        # Образ с Android SDK
├── get_path.py       # Скрипт выбора тестов по коммиту
├── README.md         # Инструкции по ИИ-инструментам
└── /src              # Исходный код на Kotlin
```

Ключевые файлы:

- **.gitlab-ci.yml**:

yaml
Copy
Download
```
stages:
  - test
  - push_report
android-test:
  image: android-emulator:latest
  script:
    - ./gradlew test # Запуск Unit-тестов
    - python get_path.py # Анализ коммитов
```

- **Dockerfile**:

dockerfile
Copy
Download
```
FROM ubuntu:24.04
RUN apt-get update && apt-get install -y android-sdk qemu-kvm kotlin
```

Задача 4: Автотесты

- **Unit-тесты**: JUnit + MockK для Kotlin.
- **UI-тесты**: Espresso/Compose Testing с поддержкой ИИ (Applitools).
- **ИИ-анализ**: Интеграция с SonarQube для оценки качества кода.

Пример теста для лабораторной работы №1:

kotlin
Copy
Download
```
@Test
fun testHelloWorldGeneratedByAI() {
    val app = launchComposeRule()
    app.onNodeWithText("Hello, AI!").assertExists()
}
```

Задача 5: Визуализация результатов

- Генерация HTML-отчетов с детализацией:
  - Успешные/неудачные тесты.
  - Покрытие кода (JaCoCo).
  - Анализ использования ИИ-инструментов.
- Пример отчета:
  https://media/report_android.png

Порядок проверки корректности
Чек-лист для студентов:

1. **Git-инфраструктура**:
   - Учетные записи в GitLab.
   - Формат коммитов: [LabX] Добавлен метод ... с помощью Copilot.
2. **Шаблон репозитория**:
   - Клонирование шаблона.
   - Интеграция с CI/CD через .gitlab-ci.yml.
3. **Автотесты**:
   - Запуск тестов через ./gradlew test.
   - Исправление кода при неудачных тестах.
4. **ИИ-инструменты**:
   - Документирование использования Copilot/JetBrains AI в README.md.
   - Соответствие сгенерированного кода стандартам Kotlin.

4.4. Методические указания по организации лабораторных работ

Условия применения:

- Курс рассчитан на студентов 7-го семестра (бакалавриат).
- Наличие доступа к:
  - GitLab
  - Android Studio + эмуляторы
  - ИИ-инструментам (Copilot, JetBrains AI)
- Инфраструктура CI/CD (GitLab, автотесты) интегрирована с лабораторными работами.

Цели, задачи и ожидаемые результаты
Цели лабораторных работ:

1. Сформировать навыки разработки нативных Android-приложений на Kotlin.
2. Научить применять ИИ-инструменты для:
   - Генерации кода
   - Тестирования
   - Оптимизации архитектуры
3. Привить культуру работы с Git и CI/CD.

Задачи преподавателя:

1. Подготовить план лабораторных работ (8 ИЗ).
2. Разработать шаблонные репозитории GitLab с CI/CD.
3. Настроить автотесты для каждого ИЗ (Unit, UI, ИИ-анализ).
4. Адаптировать задания после тестирования инфраструктуры.
5. Создать инструкции по работе с ИИ-инструментами.

Ожидаемые результаты студентов:

- Умение разрабатывать Android-приложения в среде Android Studio.
- Навыки использования ИИ для генерации кода, тестов и рефакторинга.
- Опыт работы с Git: именование коммитов, пул-реквесты, CI/CD.

