

Вопросы к экзамену по разработке мобильных приложений

В билете должно быть два теоретических и один практический вопрос

Теоретические вопросы

- Язык Kotlin: особенности языка
- Язык Kotlin: основные типы данных
- Язык Kotlin: концепции и элементы языка для поддержки null-безопасности
- Язык Kotlin: синтаксис и семантика условных выражений языка
- Язык Kotlin: синтаксис и семантика циклов
- Язык Kotlin: синтаксис и семантика создания обычных функций
- Язык Kotlin: синтаксис и семантика создания lambda-функций
- Язык Kotlin: функции высшего порядка, понятие и примеры
- Язык Kotlin: массивы
- Язык Kotlin: коллекции
- Язык Kotlin: функции, предназначенные для преобразования коллекций
- Язык Kotlin: функции, предназначенные для агрегирования коллекций
- Язык Kotlin: способы создания sequence

- Язык Kotlin: сравнение коллекций и sequence
- Язык Kotlin: особенности реализации объектно-ориентированной парадигмы в структуре класса
- Язык Kotlin: свойства (properties) класса
- Язык Kotlin: особенности реализации объектно-ориентированной парадигмы в наследовании
- Язык Kotlin: делегирование классов
- Язык Kotlin: делегирование свойств, lazy
- Язык Kotlin: поддержка парадигмы абстрактного программирования, контра- и ковариантность
- Язык Kotlin: поддержка параллельного программирования посредством корутин
- Язык Kotlin: передача данных между корутинами с помощью Flow
- Язык Kotlin: StateFlow и SharedFlow
- Язык Kotlin: перегрузка операторов
- Сравнение функционального и императивного стиля в разработке
- Разработка приложений под Android: основные компоненты
- Разработка приложений под Android: методика создания приложений
- Разработка приложений под Android: структура проекта
- Разработка приложений под Android: роль и назначение Manifest
- Разработка приложений под Android: методика использования Gradle в Android-разработке
- Разработка приложений под Android: локализация приложения
- Разработка приложений под Android: работа с базой данных, библиотека Room
- Паттерн репозиторий

- Dependency Injection: понятие, библиотека Dagger
- Dependency Injection: библиотека Hilt, отличия от Dagger
- Разработка приложений под Android: основные принципы разработки с использованием Jetpack compose
- Разработка приложений под Android: компоненты Jetpack compose, предназначенные для вёрстки
- Разработка приложений под Android: компоненты Jetpack compose, предназначенные для ввода-вывода
- Разработка приложений под Android: понятие, назначение, способ реализации сервисов
- Разработка приложений под Android: понятие, назначение и способы использования Intent
- Разработка приложений под Android: понятие, назначение и способ использования Navigation
- Разработка приложений под Android: понятие, назначение и способ использования оповещений
- Разработка приложений под Android: библиотека Retrofit
- Разработка приложений под Android: автоматическое Unit-тестирование
- Разработка приложений под Android: автоматическое UI-тестирование
- Методы создания мобильных приложений
- Разработка приложений под Android: понятие, назначение и цикл жизни Activity
- Архитектурный шаблон MVVM: описание и структура программы на примере Android-приложения
- Архитектурный шаблон MVVM: использование StateFlow при реализации MVVM
- Разработка приложений под Android: особенности параллельного программирования

Практические задания

Задание 1

Задание на разработку клиента для REST API сервера

Необходимо разработать клиент для взаимодействия с сервером, предоставляющим REST API для получения данных о музыкальных альбомах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Для выполнения запросов к API необходимо использовать библиотеку `Retrofit`. Пример запроса:

```
GET http://10.0.2.2:9080/search?title=blue
```

Ответ от сервера возвращается в формате JSON и содержит данные о музыкальных альбомах:

```
[
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
  }
]
```

Локальное хранение данных

Для хранения данных о музыкальных альбомах необходимо использовать библиотеку `Room`. После получения данных с сервера они должны быть сохранены в локальной базе данных, чтобы быть доступными для работы в офлайн-режиме.

