Сборник заданий для практических занятий
по курсу
«Разработка приложений для мобильных устройств» (МПГУ;
группа 425, 2025-2026)

Содержание

1 Общие сведения

Сборник содержит задания для практических занятий по курсу «Разработка мобильных приложений».

Лекции проводятся в форме мастер-классов, поэтому практические занятия заключаются в выполнении заданий, аналогичных по структуре работам, рассмотренным на мастер-классах.

В практических и лабораторных работах используется язык Kotlin в среде Android Studio. При этом необходимо соблюдать Coding Conventions (требования к стилю кода).

Перед сдачей работы необходимо убедиться, что среда не выдаёт предупреждений при запуске подпункта **Inspect Code** меню **Analyze**.

Во всех заданиях требуется проверять корректность входных данных: программа не должна «падать» ни при каких условиях.

Для групп, где промежуточная аттестация проводится в форме зачёта: зачёт имеет накопительный характер; оценивается каждое практическое задание (которое подлежит защите), строго учитываются сроки сдачи и, в некоторых случаях, качество выполнения. Для групп, где промежуточная аттестация проводится в форме экзамена, экзамен представляет собой защиту проекта (это может быть либо совокупность практических заданий с занятий, либо индивидуальный проект, предварительно согласованный с преподавателем).

В каждом задании предусмотрено 25 вариантов. Выбирайте вариант согласно вашему номеру в журнале.

2 Практические задания (осенний семестр)

2.1 Практическое задание «Простейшие программы для Android»

Разработайте программу, работающую под управлением Android с использованием Jetpack Compose. Проверьте, что программа корректно работает с различными размерами экрана, а также при повороте экрана.

Необходимо учитывать интернационализацию (приложение должно корректно работать как при настройке локали с русским языком по умолчанию, так и английским).

Необходимо использовать принцип DRY (Don't repeat yourself).

Проверяйте и информируйте пользователя обо всех ошибках, встречающихся при вводе (подсвечивая, в том числе, элементы управления) и при подсчетах (в частности переполнение).

1. Программа решения квадратного уравнения
2. Программа решения неравенства вида $ax + b > 0$
3. Программа решения неравенства вида $ax + b < 0$
4. Программа решения неравенства вида $ax + b \geqslant 0$
5. Программа решения неравенства вида $ax + b \leqslant 0$
6. Программа поиска дня недели по числу и месяцу в текущем году
7. Программа перевода числа из 10-ой в 16-ую, 8-ую и 2-ую систем.
8. Программа поиска времени, когда окончится интервал. Дано: часы и минуты начала интервала и количество минут, сколько он идет. Результат: часы и минуты окончания интервала.
9. Программа поиска обратной матрицы для матрицы 3×3.
10. Программа поиска длины интервала. Дано: часы и минуты начала интервала и часы и минуты конца интервала. Результат: количество минут в интервале.
11. Программа умножения и деления двух комплексных чисел.
12. Программа нахождения площади треугольника по координатам вершин.
13. Программа нахождения углов треугольника по координатам вершин (проще всего это сделать по теореме косинусов).
14. Программа перевода числа из 16-ой, 8-ой и 2-ой системы в 10-ую систему счисления.
15. Программа нахождения количества денег на вкладе после окончания его срока по начальному взносу, проценту и срока в годах.
16. Программа нахождения степени комплексного числа. Исходные данные: действительная, мнимая часть числа и степень. Результат: действительная и мнимая часть результата.
17. Программа умножения и деления чисел, представленных в виде обыкновенных дробей (состоящих из целой части, числителя и знаменателя). Не забудьте выполнить сокращение дроби и приведение ее к правильному виду.

18. Программа сложения и вычитания чисел, представленных в виде обыкновенных дробей (состоящих из целой части, числителя и знаменателя). Не забудьте выполнить сокращение дроби и приведение ее к правильному виду.

19. Программа определения по дате (число и месяц) знака зодиака.

20. Программа определения по обыкновенной дроби (числителю и знаменателю) периода десятичной дроби.

21. Программа перевода комплесного числа из обычной формы в тригонометрическую и наоборот.

22. Программа-игра Баше. При реализации этого задания не требуется ничего рисовать, вся информация вводится и выводится в виде чисел в обычные элементы управления.

23. Программа разложения числа на простые множители.

24. Программа нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел.

25. Программа-тест по предмету «Разработка мобильных приложений». Создайте программу-тест из 10 вопросов с выбором вариантов ответов и показом результатов прохождения теста.