Порядок реализации
Задача 1: План лабораторных работ
(Соответствует разделу 2.3.3 РПД)

Индивидуальное задание	Содержание	Распределение часов
ЛР1: Введение в Android-разработку	- Создание проекта (Empty Activity, Compose). - Активация Copilot/JetBrains AI. - Генерация "HelloWorld" с ИИ. - Работа с эмулятором.	10 ч (4 ч ЛР + 6 ч СР)
ЛР2: Kotlin для Android	- Приложение с 2+ активностями. - Передача данных между экранами. - Генерация data class/sealed class с ИИ.	10 ч (4 ч ЛР + 6 ч СР)
ЛР3: Основы UI/UX	- Генерация макета в Figma AI/Galileo AI. - Конвертация в Compose-код. - Доработка UI с ИИ-подсказками.	14 ч (6 ч ЛР + 8 ч СР)
ЛР4: Архитектура приложения	- MVVM с StateFlow/LiveData. - Генерация ViewModel с ИИ. - Навигация (Navigation Component).	12 ч (4 ч ЛР + 8 ч СР)
ЛР5: Работа с данными	- Room: генерация Entity/DAO с ИИ. - Асинхронные операции (Coroutines/Flow).	16 ч (6 ч ЛР + 10 ч СР)
ЛР6: Сетевое взаимодействие	- Retrofit: генерация data-классов по JSON. - Обработка ошибок с ИИ.	14 ч (4 ч ЛР + 10 ч СР)
ЛР7: Фоновые задачи	- Доступ к камере/геолокации (ИИ-генерация boilerplate). - WorkManager.	14 ч (4 ч ЛР + 10 ч СР)
ЛР8: Тестирование + ИИ-инструменты	- Генерация юнит-тестов (Copilot). - ИИ-анализ логов (Logcat). - Оценка покрытия кода.	15.8 ч (6 ч ЛР + 9.8 ч СР)

Задача 2: Пример ИЗ1 (ЛР1)
Цель: Освоить создание проекта Android Studio, генерацию кода с ИИ, работу с эмулятором.
Технологии: Kotlin, Jetpack Compose, Copilot/JetBrains AI.
Задание:

1. Создайте проект "HelloAI" (Empty Activity, Compose).
2. Активируйте Copilot/JetBrains AI в Android Studio.
3. С помощью ИИ сгенерируйте:
   kotlin
   Copy

Download

```
// Промпт: "Создай Composable-функцию с TextField для имени и кнопкой, которая выводит 'Привет, {имя}!'"
```

4. Запустите приложение на эмуляторе (Pixel 5, API 34).
5. Задокументируйте использованные промпты в README.md.

Автотесты:

kotlin
Copy
Download

```
@Test
fun testHelloWorldGeneratedByAI() {
    composeTestRule.setContent { HelloAIScreen() }
    composeTestRule.onNodeWithText("Привет, ИИ!").assertExists()
}
```

Контрольные вопросы:

1. Какой шаблон Activity выбрать для Compose?
2. Объясните разницу между `@Composable` и обычной функцией.
3. Как ИИ упрощает создание boilerplate-кода?

Критерии оценки:

- **Отлично**: Весь код сгенерирован ИИ, тесты пройдены, промпты задокументированы.
- **Неудовлетворительно**: Приложение не запускается, нет истории коммитов.

Задача 3: Интеграция автотестов

Для каждого ИЗ реализованы:

1. **Unit-тесты**: JUnit + MockK (проверка ViewModel, репозиториев).
2. **UI-тесты**: Compose Testing / Espresso (симуляция взаимодействия).
3. **ИИ-анализ**:
    - Проверка стиля кода (SonarQube).
    - Оценка тестового покрытия (JaCoCo).

Пример для ЛР5 (Room):

kotlin
Copy
Download

```
@Test
fun testRoomEntityGeneratedByAI() {
    val user = User(name = "AI_User", id = 1)
    assertEquals(1, user.id) // Проверка корректности data-класса
}
```

Порядок проверки корректности

Чек-лист:

1. **Git-инфраструктура**:
    - Учетные записи студентов в GitLab.
    - Шаблонные репозитории с CI/CD (.gitlab-ci.yml).
2. **Автотесты**:
    - Для всех 8 ИЗ (Unit, UI, ИИ-анализ).
3. **Документация**:
    - Инструкция по работе с ИИ-инструментами.
    - Правила именования коммитов:

      text
      Copy
      Download

      ```
      [LabX] Добавлен метод ... с помощью Copilot
      ```

4. **Адаптированные задания**:

- После тестирования инфраструктуры (пример: уточнение спецификаций методов для Room в ЛР5).

5. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

5.1 Основная литература:

1. Брайан Харди, Билл Филлипс. Android. Программирование для профессионалов. – СПб.: Питер, 2021. – 848 с. ISBN 978-5-4461-1329-7.
2. Android Developers. [Электронный ресурс]. – URL: https://developer.android.com/?hl=ru (дата обращения: 18.07.2025).
3. Голощапов А.Л. Android. Программирование для профессионалов. – 3-е изд. – СПб.: Питер, 2020. – 688 с. ISBN 978-5-4461-1382-2.
