АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ
«КУБАНСКИЙ ИНСТИТУТ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ»
(АНПОО «КУБАНСКИЙ ИПО»)

ОТДЕЛЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

РАБОЧАЯ ПРОГРАММА

учебной дисциплины
ОП.10 РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ
по специальности
09.02.11 РАЗРАБОТКА И УПРАВЛЕНИЕ ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ
направленность программы: Веб-разработка

Краснодар, 2026

СОГЛАСОВАНО
Зам. директора по КОД и МР

_____ / Т.В. Першакова
28.05.2026 г.

УТВЕРЖДАЮ
Директор АНПОО «Кубанский ИПО»

_____ О.Л. Шутов
Приказ №38-О от 28.05.2026 г.

ОДОБРЕНО
Педагогическим советом
Протокол №6 от 28.05.2026 г

РАССМОТРЕНО
на заседании УМО
«Информационные системы и программирование»
Протокол № 5 от 15.05.2026г.
Председатель _____ / С.А. Пясецкий

Рабочая программа учебной дисциплины ОП.10 Разработка мобильных приложений предназначена для реализации образовательной программы подготовки специалистов среднего звена.

Разработана на основе Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.11 Разработка и управление программным обеспечением (Приказ Министерства Просвещения Российской Федерации от 24 февраля 2025 г. № 138, зарегистрированного Министерством Юстиции России 31 марта 2025 г. № 81696) с учетом примерной образовательной программы, разработанной Федеральным учебно-методическим объединением в системе среднего профессионального образования по укрупненным группам профессий, специальностей 09.00.00 Информатика и вычислительная техника, с учетом профессиональных стандартов: «Программист» (Приказ Министерства труда и социальной защиты РФ от 20 июля 2022 г. № 424н, зарегистрирован Министерством юстиции Российской Федерации от 22 августа 2022г. №69720); «Разработчик Web и мультимедийных приложений» (Приказ Министерства труда и социальной защиты РФ от 18 января 2017 г. № 44н, зарегистрирован Министерством юстиции Российской Федерации от 31 января 2017 г. № 45481) и компетенции «Веб- технологии».

Организация-разработчик: АНПОО «Кубанский ИПО»

Разработчик:
Пясецкий С.А., преподаватель АНПОО «Кубанский ИПО»

Рецензенты:
1. Варкентин В.Ф. – преподаватель, АНПОО «Кубанский ИПО»
Квалификация по диплому: преподаватель информатики
2. Маслиев Р.О, генеральный директор ООО «Старт Эксперт»

СОДЕРЖАНИЕ

1 ОБЩАЯ ХАРАКТЕРИСТИКА РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ ОП.10 РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

1.1 Место дисциплины в структуре основной образовательной программы:

Учебная дисциплина ОП.10 Разработка мобильных приложений является вариативной частью общепрофессионального цикла образовательной программы в соответствии с ФГОС по специальности 09.02.11 Разработка и управление программным обеспечением.

Особое значение дисциплина имеет при формировании и развитии общих и профессиональных компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности;

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде;

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках;

ПК 2.1. Проектировать модули программного обеспечения;

ПК 2.2. Разрабатывать модули программного обеспечения;

ПК 2.3. Выполнять интеграцию модулей и компонентов программного обеспечения.

1.2 Цель и планируемые результаты освоения дисциплины:

В рамках программы учебной дисциплины обучающимися осваиваются умения и знания.

Код ПК, ОК	Умения	Знания
ПК.2.1., ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09	− *разрабатывать пользовательский интерфейс;* − *программировать на современном мобильном языке;* − *проектировать архитектуру приложения;* − *работать с локальными базами данных;* − *интегрировать REST API и Firebase;* − *тестировать приложение;* − *отлаживать и профилировать приложение;* − *оформлять техническую документацию;* − *оценивать трудозатраты и планировать работу;* − *работать в команде и презентовать результат.*	− *жизненный цикл мобильного приложения и особенности ОС;* − *методы алгоритмизации, формализации и нотации;* − *проектирование UI/UX и прототипирование;* − *основы вёрстки, типографики и графики;* − *современные языки программирования, используемые в мобильной разработке и их синтаксис;* − *архитектурные паттерны и управление состоянием;* − *базы данных и сетевое взаимодействие;* − *методы тестирования и тест-дизайна;* − *стандарты оформления кода и документации;* − *метрики качества и оценка трудозатрат;*

2 СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

2.1 Объем учебной дисциплины и виды учебной работы

Вид учебной работы	**Объем в часах**
ОБЪЕМ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ ДИСЦИПЛИНЫ	***99***
в том числе вариативная часть	*99*
- теоретическое обучение	***30***
- практические занятия	***60***
в т.ч. в форме практической подготовки	*60*
- промежуточная аттестация	***9***
в том числе:	
консультации	*6*
экзамен	*3*