Архитектура приложения

Использовать архитектуру MVVM (Model-View-ViewModel) для разделения бизнес-логики и пользовательского интерфейса. Компонент `ViewModel` должен выполнять следующие функции:

- Выполнение запросов к серверу через библиотеку **Retrofit**.
- Сохранение данных в локальной базе данных с использованием **Room**.
- Передача данных в интерфейс приложения.

Интерфейс пользователя

Использовать библиотеку **Jetpack Compose** для создания пользовательского интерфейса.

При отсутствии интернет-соединения необходимо отображать данные из локальной базы данных.

Задание 2

Задание на разработку клиента для REST API сервера

Необходимо разработать клиент для взаимодействия с сервером, предоставляющим REST API для получения данных о гражданах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Для выполнения запросов к API необходимо использовать библиотеку **Retrofit**. Пример запроса:

`GET http://10.0.2.2:9080/search?name=john`

Ответ от сервера возвращается в формате JSON и содержит данные о гражданах:

```
[
  {
    "id": "1",
    "name": "John Smith",
    "age": 35,
    "address": "123 Main St"
  }
]
```

Локальное хранение данных

Для хранения данных о гражданах необходимо использовать библиотеку **Room**. После получения данных с сервера они должны быть сохранены в локальной базе данных, чтобы быть доступными для работы в офлайн-режиме.

Архитектура приложения

Использовать архитектуру MVVM (Model-View-ViewModel) для разделения бизнес-логики и пользовательского интерфейса. Компонент **ViewModel** должен выполнять следующие функции:

- Выполнение запросов к серверу через библиотеку **Retrofit**.
- Сохранение данных в локальной базе данных с использованием **Room**.
- Передача данных в интерфейс приложения.

Интерфейс пользователя

Использовать библиотеку **Jetpack Compose** для создания пользовательского интерфейса.

При отсутствии интернет-соединения необходимо отображать данные из локальной базы данных.

Задание 3

Задание на разработку клиента для REST API сервера

Необходимо разработать клиент для взаимодействия с сервером, предоставляющим REST API для получения данных о гражданах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Для выполнения запросов к API необходимо использовать библиотеку **Retrofit**. Пример запроса:

GET <http://10.0.2.2:9080/search?minAge=25&maxAge=35>

Ответ от сервера возвращается в формате JSON и содержит данные о гражданах:

```
[
  {
    "id": "1",
    "name": "John Smith",
    "age": 35,
    "address": "123 Main St"
  }
]
```

Локальное хранение данных

Для хранения данных о гражданах необходимо использовать библиотеку **Room**. После получения данных с сервера они должны быть сохранены в локальной базе данных, чтобы быть доступными для работы в офлайн-режиме.

Архитектура приложения

Использовать архитектуру MVVM (Model-View-ViewModel) для разделения бизнес-логики и пользовательского интерфейса. Компонент **ViewModel** должен выполнять следующие функции:

- Выполнение запросов к серверу через библиотеку **Retrofit**.
- Сохранение данных в локальной базе данных с использованием **Room**.
- Передача данных в интерфейс приложения.

Интерфейс пользователя

Использовать библиотеку **Jetpack Compose** для создания пользовательского интерфейса.

При отсутствии интернет-соединения необходимо отображать данные из локальной базы данных.

Задание 4

Задание на разработку клиента для REST API сервера

Необходимо разработать клиент для взаимодействия с сервером, предоставляющим REST API для получения данных об аниме. Клиент должен выполнять следующие задачи:

Подключение к серверу

Для выполнения запросов к API необходимо использовать библиотеку `Retrofit`. Пример запроса:

```
GET http://10.0.2.2:9080/search?genre=action
```

Ответ от сервера возвращается в формате JSON и содержит данные об аниме:

```
[
  {
    "id": "1",
    "title": "Naruto",
    "genre": "Action, Adventure",
    "episodes": 220
  }
]
```

Локальное хранение данных

Для хранения данных об аниме необходимо использовать библиотеку `Room`. После получения данных с сервера они должны быть сохранены в локальной базе данных, чтобы быть доступными для работы в офлайн-режиме.