4. Kotlin Lang. Документация Kotlin [Электронный ресурс]. – URL: https://kotlinlang.org/docs/home.html (дата обращения: 18.07.2025).
5. Retrofit Documentation [Электронный ресурс]. – URL: https://square.github.io/retrofit/ (дата обращения: 18.07.2025).
6. OkHttp Documentation [Электронный ресурс]. – URL: https://square.github.io/okhttp/ (дата обращения: 18.07.2025).
7. Gson User Guide [Электронный ресурс]. – URL: https://github.com/google/gson/blob/master/UserGuide.md (дата обращения: 18.07.2025).
8. Moshi Documentation [Электронный ресурс]. – URL: https://github.com/square/moshi (дата обращения: 18.07.2025).
9. Testim AI-Powered Testing [Электронный ресурс]. – URL: https://www.testim.io/ (дата обращения: 18.07.2025).
10. Applitools Visual AI [Электронный ресурс]. – URL: https://applitools.com/ (дата обращения: 18.07.2025).
11. JetBrains AI Assistant Documentation [Электронный ресурс]. – URL: https://www.jetbrains.com/help/idea/ai-assistant.html (дата обращения: 18.07.2025).
12. GitHub Copilot Documentation [Электронный ресурс]. – URL: https://docs.github.com/en/copilot (дата обращения: 18.07.2025).

Для освоения дисциплины инвалидами и лицами с ограниченными возможностями здоровья имеются издания в электронном виде в электронно-библиотечных системах «Лань» и «Юрайт».

5.2 Дополнительная литература:

1. Бурков А. Android за неделю. Создание приложений под смартфоны и планшеты. – СПб.: БХВ-Петербург, 2022. – 352 с. ISBN 978-5-9775-3895-9.
2. Жемеров Д., Исакова С. Kotlin в действии. – СПб.: Питер, 2021. – 496 с. ISBN 978-5-4461-1583-3.
3. Гриффитс Д., Гриффитс Д. Head First. Изучаем Kotlin. – СПб.: Питер, 2022. – 640 с. ISBN 978-5-4461-2206-0.
4. Сьярма Д., Грин Х. Kotlin. Программирование профессионального уровня для Android. – М.: ДМК Пресс, 2023. – 476 с. ISBN 978-5-93700-150-0.
5. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2021. – 352 с. ISBN 978-5-4461-1535-2.
6. Васильев А.Н. Паттерны проектирования для Android. – СПб.: БХВ-Петербург, 2023. – 288 с. ISBN 978-5-9775-4098-3.
7. Фримен Э., Фримен Э., Сьерра К., Бейтс Б. Паттерны проектирования. – СПб.: Питер, 2022. – 656 с. ISBN 978-5-4461-1702-8.
8. JUnit 5 User Guide [Электронный ресурс]. – URL: https://junit.org/junit5/docs/current/user-

guide/ (дата обращения: 18.07.2025).
9. Espresso Cheat Sheet [Электронный ресурс]. – URL: https://developer.android.com/training/testing/espresso/cheat-sheet (дата обращения: 18.07.2025).

5.3. Периодические издания:

1. Базы данных компании «Ист Вью» http://dlib.eastview.com
2. Электронная библиотека GREBENNIKON.RU https://grebennikon.ru/

5.4. Интернет-ресурсы, в том числе современные профессиональные базы данных и информационные справочные системы

Электронно-библиотечные системы (ЭБС):

1. ЭБС «ЮРАЙТ» https://urait.ru/
2. ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» http://www.biblioclub.ru/
3. ЭБС «BOOK.ru» https://www.book.ru
4. ЭБС «ZNANIUM.COM» www.znanium.com
5. ЭБС «ЛАНЬ» https://e.lanbook.com

Профессиональные базы данных

1. Scopus http://www.scopus.com/
2. ScienceDirect https://www.sciencedirect.com/
3. Журналы издательства Wiley https://onlinelibrary.wiley.com/
4. Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
5. Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru
6. Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
7. Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/
8. База данных CSD Кембриджского центра кристаллографических данных (CCDC) https://www.ccdc.cam.ac.uk/structures/
9. Springer Journals: https://link.springer.com/
10. Springer Journals Archive: https://link.springer.com/
11. Nature Journals: https://www.nature.com/
12. Springer Nature Protocols and Methods: https://experiments.springernature.com/sources/springer-protocols
13. Springer Materials: http://materials.springer.com/
14. Nano Database: https://nano.nature.com/
15. Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
16. "Лекториум ТВ" http://www.lektorium.tv/
17. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Информационные справочные системы

1. Консультант Плюс - справочная правовая система (доступ по локальной сети с компьютеров библиотеки)

Ресурсы свободного доступа

1. КиберЛенинка http://cyberleninka.ru/;