Тематический план учебной дисциплины

Наименование разделов и тем	Количество аудиторных часов				
	Всего	в т.ч. в форме практической подготовки	самост. работа	теоретич. обучение	практич. занятия
Раздел 1. Введение. Основы мобильной разработки и организация труда	***4***	***2***	***-***	***2***	***2***
Тема 1.1 Мобильные платформы и их архитектура. Организация труда и управления разработкой	*4*	***2***	*-*	*2*	*2*
Раздел 2. Проектирование мобильных приложений	***10***	***6***	***-***	***4***	***6***
Тема 2.1 Алгоритмизация и формализация задач. Архитектура мобильного приложения	*4*	*2*	*-*	*2*	*2*
Тема 2.2 Прототипирование и дизайн интерфейсов. Верстка и графический дизайн	*6*	*4*	*-*	*2*	*4*
Раздел 3. Программирование мобильных приложений	***22***	***16***	***-***	***6***	***16***
Тема 3.1 Языки программирования для мобильных платформ. Декларативный UI	*4*	*2*	*-*	*2*	*2*
Тема 3.2 Навигация и жизненный цикл. Обработка событий и мультимедиа	*12*	*10*	*-*	*2*	*10*
Тема 3.3 Оптимизация интерфейса и графики. Стандарты оформления кода и рефакторинг	*6*	*4*	*-*	*2*	*4*
Итого за семестр	*36*	*24*	*-*	*12*	*24*
Раздел 4. Работа с данными, API, базами данных	***18***	***12***	***-***	***6***	***12***
Тема 4.1 Сетевые запросы и API	*6*	*4*	*-*	*2*	*4*
Тема 4.2 Локальное и облачное хранение данных	*10*	*8*	*-*	*2*	*8*
Тема 4.3 Паттерны работы с данными. Внедрение зависимостей и модульность	*2*	*-*	*-*	*2*	*-*
Раздел 5. Тестирование и отладка приложений	***12***	***8***	***-***	***4***	***8***
Тема 5.1 Виды, уровни и инструменты тестирования	*8*	*6*	*-*	*2*	*6*
Тема 5.2 Техники тест-дизайна. Отладка и профилирование	*4*	*2*	*-*	*2*	*2*
Раздел 6. Документирование и подготовка к внедрению	***10***	***6***	***-***	***4***	***6***
Тема 6.1 Стандарты оформления документации. Разработка пользовательской и эксплуатационной документации	*6*	*4*	*-*	*2*	*4*
Тема 6.2 Подготовка к публикации в магазины приложений. Методы оценки качества ИТ-продукта	*4*	*2*	*-*	*2*	*2*
Раздел 7. Командная работа и презентация проекта	***14***	***10***	***-***	***4***	***10***
Тема 7.1 Командная работа и взаимодействие с заказчиком. Анализ функциональных разрывов	*6*	*4*	*-*	*2*	*4*
Тема 7.2 Подготовка к демонстрации и защите проекта. Оценка и метрология в мобильной разработке	*8*	*6*	*-*	*2*	*6*
Консультации	***6***	*-*	*-*	*-*	*-*
Экзамен	***3***	*-*	*-*	*-*	*-*
Итого за семестр	*63*	*36*	*-*	*18*	*36*
ВСЕГО	***99***	***60***	***-***	***30***	***60***

2.2. Тематический план и содержание учебной дисциплины

Наименование разделов и тем	Содержание учебного материала и формы организации деятельности обучающихся	Объем в часах	Коды компетенций, формированию которых способствует элемент программы
1	***2***	***3***	***4***
	СЕМЕСТР 5 (3)		
Раздел 1. Введение. Основы мобильной разработки и организация труда		**4**	
Тема 1.1 *Мобильные платформы и их архитектура. Организация труда и управления разработкой*	***Содержание учебного материала***	**4**	ПК.2.1., ОК.01, ОК.02, ОК.04, ОК.09
	1 ***Мобильные платформы и их архитектура. Организация труда и управления разработкой*** *Платформы iOS, Android, кроссплатформа (Flutter, React Native). Особенности ОС. Жизненный цикл мобильного приложения. Инструменты разработчика (Android Studio, Xcode, VS Code). Организация работы: Git, системы отслеживания задач (Jira/Trello), техники формализации задач. Оценка трудозатрат, календарное планирование. Понятие человеко-ориентированного подхода.*	2	
	в том числе, практических занятий	**2**	
	ПЗ №1. *Установка и настройка среды разработки. Создание первого приложения*	2	
Раздел 2. Проектирование мобильных приложений		**10**	
Тема 2.1 *Алгоритмизация и формализация задач. Архитектура мобильного приложения*	***Содержание учебного материала***	**4**	ПК.2.1., ОК.01, ОК.02, ОК.04, ОК.09
	2 ***Алгоритмизация и формализация задач*** *Методы алгоритмизации: структурная, модульная, нисходящая/восходящая разработка. Приёмы формализации: математическая постановка, конечные автоматы, таблицы решений, псевдокод. Нотации для графического отображения алгоритмов: IDEF0, BPMN, UML (диаграммы деятельности, последовательности, состояний). Программное обеспечение для схем: draw.io, PlantUML, Miro. Паттерны проектирования: MVC, MVP, MVVM (сравнение, применение). Clean Architecture (Uncle Bob): слои данных, домена, представления. Dependency Inversion. Реактивное программирование: Flow, RxJava, Combine. Состояние приложения (StateFlow, SharedFlow). Мультиарендность (multitenancy): изоляция данных разных пользователей, архитектурные подходы. Внедрение зависимостей (DI): ручное, Dagger Hilt, Koin, Swinject*	2	
	в том числе, практических занятий	**2**	