Архитектура приложения

Использовать архитектуру MVVM (Model-View-ViewModel) для разделения бизнес-логики и пользовательского интерфейса. Компонент `ViewModel` должен выполнять следующие функции:

- Выполнение запросов к серверу через библиотеку **Retrofit**.
- Сохранение данных в локальной базе данных с использованием **Room**.
- Передача данных в интерфейс приложения.

Интерфейс пользователя

Использовать библиотеку **Jetpack Compose** для создания пользовательского интерфейса.

При отсутствии интернет-соединения необходимо отображать данные из локальной базы данных.

Задание 5

Задание на разработку клиента для REST API сервера

Необходимо разработать клиент для взаимодействия с сервером, предоставляющим REST API для получения данных об аниме. Клиент должен выполнять следующие задачи:

Подключение к серверу

Для выполнения запросов к API необходимо использовать библиотеку **Retrofit**. Пример запроса:

GET `http://10.0.2.2:9080/search?minEpisodes=20&maxEpisodes=50`

Ответ от сервера возвращается в формате JSON и содержит данные об аниме:

```
[
  {
    "id": "1",
    "title": "Naruto",
    "genre": "Action, Adventure",
    "episodes": 220
  }
]
```

Локальное хранение данных

Для хранения данных об аниме необходимо использовать библиотеку **Room**. После получения данных с сервера они должны быть сохранены в локальной базе данных, чтобы быть доступными для работы в офлайн-режиме.

Архитектура приложения

Использовать архитектуру MVVM (Model-View-ViewModel) для разделения бизнес-логики и пользовательского интерфейса. Компонент **ViewModel** должен выполнять следующие функции:

- Выполнение запросов к серверу через библиотеку **Retrofit**.
- Сохранение данных в локальной базе данных с использованием **Room**.
- Передача данных в интерфейс приложения.

Интерфейс пользователя

Использовать библиотеку **Jetpack Compose** для создания пользовательского интерфейса.

При отсутствии интернет-соединения необходимо отображать данные из локальной базы данных.

Задание 6

Задание на разработку Android-клиента

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для поиска информации об аниме. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу **http://10.0.2.2:9080/search**. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/search?title=Naruto
```

Ответ от сервера будет содержать данные об аниме в формате JSON:

```
[ { "id": "1", "title": "Naruto",  
  "genre": "Action, Adventure", "episodes": 220 } ]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и **UI**. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в **UI**.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен предоставлять следующие возможности:

Поле для ввода текста, позволяющее пользователю ввести название аниме для поиска. Список аниме, соответствующих запросу, отображаемый с названием, жанром и количеством эпизодов. Сообщение об отсутствии результатов, если аниме не найдено.

Функциональные требования

При подключении к интернету данные должны обновляться и отображаться в реальном времени. В случае отсутствия интернет-соединения данные должны браться из локальной базы данных. Пользовательский ввод должен быть учтён в реальном времени (по мере ввода текста в поле поиска).

Рекомендации

Использовать аннотации **@Query** в **Retrofit** для передачи параметров запроса. Настроить обработку ошибок: в случае недоступности сервера отображать соответствующее сообщение. Соблюдать принципы архитектуры **MVVM** и использовать **Retrofit**, **Room**, и **Jetpack Compose**.

Задание 7

Задание на разработку Android-клиента для поиска туристических мест

Необходимо разработать Android-клиент для взаимодействия с сервером, предоставляющим REST API для поиска туристических мест. Сервер доступен по адресу `http://10.0.2.2:9080/search`. Приложение должно выполнять следующие задачи:

Подключение к серверу

Клиент должен отправлять запросы к API сервера для поиска мест по имени. Используйте библиотеку `Retrofit` для выполнения запросов. Пример запроса:

```
GET http://10.0.2.2:9080/search?name=eiffel
```

Ответ от сервера будет содержать данные в формате JSON:

```
[ { "id": "1", "name": "Eiffel Tower",  
  "country": "France", "description": "Iconic iron lattice tower in Paris.",  
  "popularity": 98 } ]
```

Локальное хранение данных

Используйте библиотеку `Room` для сохранения данных о туристических местах в локальную базу данных. Это позволит отображать сохранённые результаты в офлайн-режиме.