2. Американская патентная база данных http://www.uspto.gov/patft/
3. Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
4. Федеральный портал "Российское образование" http://www.edu.ru/;

5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
6. Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
7. Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
8. Справочно-информационный портал "Русский язык" http://gramota.ru/;
9. Служба тематических толковых словарей http://www.glossary.ru/;
10. Словари и энциклопедии http://dic.academic.ru/;
11. Образовательный портал "Учеба" http://www.ucheba.com/;
12. Законопроект "Об образовании в Российской Федерации". Вопросы и ответы http://xn--273--84d1f.xn--p1ai/voprosy_i_otvety

Собственные электронные образовательные и информационные ресурсы КубГУ

1. Электронный каталог Научной библиотеки КубГУ http://megapro.kubsu.ru/MegaPro/Web
2. Электронная библиотека трудов ученых КубГУ http://megapro.kubsu.ru/MegaPro/UserEntry?Action=ToDb&idb=6
3. Среда модульного динамического обучения http://moodle.kubsu.ru
4. База учебных планов, учебно-методических комплексов, публикаций и конференций http://infoneeds.kubsu.ru/
5. Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
6. Электронный архив документов КубГУ http://docspace.kubsu.ru/
7. Электронные образовательные ресурсы кафедры информационных систем и технологий в образовании КубГУ и научно-методического журнала "ШКОЛЬНЫЕ ГОДЫ" http://icdau.kubsu.ru/

6. Методические указания для обучающихся по освоению дисциплины (модуля)

По курсу предусмотрено проведение лекционных занятий, на которых дается основной систематизированный материал. В ходе лекционных занятий разбираются свойства, методы и события основных элементов мобильного программирования, приводятся примеры их использования, проводится анализ наиболее распространенных ошибок реализации. После прослушивания лекции рекомендуется выполнить упражнения, приводимые в аудитории для самостоятельной работы.

По курсу предусмотрено проведение лабораторных занятий, на которых дается прикладной систематизированный материал. В ходе занятий разбираются готовые программные приложения использующие свойства, методы и события основных объектов библиотек Android Studio, а также приводятся примеры разработки программных приложений. После занятия рекомендуется выполнить упражнения, приводимые в аудитории для самостоятельной работы.

При самостоятельной работе студентов необходимо изучить литературу, приведенную в перечнях выше, для осмысления вводимых понятий, анализа предложенных подходов и методов разработки программ. Разрабатывая решение новой задачи студент должен уметь выбрать эффективные и надежные структуры данных для представления информации, подобрать соответствующие алгоритмы для их обработки, учесть специфику языка программирования, на котором будет выполнена реализация. Студент должен уметь выполнять тестирование и отладку алгоритмов решения задач с целью обнаружения и устранения в них ошибок.

Важнейшим компонентом курса является самостоятельная проектная работа, в ходе которой студент разрабатывает законченное решение с уровнем технологической готовности (УТГ) 5-9 с применением CI/CD/CT для решения задач (кейсов) индустриальных партнеров. Допускается

выполнение проектов в командах.

Кейсы ПАО «Сбербанк»

1. Генерация пользовательских сценариев работы в мобильном приложении
Описание:
Банк хочет использовать генеративный ИИ для быстрой симуляции пользовательских сценариев — например, как клиент оформляет вклад, переводит средства, получает уведомление о риске мошенничества.
Цель:
Разработать генератор пошаговых сценариев пользовательского поведения с вариативностью (молодой клиент, пенсионер, ИП).
Ожидаемый результат:
Набор автоматически сгенерированных UX-сценариев, оформленных в виде сценариев для QA или UX-исследований, с логикой действий и типичными ошибками пользователя.
2. Анализ поведения пользователей в экосистеме цифрового рубля
Описание:
Сбербанк участвует в пилотных проектах по внедрению цифрового рубля. Интерес представляет исследование пользовательских паттернов: как изменяются модели потребления, скорости операций, уровень доверия, сравнение с классическим безналом.