	ПЗ №2. *Создание графических схем и блок-схем алгоритмов*		2	
Тема 2.2 *Прототипирование и дизайн интерфейсов. Верстка и графический дизайн*	***Содержание учебного материала***		**6**	ПК.2.1., ОК.01, ОК.02, ОК.04, ОК.09
	3	***Прототипирование и дизайн интерфейсов. Верстка и графический дизайн*** *Инструменты прототипирования: Figma (автолейауты, компоненты, variants), Sketch, Adobe XD. Техники эскизирования. Принципы анимации в интерфейсах: материальная анимация (Material Design), нативная iOS (HIG), переходы, жесты. Правила написания интерфейсных текстов: микро-копирайтинг, локализация, форматирование строк (String resources, plurals). Основы верстки: иерархическая модель (View tree), ConstraintLayout, Compose Layout, SwiftUI Stacks. Языки разметки: XML (Android), SwiftUI DSL, Compose Kotlin DSL, HTML/CSS для WebView. Основы типографики: кегль, интерлиньяж, гарнитуры (Roboto, San Francisco), правила набора. Теория цвета и композиции: цветовые модели (RGB, HEX), контрастность, иерархия элементов, золотое сечение. Работа с графикой: растровая (PNG, JPEG, WebP) vs векторная (SVG, VectorDrawable, PDF). Правила включения рисунков в документацию*	2	
	в том числе, практических занятий		**4**	
	ПЗ №3. *Разработка интерактивного прототипа приложения в Figma*		2	
	ПЗ №4. *Разработка интерактивного прототипа приложения в Figma*		2	
Раздел 3. Программирование мобильных приложений			**22**	
Тема 3.1 *Языки программирования для мобильных платформ. Декларативный UI*	***Содержание учебного материала***		**4**	ПК.2.2., ОК.01, ОК.02, ОК.04, ОК.09
	4	***Языки программирования для мобильных платформ. Декларативный UI*** *Язык Kotlin: синтаксис (data class, sealed class, extension functions), корутины (CoroutineScope, Dispatchers), null safety. Язык Swift: optional binding, closures, struct vs class, async/await, Combine. Язык Dart: sound null safety, async/await, isolates, streams, mixins. Сравнение производительности, кривая обучения, экосистема библиотек. Jetpack Compose: composable функции, рекомпозиция, состояние (remember, mutableStateOf), Modifier. SwiftUI: View protocol, @State, @Binding, @ObservedObject, modifiers. Отличия от императивного подхода (XML + findViewById / UIKit). Композиция и переиспользование компонентов: Slot API (Compose), ViewBuilder (SwiftUI)*	2	
	в том числе, практических занятий		**2**	
	ПЗ №5. *Проектирование архитектуры приложения «Заметки» в MVVM*		2	
Тема 3.2 *Навигация и жизненный цикл. Обработка событий и мультимедиа*	***Содержание учебного материала***		**12**	ПК.2.2., ОК.01, ОК.02, ОК.04, ОК.09
	5	***Навигация и жизненный цикл. Обработка событий и мультимедиа*** *Навигация: Compose Navigation, Navigation Component (Android), NavigationStack (SwiftUI), Navigator 2.0 (Flutter). Глубинные ссылки (Deep Links), маршрутизация, обработка результатов. Жизненный цикл экрана: onCreate/onStart/onResume (Android), viewDidLoad/viewWillAppear (iOS), init/build/dispose (Flutter). Сохранение состояния при*	2	

		повороте: ViewModel, SavedStateHandle, @Parcelize, SceneStorage. Система событий: сенсоры (Touch, Gestures), жесты (Click, Long press, Swipe), обратные вызовы. Камера и галерея: CameraX (Android), AVFoundation (iOS), image_picker (Flutter). Разрешения (Manifest/Info.plist). Аудио и видео: MediaPlayer, ExoPlayer, AVPlayer. Фоновое воспроизведение. Графика: Canvas, рисование примитивов, анимация (AnimatedVisibility, .animation(), Twee		
	в том числе, практических занятий		**10**	
	ПЗ №6. *Реализация экрана входа/регистрации с валидацией полей*		2	
	ПЗ №7. *Верстка списка задач с использованием LazyColumn и карточек*		2	
	ПЗ №8. *Добавление навигации между экранами*		2	
	ПЗ №9. *Работа с камерой*		2	
	ПЗ №10. *Создание анимации переходов и состояний*		2	
Тема 3.3 *Оптимизация интерфейса и графики. Стандарты оформления кода и рефакторинг*	***Содержание учебного материала***		**6**	ПК.2.1., ПК.2.2., ОК.01, ОК.02, ОК.04, ОК.09
	6	***Оптимизация интерфейса и графики. Стандарты оформления кода и рефакторинг*** *Адаптивность: плотность пикселей (dp/dip, pt), разные экраны (small/normal/large/xlarge). Ресурсы для разных конфигураций: layout-land, drawable-hdpi, night mode. Форматы графики: WebP (лучшее сжатие), SVG (масштабирование), VectorDrawable (Android). Типографские правила для интерфейсов: иерархия заголовков, читаемость, переменные шрифты. Соглашения по именованию: Kotlin Coding Conventions, Swift API Design Guidelines, Effective Dart. Статический анализ: detekt (Kotlin), SwiftLint, dart analyze. Рефакторинг: выделение методов, инлайн, замена условного оператора полиморфизмом. Инспекция кода: код-ревью, технический долг, метрики сложности (цикломатическая сложность)*	2	
	в том числе, практических занятий		**4**	
	ПЗ №11. *Создание адаптивного дизайна под планшеты и телефоны*		2	
	ПЗ №12. *Проведение код-ревью, рефакторинга, применение стандартов оформления кода*		2	
	Итого за семестр		**36**	
СЕМЕСТР 6(4)				
Раздел 4. Работа с данными, API, базами данных				
Тема 4.1 *Сетевые запросы и API*	***Содержание учебного материала***		**6**	ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	7	***Сетевые запросы и API*** *REST принципы: ресурсы, методы (GET, POST, PUT, DELETE), коды состояния HTTP. Форматы данных: JSON (сериализация/десериализация через kotlinx. serialization, Codable, json_serializable), XML (не для новых проектов). Клиенты: Retrofit (Kotlin), Alamofire + URLSession (Swift), http + Dio (Dart). Асинхронность: Coroutines (suspend, withContext), async/await (Swift/Dart), RxJava/RxSwift. Обработка ошибок: таймауты, повторные*	2	