Архитектура приложения

Организуйте архитектуру приложения по шаблону MVVM:

`ViewModel` отвечает за выполнение запросов через `Retrofit` и работу с `Room`. UI получает данные из `ViewModel`.

Интерфейс пользователя

Создайте интерфейс с помощью `Jetpack Compose`, который включает:

Поле для ввода имени места для поиска. Кнопку "Найти". Список найденных мест, отображающий имя, страну и описание.

Функциональные требования

При вводе имени места приложение отправляет запрос к серверу и отображает список найденных мест. Результаты поиска автоматически сохраняются в локальной базе данных. Если подключение к серверу недоступно, приложение отображает данные из локальной базы данных. Реализуйте возможность обновления данных при наличии интернет-соединения.

Рекомендации

Для обработки запросов используйте корутины (Kotlin Coroutines). Реализуйте обработку ошибок, таких как отсутствие интернета или недоступность сервера. Предусмотрите автоматическое обновление данных в случае изменения состояния сети.

Задание 8

Задание на разработку Android-клиента для туристического сервера

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для получения информации о туристических местах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/searchByCountry`. Используйте библиотеку `Retrofit` для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/searchByCountry?country=France
```

Ответ от сервера будет содержать данные о туристических местах в формате JSON:

```
{ "places": [ { "id": "1", "name": "Eiffel Tower",  
  "country": "France", "description": "Iconic iron lattice tower in Paris.",  
  "popularity": 98 } ] }
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и **UI**. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в **UI**.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список туристических мест, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

При подключении к интернету данные должны обновляться. В случае отсутствия интернета — отображать данные из локальной базы данных. Соблюдать принципы архитектуры **MVVM** и использовать **Retrofit** и **Room** для работы с данными. Приложение должно отображать название, страну, описание и рейтинг популярности для каждого туристического места.

Задание 9

Задание на разработку Android-клиента для туристического сервера

Необходимо разработать Android-клиент для работы с сервером, предоставляющим **REST API** для получения информации о туристических местах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу <http://10.0.2.2:9080/searchByPopularity>. Используйте библиотеку **Retrofit** для выполнения запросов к **API**. Пример запроса:

GET `http://10.0.2.2:9080/searchByPopularity?min=80&max=95`

Ответ от сервера будет содержать данные о туристических местах в формате JSON:

```
[
  {
    "id": "1",
    "name": "Eiffel Tower",
    "country": "France",
    "description": "Iconic iron lattice tower in Paris.",
    "popularity": 98
  }
]
```

Локальное хранение данных

Для хранения результатов запросов используйте `Room`. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру `MVVM` для разделения бизнес-логики и `UI`. `ViewModel` будет отвечать за выполнение запросов через `Retrofit`, сохранение данных в `Room` и их передачу в `UI`.

Интерфейс пользователя

Используйте `Jetpack Compose` для создания интерфейса. Интерфейс должен отображать список туристических мест, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться. - В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры `MVVM` и использовать `Retrofit` и `Room` для работы с данными. - Приложение должно отображать название, страну, описание и рейтинг популярности для каждого туристического места.

Задание 10

Задание на разработку мобильного клиента для поиска книг по году издания

Необходимо разработать мобильный клиент для работы с сервером, предоставляющим API для поиска книг по диапазону годов издания. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://localhost:9090/search`. Для выполнения запросов к API использовать стандартные HTTP-запросы.