2.2 Лабораторная работа «Проектирование приложения с использованием MVVM» (2 часа)

1. Осуществите рефакторинг проекта, полученного в предыдущей работе, путём внедрения паттерна MVVM. Обратите внимание на то, что для реализации прослушивания изменений значений во ViewModel (и в Model при необходимости) необходимо использовать StateFlow. Используйте библиотеки Jetpack везде, где они применимы.

2. Реализуйте автоматическое UNIT-тестирование класса (объекта) модели и класса ViewModel. Осуществите необходимый рефакторинг. UNIT-тесты должны соответствовать всем стандартными принципам и иметь полное покрытие.

3. Реализуйте автоматечское интеграционное тесирование модели и ViewModel.

4. Реализуйте автоматическое UI-тестирование приложения.

2.3 Практическая работа «Работа со списками»

Реализуйте приложение с использованием паттерна MVVM, предназначенное для ввода, изменения, удаления и просмотра информации согласно вашему варианту.

Требования к реализации:

- необходимо использовать Navigation Compose;
- необходимо создать столько ViewModel, сколько и страниц;
- необходимо строго соблюдать требования к ViewModel, Model и View (даже строже, чем это предлагают авторы Android).

1. База данных студентов группы. Поля: фамилия, имя, отчество, пол, возраст.
2. База данных расходов семьи. Поля: товар, стоимость, количество, дата.
3. База данных загрузки аудиторий. Поля: дата и время, начала, дата и время конца, аудитория, преподаватель.
4. База данных учета доходов и расходов предпринимателя. Поля: дата, тип операции (доход/расход), объем операции, описание, корреспондент.
5. База данных велоклуба. Поля: ФИО, тип велосипеда (MTB и др.), стаж участия в велоклубе.
6. База данных рейсов авиакомпании. Поля: дата и время вылета, аэропорт вылета, аэропорт прилета, дата и время прилета, марка самолета.
7. База данных автобусных маршрутов. Поля: номер маршрута, номер парка, времена начала и окончания движения, длина маршрута в км.
8. База данных электричек. Поля: вокзал, номер поезда, количество вагонов, тип (экспресс/обычный/спутник).
9. База данных товаров Интернет-магазина. Поля: название товара, категория, цена товара, описание товара.
10. База заказов Интернет-магазина. Поля: ФИО заказчика, стоимость заказа, скидка (в процентах), адрес доставки.
11. База данных выборов. Поля: участок, кандидат, количество голосов.
12. База данных практических работ. Поля: практическая работа, студент, номер варианта, номер уровня, дата сдачи, оценка.
13. База данных операторов и телеканалов. Поля: Название, тип (спутник, кабель, Интернет), охват (кол-во миллионов домохозяйств), минимальная стоимость подписки.
14. База данных тарифных планов оператора. Поля: название, тип вещания (обычный-/HD), флаг общедоступности.
15. База данных незаконно огороженных берегов. Поля: водный объект, регион, GPS-координаты, длина недоступного участка берега, дата фиксации нарушения.

16. База данных временного прекращения движения в метро. Поля: дата и время начала прекращения движения, дата и время окончания прекращения движения, станция, станция (от какой до какой станции прекращено движение).

17. База данных проката фильмов. Поля: дата, время, кинотеатр, фильм, номер зала, тип сеанса (3D/Imax/обычный).

18. База данных эвакуированных автомобилей. Поля: улица, автостоянка, GPS-координаты, тип нарушения (стоянка на проезжей части в месте запрета, стояна на тротуаре, стоянка на газоне), номер автомобиля, тип автомобиля (легковой/грузовой малой тонажности/грузовой большой тонажности).

19. База данных средних специальных учебных учреждений. Поля: название, адрес, тип подчинения (федеральный/региональный), год основания, номер лицензии, номер аккредитации, дата окончания действия аккредитации.

20. База данных поселков. Поля: название, девелопер, площадь, количество жителей.

21. База данных сухопутной военной техники. Поля: название, модель, разработчик, предприятие, стоимость, тип.

22. База данных деревьев в городе. Поля: GPS-координаты, вид дерева, округ, год посадки.

23. База данных футбольных матчей. Поля: дата, команда, команда, счет, место проведения.

24. База данных обращений жителей. Поля: дата, время, объект, заявитель, содержание обращения (до 255 символов), дата ответа, ответ на обращение (до 255 символов).