		попытки (retrofit-retry), проверка сети (ConnectivityManager/NetworkMonitor		
	в том числе, практических занятий		**4**	
	ПЗ №13. *Запрос к публичному API с отображением через RecyclerView/LazyColumn*		2	
	ПЗ №14. *Запрос к публичному API с отображением через RecyclerView/LazyColumn*		2	
Тема 4.2 *Локальное и облачное хранение данных*	***Содержание учебного материала***		**10**	ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	8	***Локальное и облачное хранение данных*** *Локальное хранение данных. Ключ-значение: SharedPreferences (Android), UserDefaults (iOS). DataStore (Preferences/Proto) для сложных данных. Файловое хранилище: внутренняя/внешняя память, Documents/Application Support, кэш. Реляционные базы: SQLite, Room (Android), Core Data (iOS), sqflite (Flutter). Определения сущностей: @Entity, DAO (Data Access Object), миграции, отношения (OneToMany и т.д.). Firebase и backend-as-a-service. Аутентификация: email/password, Google Sign-In, Apple ID. Токены, сессии. Базы данных Firebase: Realtime Database (JSON-дерево) vs Firestore (документно-коллекционная). Правила безопасности (security rules). Cloud Storage (файлы), Cloud Functions (serverless). Push-уведомления: FCM (Firebase Cloud Messaging), APNs (Apple). Токены, каналы уведомлений*	2	
	в том числе, практических занятий		**8**	
	ПЗ №15. *Реализация сохранения темы через DataStore*		2	
	ПЗ №16. *Создание локальной базы Room для списка дел*		2	
	ПЗ №17. *Интеграция Firebase Auth: регистрация и вход*		2	
	ПЗ №18. *Отправка и получение push-уведомлений (Firebase Cloud Messaging)*		2	
Тема 4.3 *Паттерны работы с данными. Внедрение зависимостей и модульность*	***Содержание учебного материала***		**2**	ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	9	***Паттерны работы с данными. Внедрение зависимостей и модульность*** *Паттерны работы с данными. Repository pattern: абстракция источника данных (локальный/удаленный), единая точка доступа. Кэширование: стратегии (LRU, TTL), библиотеки (OkHttp Cache, CachedNetworkImage). Синхронизация данных: конфликты (последняя запись побеждает, слияние векторов, CRDT). Алгоритмы решения типовых задач: пагинация (offset/limit, курсоры), поиск (индексы, полнотекстовый). Внедрение зависимостей и модульность. Проблема связывания (tight coupling), инверсия управления (IoC). Dagger Hilt: компоненты, скоузы (Singleton, ViewModelScoped), модули. Koin: легковесный DSL, старт приложения. SwiftUI + Swinject: сервис-локатор vs DI-контейнер. Модульная архитектура: feature modules, граф зависимостей*	2	
	в том числе, практических занятий		-	
Раздел 5. Тестирование и отладка приложений			**12**	
Тема 5.1 *Виды, уровни и*	***Содержание учебного материала***		**8**	ПК.2.2.,
	10	***Виды, уровни и инструменты тестирования***	2	ПК.2.3.,

инструменты тестирования		*Виды и уровни тестирования. Модульное тестирование (Unit): изоляция, моки (MockK, Mockito), тестирование ViewModel, use cases. Интеграционное тестирование: взаимодействие модулей, тестирование репозитория с реальной БД. UI-тестирование: Espresso (Android), XCTest UI (iOS), Flutter Driver. Ручное тестирование: чек-листы, тест-кейсы, исследовательское тестирование. Инструменты тестирования. Фреймворки: JUnit 5, TestNG (Android), XCTest (iOS), test (Dart). Генераторы тестовых данных: faker, FactoryBot, генерация граничных значений. Автотесты: Page Object pattern, скриншотное тестирование (Roborazzi, snapshot). CI/CD тестирования: GitHub Actions, GitLab CI, Firebase Test Lab*		ОК.01, ОК.02, ОК.04, ОК.09
	в том числе, практических занятий		**6**	
	ПЗ №19. *Написание юнит-тестов для ViewModel*		2	
	ПЗ №20. *UI-тесты с Espresso / XCTest*		2	
	ПЗ №21. *Составление тест-кейсов и чек-листов, ручное тестирование приложения*		2	
Тема 5.2 *Техники тест-дизайна. Отладка и профилирование*	***Содержание учебного материала***		**4**	ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	11	***Техники тест-дизайна. Отладка и профилирование*** *Классы эквивалентности: разбиение входных данных (валидные/невалидные классы). Анализ граничных значений: мин-1, мин, макс, макс+1. Попарное тестирование (Pairwise): сокращение комбинаций параметров. Техники, основанные на спецификации: таблицы решений, автоматное тестирование. Техники, ориентированные на код: покрытие операторов, ветвей, путей (statement/branch/path coverage). Отладка и профилирование. Логирование: Logcat (Android), OSLog (iOS), debugPrint (Flutter). Уровни (verbose, debug, error). Брейкпоинты: условные, исключения, точки наблюдения. Профилирование: Android Profiler (CPU, Memory, Network), Instruments (iOS), DevTools (Flutter). Типовые метрики: время запуска, кадры в секунду (FPS), утечки памяти (LeakCanary), потребление батареи. Диагностические данные: crash-логи, ANR (Application Not Responding), watchdogs (iOS)*	2	
	в том числе, практических занятий		**2**	
	ПЗ №22. *Сбор и анализ crash-логов, профилирование*		2	
Раздел 6. Документирование и подготовка к внедрению			**10**	
Тема 6.1 *Стандарты оформления документации. Разработка пользовательской и эксплуатационной документации*	***Содержание учебного материала***		**6**	ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	12	***Стандарты оформления документации. Разработка пользовательской и эксплуатационной документации*** *ГОСТ 19.XXX (ЕСПД) и ГОСТ 34.601 (комплексы автоматизированных систем): адаптация под мобильную разработку. Структура технического документа: титульный лист, аннотация, содержание, главы, приложения. Правила оформления рисунков: скриншоты, схемы, подписи, ссылки на рисунки. Форматы электронных документов: PDF (для печати), HTML (для веба), формат электронной справки (CHM, DocSet). Руководства*	2	