Цель:
Построить модель анализа поведения клиентов, участвующих в транзакциях с цифровым рублем: частота, средний чек, контексты.
Ожидаемый результат:
Отчёт и ML-модель, классифицирующая типы пользователей и выявляющая ключевые различия в предпочтениях и барьерах цифровой валюты.

Кейсы от «АВАЛАБ»

1. Обратная генерация — ИИ-помощник для покупателей квартир
Описание:
Будущие покупатели часто задают типовые вопросы о квартирах, планировках, ипотеке, акциях, сроках. Вместо call-центра предлагается реализовать LLM-бота, который обрабатывает текстовые и голосовые запросы, показывает планировки, ссылается на PDF-документы и может «объяснять» информацию простым языком.
Цель:
Упростить коммуникацию с клиентами на этапе выбора квартиры и повысить качество первичного контакта.
Ожидаемый результат:
Демо-бот, способный отвечать на вопросы о жилом комплексе, ориентируясь в его характеристиках и маркетинговых документах.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Учебно-методическим обеспечением курсовой работы студентов являются:

1. учебная литература;
2. нормативные документы ВУЗа;
3. методические разработки для студентов.

Самостоятельная работа студентов включает:

– оформление итогового отчета (пояснительной запики).
– анализ нормативно-методической базы организации;
– анализ научных публикации по заранее определённой теме;
– анализ и обработку информации;
– работу с научной, учебной и методической литературой,
– работа с конспектами лекций, ЭБС.

Для самостоятельной работы представляется аудитория с компьютером и доступом в Интернет, к электронной библиотеке вуза и к информационно-справочным системам. Перечень учебно-методического обеспечения:

1. Основная образовательная программа высшего профессионального образования федерального государственного бюджетного образовательного учреждения высшего образования «Кубанский государственный университет» по направлению подготовки.
2. Положение о проведении текущего контроля успеваемости и промежуточной аттестации в федеральном государственном бюджетном образовательном учреждении высшего образования «Кубанский государственный университет».
3. Общие требования к построению, содержанию, оформлению и утверждению рабочей программы дисциплины Федерального государственного образовательного стандарта высшего профессионального образования.
4. Методические рекомендации по содержанию, оформлению и применению образовательных технологий и оценочных средств в учебном процессе, основанном на Федеральном государственном образовательном стандарте.
5. Учебный план основной образовательной программы по направлению подготовки.
6. Федеральный государственный образовательный стандарт высшего профессионального образования по направлению подготовки.

В освоении дисциплины инвалидами и лицами с ограниченными возможностями здоровья большое значение имеет индивидуальная учебная работа (консультации) – дополнительное разъяснение учебного материала.

Индивидуальные консультации по предмету являются важным фактором, способствующим индивидуализации обучения и установлению воспитательного контакта между преподавателем и обучающимся инвалидом или лицом с ограниченными возможностями здоровья.

7. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю)

7.1 Перечень информационно-коммуникационных технологий

– Проверка домашних заданий и консультирование посредством электронной почты.
– Использование электронных презентаций при проведении лекционных занятий
- Система MOODLE
- Проверка домашних заданий и консультирование посредством ЭОИС КубГУ

7.2 Перечень лицензионного и свободно распространяемого программного обеспечения и инструментов

Android Studio (с последним SDK).
Git (GitHub/GitLab/Bitbucket).
GitHub Copilot (https://github.com/features/copilot)
JetBrains AI Assistant (https://www.jetbrains.com/ai/)
Amazon CodeWhisperer ([https://aws.amazon.com/codewhisperer/] (https://aws.amazon.com/codewhisperer/))
Figma + AI плагины / Uizard (https://uizard.io/) / Galileo AI
TensorFlow Lite (https://www.tensorflow.org/lite) + Model Maker
Testim / Applitools (https://www.testim.io/, https://applitools.com/) (для ИИ-тестирования)
SonarQube / DeepCode (Snyk)
cloud.ru,YandexCloud,AWS/GCP/Azure– облачные вычисления

8. Материально-техническое обеспечение по дисциплине

№	Вид работ	Наименование учебной аудитории, ее оснащенность оборудованием и техническими средствами обучения
1.	Лекционные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
2.	Лабораторные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, проектором, программным обеспечением
3.	Групповые (индивидуальные) консультации	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
4.	Текущий контроль, промежуточная аттестация	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
5.	Самостоятельная работа	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.