25. База данных студентов колледжа. Поля: ФИО, группа, признак бюджетности, стипендия (нет/обычная/повышенная), флаг наличия социальной стипендии, дата рождения.

2.4 Семинар «Работа с базами данных»

Внедрите в приложение, полученное на прошлом занятии, постоянное хранение с использованием библиотеки Room, а также реализуйте частично архитектуру Clear Architecture (в части UseCase и Repository).

Также реализуйте автоматическое внедрение зависимостей посредством Hilt, а также реализуйте автоматическое тестирование usecases.

2.5 Лабораторная работа «Unit-тесты и внедрение зависимостей» (1/2 недели)

Внедрите в приложение, полученное на прошлом занятии, автоматическое внедрение зависимостей посредством Hilt (если требуется) а также реализуйте автоматическое тестирование всех классов приложения, созданных Вами, кроме Activity с полным покрытием, а также UI-тестирование (кроме Room для группы 425).

2.6 Зачёт

Зачёт выставляется посредством вычисления средней арифметической оценки по всем лабораторным работам с учетом:

- сроков сдачи,
- полного понимания представленного исходного кода (как были написаны самописные части кода, откуда брались части кода, что есть в документации, как в целом производится работа с проектом в среде разработки, полное понимание всех использованных конструкций языка и классов/функций фреймворков и библиотек),
- доли выполнения задания по отношению к идеально выполненному заданию.

3 Практические задания (весенний семестр)

3.1 Лабораторная работа «UI-тестирование и работа с Rest API» (2 недели)

Цель работы: Реализовать Android-приложение с чистой архитектурой и UNIT- и UI-тестами, реализующего доступ к публичному сервису посредством REST API.

- Приложение должно быть реализовано в соответствии с принципами **чистой архитектуры** (Clean Architecture) и использовать паттерн **MVVM**.
- Для внедрения зависимостей обязателен **Hilt**.
- Для сетевых запросов используется библиотека **Retrofit** (в фоне, через `Coroutine`).
- Все сетевые ответы должны **кешироваться локально** (с помощью Room), чтобы обеспечивать работу приложения при отсутствии интернет-соединения.
- Обязательна реализация как **Unit-тестов** (для Repository, UseCase и т.д.), так и **UI-тестов**.
- Используйте Navigation для переходов между экранами.

Важные примечания

- Перед началом работы проверьте работоспособность выбранного API.
- Если выбранный API оказался неработоспособен, замените его на другой из списка ниже. Сайты-заменители у разных студентов не должны совпадать.
- Для всех запросов к публичным API **обязательно указывайте заголовок** `User-Agent: YourAppName/1.0` (требование многих сервисов).
- Все варианты содержат как минимум один параметризованный запрос и подходят для кеширования.

Задания Реализуйте клиент для **одного** из следующих REST API. Каждое задание предполагает:

- наличие хотя бы одного параметризованного запроса;
- отображение результата в UI;
- сохранение истории или кэша для офлайн-доступа.

1. **Open-Meteo Weather API** — прогноз погоды (температура, осадки) по координатам.
   `https://api.open-meteo.com/v1/forecast?latitude=55.75&longitude=37.62&hourly=temperature_2m`
2. **TVMaze API** — поиск телешоу по названию (`GET /search/shows?q=sherlock`).
   `https://api.tvmaze.com/search/shows?q=sherlock`
3. **Open Trivia Database** — вопросы для викторины по категории и сложности (`GET /api.php?amount=5&category=9&difficulty=easy`).
   `https://opentdb.com/api.php?amount=5&category=9&difficulty=easy`

4. **BoredAPI** — получение предложения занятия по типу и числу участников (`GET /activity?type=educatio`
   `https://www.boredapi.com/api/activity?type=education&participants=1`
5. **Open Library API** — поиск книг по ключевому слову (`GET /search.json?q=python`).
   `https://openlibrary.org/search.json?q=python`
6. **Open Library API** — список произведений по имени автора (`GET /search/authors.json?q=tolkien`).
   `https://openlibrary.org/search/authors.json?q=tolkien`
7. **JokeAPI** — получение шутки по категории (`GET /joke/Programming?type=single`).
   `https://v2.jokeapi.dev/joke/Programming?type=single`
8. **Random User API** — генерация случайного профиля пользователя с параметрами (`GET /?nat=RU&gender=female`).
   `https://randomuser.me/api/?nat=RU&gender=female`
9. **PokéAPI** — получение данных о покемоне по имени или идентификатору (`GET /pokemon/{name}`).
   `https://pokeapi.co/api/v2/pokemon/pikachu`
10. **Advice Slip API** — поиск жизненных советов по ключевому слову (`GET /advice/search/{query}`).
    `https://api.adviceslip.com/advice/search/love`
    Примечание: обязательно указывать заголовок `User-Agent`*.*
11. **TheMealDB** — поиск рецептов по ингредиенту, названию или категории (`GET /filter.php?i=chicken_br`
    `https://www.themealdb.com/api/json/v1/1/filter.php?i=chicken_breast`
12. **DictionaryAPI** — получение определений английского слова (`GET /entries/en/{word}`).
    `https://api.dictionaryapi.dev/api/v2/entries/en/hello`
13. **ExchangeRate-API** — курсы валют относительно базовой валюты (`GET /latest/{currency}`).
    `https://open.er-api.com/v6/latest/USD`