		пользователя и эксплуатационная документация. Виды документов для пользователей: руководство по установке, руководство пользователя, учебное пособие, инструкция по настройке. Требования к содержанию: как установить, как зарегистрироваться, основные сценарии, FAQ. Создание гиперссылок: якоря, оглавление с ссылками, перекрестные ссылки. Человеко-ориентированный подход в документации: понятный язык, примеры, глоссарий		
	в том числе, практических занятий		**4**	
	ПЗ №23. *Написание руководства пользователя в текстовом процессоре*		2	
	ПЗ №24. *Оформление технического описания проекта*		2	
Тема 6.2 *Подготовка к публикации в магазины приложений. Методы оценки качества ИТ-продукта*	***Содержание учебного материала***		**4**	ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	13	***Подготовка к публикации в магазины приложений. Методы оценки качества ИТ-продукта*** *Google Play: учетная запись разработчика, подпись ключом (keystore), генерация AAB/APK, трекеры (internal, alpha, beta, production). App Store: Apple Developer Program, сертификаты (Development/Distribution), профили (provisioning), TestFlight, App Store Connect. Политика конфиденциальности, возрастной рейтинг, категории. Требования магазинов: минимальные версии ОС, размер иконок, скриншоты, описание. Метрики качества по ISO 25010: функциональная пригодность, производительность, совместимость, надежность, безопасность, сопровождаемость. Измерение характеристик: время отклика, индекс удобства (SUS), коэффициент ошибок. Оценка соответствия требованиям: трассировочная матрица, приёмочные тесты. Непрерывное улучшение: цикл PDCA, метрики Crash Free Users, пользовательские оценки*	2	
	в том числе, практических занятий		**2**	
	ПЗ №25. *Сборка релизной версии и размещение в эмуляторе магазина*		2	
Раздел 7. Командная работа и презентация проекта			**14**	
Тема 7.1 *Командная работа и взаимодействие с заказчиком. Анализ функциональных разрывов*	***Содержание учебного материала***		**6**	ПК.2.1., ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	14	***Командная работа и взаимодействие с заказчиком. Анализ функциональных разрывов*** *Сбор требований: интервью, анкетирование, наблюдение. Методы выявления требований (MoSCoW, user story mapping). Коммуникации: презентации, переговоры, консультации для разработчиков требований. Протоколы мероприятий: оформление встреч, решений, замечаний. Конфликтология в IT-проектах: типы конфликтов, стратегии разрешения Что такое gap-анализ: ожидаемое против реального состояния системы. Методы выявления недостающей информации: опрос экспертов, анализ документации, прототипирование. Формализация требований: спецификации (SRS), Use Case Specification. Работа с записями по качеству: корректирующие и предупреждающие действия, запросы на исправление несоответствий*	2	

	в том числе, практических занятий	**4**	
	ПЗ №26. *Разработка комплексного приложения в мини-командах*	2	
	ПЗ №27. *Разработка комплексного приложения в мини-командах*	2	
Тема 7.2 *Подготовка к демонстрации и защите проекта. Оценка и метрология в мобильной разработке*	***Содержание учебного материала***	**8**	ПК.2.1., ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	15 ***Подготовка к демонстрации и защите проекта. Оценка и метрология в мобильной разработке*** *Технологии подготовки презентаций: структура. Правила проведения демо: тайминг, live-демо вместо видео, запасной план. Вопросы и ответы: типовые вопросы по архитектуре, тестированию, документации. Критерии оценки по стандартам чемпионата «Профессионалы». Расчёт стоимости проекта: трудозатраты (человеко-часы), ставка, накладные расходы. Методы расчёта выполненных работ: по факту затрат, по этапам (milestone). Математическая статистика для анализа: среднее время выполнения задач, прогнозирование сроков (метод PERT). Основы управленческого учета в IT-проектах: бюджет, освоенный объём (EVM)*	2	
	в том числе, практических занятий	**6**	
	ПЗ №28. *Презентация проекта, демонстрация работоспособности*	2	
	ПЗ №29. *Составление отчёта о тестировании и подготовке данных*	2	
	ПЗ №30. *Защита проекта: ответы на вопросы, демонстрация исходного кода и документации*	2	
Консультации к экзамену	***Содержание***	**6**	ПК.2.1., ПК.2.2., ПК.2.3., ОК.01, ОК.02, ОК.04, ОК.09
	1. Навигация и жизненный цикл. Обработка событий и мультимедиа	2	
	2. Виды, уровни и инструменты тестирования	2	
	3. Стандарты оформления документации. Разработка пользовательской и эксплуатационной документации	2	
Экзамен		**3**	
ВСЕГО:		**99**	

3 УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ

3.1 Для реализации программы учебной дисциплины предусмотрены следующие специальные помещения:

Кабинет «Разработки информационных систем»

оснащен оборудованием:

− рабочее место педагога (преподавательский стол (1 шт.), стул (1 шт.))

− рабочие места обучающихся (парты ученические (13 шт.), стулья ученические (25 шт.))

− доска учебная (меловая трех-секционная) (1 шт.)

− книжный шкаф – 1 шт.;

− учебно-методическая литература по дисциплине;

− комплект учебно-наглядных пособий;

техническими средствами обучения:

− персональный компьютер, подключение к сети Интернет с модулем контентной фильтрации Traffic Inspector, NetPolice и YandexDNS, возможность трансляции на экран аудио и видео информации (1 шт.)

− программное обеспечение на ПК – Microsoft Windows 10 или аналог, Microsoft Office (Word, Excel, PowerPoint) или аналог, 7Zip, 24PDF, Яндекс Браузер (1 шт.)

− программное обеспечение Android Studio, Visual Studio, Visual Studio Code или налоги

− монитор (1 шт.)

− клавиатура (1 шт.)

− мышь (1 шт.)

− телевизор (1 шт.)

− кабель для подключения HDMI (1 шт.)

3.2 Информационное обеспечение реализации программы

Для реализации программы библиотечный фонд института имеет печатные и электронные образовательные и информационные ресурсы, в том числе рекомендованные ФУМО, для использования в образовательном процессе. Список дополнен новыми изданиями.

3.2.1 Основные источники

1. Соколова, В. В. Разработка мобильных приложений : учебник для среднего профессионального образования / В. В. Соколова. — Москва : Издательство Юрайт, 2025. — 160 с. — (Профессиональное образование). — ISBN 978-5-534-16868-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/566082

2. Коржинский, С. Н. Разработка мобильных приложений : учебник / С. Н. Коржинский. — Москва: КноРус, 2025. — 421 с. — ISBN 978-5-406-14290-5. — URL: https://book.ru/book/956945 — Текст : электронный.

3. Попов, А. А. Разработка мобильных приложений : учебник / А. А. Попов. — Москва: КноРус, 2025. — 602 с. — ISBN 978-5-406-14337-7. — URL: https://book.ru/book/957000 — Текст : электронный.

4. Бессмертный, И. А. Интеллектуальные системы : учебник и практикум для среднего профессионального образования / И. А. Бессмертный, А. Б. Нугуманова, А. В. Платонов. — 2-е изд. — Москва : Издательство Юрайт, 2026. — 250 с. — (Профессиональное образование). — ISBN 978-5-534-20730-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/586775

5. Замятина, О. М. Инфокоммуникационные системы и сети. Основы моделирования : учебник для среднего профессионального образования / О. М. Замятина. —

Москва : Издательство Юрайт, 2025. — 167 с. — (Профессиональное образование). — ISBN 978-5-534-17558-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/566086

6. Зараменских, Е. П. Информационные системы: управление жизненным циклом : учебник и практикум для среднего профессионального образования / Е. П. Зараменских. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 486 с. — (Профессиональное образование). — ISBN 978-5-534-21416-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/571329

3.2.2 Дополнительные источники

7. Черпаков, И. В. Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 196 с. — (Профессиональное образование). — ISBN 978-5-534-18760-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/584552

8. Гниденко, И. Г. Технология разработки программного обеспечения : учебник для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 248 с. — (Профессиональное образование). — ISBN 978-5-534-18131-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/585518

4 КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ

Результаты обучения	Критерии оценки	Методы оценки
− *жизненный цикл мобильного приложения и особенности ОС;* − *методы алгоритмизации, формализации и нотации;* − *проектирование UI/UX и прототипирование;* − *основы вёрстки, типографики и графики;* − *современные языки программирования, используемые в мобильной разработке и их синтаксис;* − *архитектурные паттерны и управление состоянием;* − *базы данных и сетевое взаимодействие;* − *методы тестирования и тест-дизайна;* − *стандарты оформления кода и документации;* − *метрики качества и оценка трудозатрат;*	**Оценка «5»:** Анализирует сценарии с системными событиями (звонок, сворачивание, низкий уровень памяти, изменение конфигурации). Объясняет работу фоновых задач и разрешений. Сравнивает подходы к управлению памятью; Самостоятельно строит диаграммы деятельности для сложных сценариев. Применяет конечные автоматы для проектирования экранов. Использует таблицы решений для бизнес-логики. Обосновывает выбор нотации под конкретную задачу; Анализирует существующий интерфейс на предмет эргономики и доступности (WCAG). Проектирует интерактивный прототип с учётом анимации и жестов. Обосновывает выбор UI-паттерна. Проводит оценку удобства (SUS-опросник); Анализирует макет на соответствие платформенным гайдлайнам. Обосновывает выбор формата графики для разных сценариев. Оптимизирует графику под разрешения. Применяет переменные шрифты и кастомную типографику. Сравнивает языки по характеристикам (производительность, кривая обучения, экосистема). Объясняет работу корутин (Kotlin) / async-await (Swift/Dart) на уровне планировщиков и диспетчеров. Понимает generics, sealed classes, pattern matching. Может прочитать и объяснить чужой код на любом из трёх языков; Проектирует архитектуру приложения под конкретные требования. Обосновывает выбор паттерна. Реализует внедрение зависимостей вручную или через Dagger/Koin/Swinject. Понимает многопоточность и безопасность состояния. Анализирует компромиссы (сложность vs тестируемость). Проектирует схему локальной БД с учётом нормализации и производительности. Реализует кэширование (LRU, TTL) и синхронизацию с разрешением конфликтов. Настраивает безопасные правила Firestore. Объясняет пагинацию (offset/limit, курсоры). Оптимизирует сетевые запросы (retry, timeout, кэширование). Составляет тест-план и чек-листы для мобильного приложения. Применяет техники, основанные на спецификации и на интуиции (исследовательское тестирование). Анализирует покрытие кода (statement, branch, path). Понимает регрессионное	**Текущий контроль:** Компьютерное тестирование по основным разделам программы. Решение задач по основным разделам программы **Промежуточная аттестация:** экзамен