3.2 Лабораторная работа «Расширенные сценарии работы с данными: отложенное получение данных, уведомления» (1 неделя)

Цель работы: Расширить приложение из предыдущей лабораторной работы механизмами обработки отсутствия сети и оповещения пользователя через уведомления.

- При отсутствии сети пользовательский запрос сохраняется в историю со статусом «ожидает» и ставится в очередь на фоновую загрузку через **WorkManager**.
- `Worker` запускается автоматически при появлении сети (через `Constraints`) и использует существующий `Repository` из первой части работы.
- По завершении фоновой загрузки показывается **уведомление** с результатом (успех/ошибка) и действием для перехода к данным.
- Реализуется экран **истории запросов** с отображением статуса каждого запроса («в процессе», «успешно», «ошибка»).

Обязательные компоненты

1. **Единая таблица истории запросов**
   Расширьте существующую или создайте новую таблицу `search_history` с полями:
   - `id` — автоинкрементный идентификатор
   - `query_params` — параметры запроса (например, имя покемона, сериализовано в JSON)
   - `timestamp` — время запроса
   - `status` — статус: `PENDING` / `SUCCESS` / `FAILED`
   - `result_id` — идентификатор сохранённого результата в основной таблице кеша (для перехода)
2. **WorkManager для фоновой загрузки**
   Реализуйте `DataFetchWorker`, который:
   - Выполняет запрос через существующий `Repository`
   - При успехе: обновляет статус запроса на `SUCCESS`, сохраняет результат в кеш (как в первой части)
   - При ошибке сети: завершается со статусом `Retry` (WorkManager применит бэкофф автоматически)
   - При 3 неудачах: обновляет статус на `FAILED`
   - Показывает уведомление только при финальном результате (`SUCCESS` или `FAILED`)
3. **Уведомления с действиями**
   Уведомления должны содержать:
   - Заголовок: «Данные получены» / «Ошибка загрузки»
   - Текст: краткое описание (например, «Покемон pikachu загружен»)
   - Действие «Открыть» — `PendingIntent` для перехода на экран результата (только при успехе)
   - Действие «Повторить» — запуск нового `DataFetchWorker` с теми же параметрами (только при ошибке)

3.3 Лабораторная работа «Интеграция с экосистемой Android: deep links, обмен данными»

Цель работы: Расширить Compose-приложение механизмами взаимодействия с другими приложениями и внешними источниками: приём данных через ссылки и системное меню «Поделиться», отправка результатов.

- Приложение должно обрабатывать **Deep Links** (схема `myapp://`) и HTTPS-ссылки для мгновенного открытия контента (заполнения формы поиска)
- Приложение должно выступать получателем данных через системное меню «Поделиться» (**Share Target**) с обработкой текста (содержащего URL или текст для поиска)
- Должна быть реализована отправка результатов в другие приложения через `Intent.ACTION_SEND`.
- Обязательна базовая валидация внешних данных.

3.4 Экзамен

Экзамен проводится в форме защиты проекта, реализованного в течение весеннего семестра или собственного проекта, который разработан для Android с использованием Jetpack Compose с обязательной реализацией чистой архитектуры, кеширование в Room, использованием Worker (ожидание сетевого соединения, политика повторных обращений к сервису, взаимодействие с другими приложениями (deep links, web links, send action).

4 Список литературы

1. Официальная документация по языку Kotlin: `https://kotlinlang.org/docs/reference/`
2. Официальная документация по платформе Android: `https://developer.android.com/docs`
3. Быстрое введение в Kotlin от авторов языка: `https://stepik.org/course/4222/promo`