	тестирование и автоматизацию (CI/CD пайплайны). Применяет статические анализаторы (detekt, SwiftLint). Пишет самодокументируемый код с KDoc/ Swift-DocC. Оформляет техническую документацию по ГОСТ (титульный лист, рисунки, ссылки, оглавление). Создаёт гипертекстовую справку (HTML, PDF). Рассчитывает стоимость проекта на основе трудозатрат и ставки. Применяет EVM (освоенный объём) для контроля исполнения. Интерпретирует метрики производительности (профилирование памяти, CPU, батареи). Обосновывает приоритет задач по бизнес-ценности (MoSCoW, Kano). **Оценка «4»:** Объясняет, когда и зачем вызывается каждый метод жизненного цикла. Понимает, как сохранять и восстанавливать состояние при повороте экрана. Различает особенности iOS и Android; Умеет формализовать простую задачу в виде алгоритма. Различает основные виды UML-диаграмм. Понимает нотации BPMN и IDEF0 на базовом уровне. Объясняет принципы человеко-ориентированного проектирования. Знает инструменты прототипирования (Figma, Sketch). Понимает принципы Material Design и Human Interface Guidelines. Объясняет принципы адаптивной вёрстки под разные экраны и плотности пикселей. Знает базовые правила типографики. Понимает теорию цвета; Понимает ООП (классы, наследование, полиморфизм, инкапсуляцию) на выбранном языке. Объясняет null safety (Kotlin/Swift/Dart). Знает функциональные возможности (lambda, higher-order functions, extension functions). Объясняет различия между паттернами, их сильные и слабые стороны. Понимает реактивное программирование (StateFlow, SharedFlow, Combine, Rx). Знает принципы Clean Architecture (слои данных, домена, представления). Объясняет принципы работы Room (сущности, DAO, миграции). Понимает асинхронность (корутины, async/await). Знает форматы JSON/XML и умеет их парсить. Понимает основы Firebase (аутентификация, Firestore). Объясняет техники тест-дизайна (классы эквивалентности, граничные значения, попарное тестирование). Знает инструменты (JUnit, Espresso, XCTest). Понимает разницу между моками и стабами.	

	Объясняет основные правила именования, форматирования, комментирования. Знает структуру руководства пользователя. Понимает требования ГОСТ к технической документации. Объясняет модель качества ISO 25010 (функциональность, производительность, надёжность, сопровождаемость). Умеет оценивать задачу методом аналогий или PERT. Понимает метрики Crash Free Users, ANR, FPS. **Оценка «3»:** Знает названия основных состояний жизненного цикла Activity/ViewController. Понимает разницу между foreground и background; Знает, что такое блок-схема и псевдокод. Может прочитать простую UML-диаграмму последовательности; Знает назначение прототипирования. Понимает разницу между wireframe, mockup и прототипом. Знает разницу между растровой и векторной графикой. Понимает, что такое dp/dip и pt. Может назвать основные форматы (PNG, JPEG, SVG, WebP); Знает синтаксис базовых конструкций (переменные, циклы, условия, функции, массивы) на одном из языков (Kotlin/Swift/Dart). Знает названия паттернов MVC, MVP, MVVM. Понимает, зачем нужен ViewModel. Знает, что такое SQLite и REST API. Понимает разницу между GET и POST. Знает виды тестирования (модульное, интеграционное, UI). Понимает, что такое тест-кейс. Знает, что существуют стандарты оформления кода (Kotlin Coding Conventions, Swift API Design Guidelines). Понимает, зачем нужен Git. Знает, что такое трудозатраты в человеко-часах. Понимает метрику «время запуска приложения».	
− *разрабатывать пользовательский интерфейс;* − *программировать на современном мобильном языке;* − *проектировать архитектуру приложения;* − *работать с локальными базами данных;* − *интегрировать REST API и Firebase;* − *тестировать*	**Оценка «5»** Разрабатывает интерфейс, полностью следующий платформенным гайдлайнам (Material Design / HIG). Реализует сложную анимацию (переходы, shared elements, Lottie). Интерфейс доступен (TalkBack/VoiceOver, контрастность, крупные элементы). Использует кастомные View/Composable. Использует расширенные возможности языка: generics, sealed classes, extension functions, корутины/async-await, потоки (Flow/Combine). Код модульный, переиспользуемый, соответствует индустриальным стандартам.	**Текущий контроль:** Оценка результатов выполнения практической работы Экспертное наблюдение за ходом выполнения практической работы (деятельностью студента) Оценка выполнения практического задания (работы)

приложение; − *отлаживать и профилировать приложение;* − *оформлять техническую документацию;* − *оценивать трудозатраты и планировать работу;* *работать в команде и презентовать результат.*	Пишет самодокументируемый код с комментариями там, где логика неочевидна. Строит Clean Architecture (слои data, domain, presentation). Настраивает полноценное DI (Dagger/Hilt/Koin/Swinject). Управляет состоянием с помощью реактивных потоков. Проект готов к масштабированию (модули, feature-флаги, мультимодульность). Оптимизирует производительность БД (индексы, асинхронные запросы, транзакции). Реализует отношения One-to-Many, Many-to-Many. Настраивает предзаполнение БД (prepopulated). Понимает и избегает проблем с блокировками потока Реализует кэширование ответов (OkHttp Cache, offline-first). Настраивает повторные попытки (retry with backoff). Интегрирует Firebase Auth с Google/Apple Sign-In. Использует Firestore с правилами безопасности и реальным временем (snapshots). Отправляет push-уведомления (FCM/APNs). Пишет интеграционные тесты (взаимодействие с БД/API) и UI-тесты (Espresso/XCTest). Применяет техники тест-дизайна (граничные значения, классы эквивалентности). Использует CI для автоматического запуска тестов (GitHub Actions). Анализирует покрытие кода (не менее 70% для критической логики). Находит и устраняет утечки памяти (LeakCanary, Memory Graph). Анализирует производительность (FPS, время запуска, задержки). Снимает и анализирует heap dump. Работает с системными логами (logcat, OSLog) и фильтрует их. Оптимизирует код по результатам профилирования. Составляет техническую документацию по ГОСТ или корпоративному шаблону (титульный лист, аннотация, разделы, приложения). Вставляет схемы архитектуры и диаграммы. Создаёт гиперссылки (оглавление с переходами). Оформляет документацию в нескольких форматах (PDF, HTML, электронная справка). Применяет story points в команде. Учитывает риски и непредвиденные работы. Составляет диаграмму Ганта. Планирует спринты и декомпозирует эпики на задачи. Оценивает трудоёмкость в человеко-часах с обоснованием Проводит полноценную презентацию проекта (проблема → решение → архитектура → демо → результаты). Участвует в код-ревью (проверяет чужой код и исправляет замечания по-своему). Интервьюирует «заказчика» для сбора требований. Составляет протоколы встреч и решений. Работает в паре (pair	**Промежуточная аттестация:** экзамен

	programming) эффективно. **Оценка «4»** Верстает адаптивный интерфейс под разные размеры и ориентации (ConstraintLayout, Compose, SwiftUI). Использует кастомные стили, темы, тёмную/светлую тему. Обрабатывает базовые жесты (tap, long press, swipe). Применяет ООП (классы, наследование, интерфейсы/протоколы). Использует коллекции, лямбда-выражения, обработку исключений. Пишет читаемый код с осмысленными именами. Понимает null safety и применяет безопасные вызовы. Реализует MVVM с ViewModel, LiveData/StateFlow. Отделяет бизнес-логику от UI. Использует репозиторий для работы с данными. Применяет внедрение зависимостей вручную или через простой DI-контейнер. Самостоятельно проектирует схему БД (сущности, типы полей, первичные ключи). Пишет сложные запросы (@Query с условиями, JOIN). Выполняет миграции при изменении структуры. Реализует все типы запросов (GET, POST, PUT, DELETE). Обрабатывает ошибки сети (таймауты, коды 4xx/5xx). Парсит JSON с помощью сериализатора (kotlinx.serialization, Codable). Настраивает базовую аутентификацию через Firebase (email/password). Пишет юнит-тесты для ViewModel с моками зависимостей (MockK/Mockito). Составляет тест-кейсы для ручного тестирования (чек-лист). Выполняет ручное тестирование по чек-листу, фиксирует баги. Использует условные брейкпоинты, точки наблюдения (watchpoints). Читает stack trace и находит место ошибки. Запускает профилировщик (Android Profiler / Instruments) и видит базовые метрики (CPU, память). Оформляет руководство пользователя с оглавлением, скриншотами экранов и подписями к ним. Использует стили форматирования (заголовки, списки) в текстовом процессоре. Сохраняет документ в PDF. Использует метод аналогий (сравнивает с похожей задачей) или PERT (оптимистичная, пессимистичная, наиболее вероятная оценка). Составляет календарный план с датами. Работает с системой контроля версий (Git: commit, push, pull). Взаимодействует с другими членами команды через Git (разрешение конфликтов, pull requests). Проводит мини-презентацию (5–7	

	минут) с демонстрацией работающего приложения. Отвечает на вопросы по функционалу. **Оценка «3»** Создаёт экран с базовыми элементами (кнопки, текстовые поля, списки) с помощью визуального редактора или простейшей вёрстки. Интерфейс работает, но может «плыть» на разных экранах. Пишет работающий код с использованием базовых конструкций: переменные, циклы, условия, функции, массивы/списки. Код содержит незначительные синтаксические ошибки, но выполняется. Разделяет код на «всё в одном Activity/Fragment/View» и хотя бы один отдельный класс (например, для работы с данными). Выполняет базовые CRUD-операции через готовый DAO (пример из документации). Приложение сохраняет и загружает данные из SQLite/Room. Выполняет GET-запрос к публичному API и отображает полученные данные в списке. Использует готовый клиент (например, Retrofit) с минимальной настройкой. Пишет хотя бы один простой юнит-тест (проверка метода без зависимостей). Запускает тесты в среде разработки. Использует print() / Log.d() для вывода сообщений. Умеет ставить простой брейкпоинт и смотреть значения переменных. Пишет простой текстовый файл (README), где перечислены основные функции приложения и как его запустить. Прикидывает время на задачу «на глаз» в часах. Составляет простой список задач (что сделать). Участвует в командной работе (делает свою часть). Может коротко рассказать, что сделано и как это запустить.	