Пример запроса:

```
GET http://localhost:9090/search?start=1900&end=2000
```

Ответ от сервера будет содержать список книг в формате JSON:

```
[ { "id": "1", "title": "1984", "author": "George Orwell", "year": 1949 } ]
```

Локальное хранение данных

Для хранения результатов запросов необходимо использовать SQLite или другую подходящую локальную базу данных. После получения данных с сервера, их нужно сохранять в базе данных для дальнейшей работы в офлайн-режиме.

Архитектура приложения

Рекомендуется использовать архитектуру MVVM для разделения бизнес-логики и интерфейса пользователя. ViewModel будет отвечать за выполнение запросов к API, обработку данных и передачу их в UI.

Интерфейс пользователя

Интерфейс должен отображать список книг, полученных с сервера или из локальной базы данных, если нет интернет-соединения. Для отображения данных использовать стандартные компоненты интерфейса мобильной платформы.

Рекомендации

При подключении к интернету данные должны обновляться. В случае отсутствия интернета отображать данные из локальной базы данных. Соблюдать принципы архитектуры MVVM. Использовать библиотеки Retrofit и Room.

Задание 11

Задание на разработку мобильного клиента

Необходимо разработать мобильный клиент для работы с сервером, предоставляющим API для поиска книг по названию. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://localhost:8080/books/search`. Используйте стандартные HTTP-запросы для выполнения запросов к API. Пример запроса:

```
GET http://localhost:8080/books/search?title=1984
```

Ответ от сервера будет содержать данные о книгах в формате JSON:

```
[ { "id": "1", "title": "1984", "author": "George Orwell", "year": 1949 } ]
```

Локальное хранение данных

Для хранения результатов запросов используйте SQLite или другую подходящую локальную базу данных. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру MVVM для разделения бизнес-логики и UI. ViewModel будет отвечать за выполнение запросов к API, сохранение данных в базу данных и их передачу в UI.

Интерфейс пользователя

Используйте нативные компоненты интерфейса для создания интерфейса. Интерфейс должен отображать список книг, полученных с сервера или из локальной базы данных в случае отсутствия интернет-соединения.

Рекомендации

При подключении к интернету данные должны обновляться. В случае отсутствия интернета — отображать данные из локальной базы данных. Соблюдать принципы архитектуры MVVM и использовать подходящие библиотеки для HTTP-запросов и локального хранения данных.

Задание 12

Задание на разработку Android-клиента

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для поиска книг по описанию. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:8080/books/search`. Используйте библиотеку `Retrofit` для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:8080/books/search?description=love
```

Ответ от сервера будет содержать данные о книгах в формате JSON:

```
[
  {
    "id": "1",
    "title": "1984",
    "author": "George Orwell",
    "year": 1949,
    "description": "Dystopian novel set in a totalitarian society."
  },
  {
    "id": "2",
```

```
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "year": 1960,
        "description": "A novel about racial injustice in the Deep South."
    }
]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и **UI**. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в **UI**.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список книг, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться. - В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры **MVVM** и использовать **Retrofit**, **Room** и **Jetpack Compose**.

Задание 13

Задание на разработку Android-клиента для работы с точками интереса

Необходимо разработать Android-клиент для работы с сервером, предоставляющим **REST API** для получения информации о точках интереса. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:8080/points-of-interest`. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:8080/points-of-interest?category=Landmark
```

Ответ от сервера будет содержать данные о точках интереса в формате JSON:

```
[ { "id": "1", "name": "Eiffel Tower",  
  "description": "Iconic symbol of Paris", "category": "Landmark",  
  "latitude": 48.8584, "longitude": 2.2945 } ]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных, чтобы обеспечить возможность работы в офлайн-режиме. Кэшированные данные должны быть доступны при отсутствии интернета.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и UI. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в UI.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список точек интереса, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры **MVVM** и использовать **Retrofit**, **Room** и **Jetpack Compose**.

Задание 14

Задание на разработку Android-клиента для работы с сервером достопримечательностей

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для получения информации о достопримечательностях. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:8080/locations`. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:8080/locations?category=Historical&latitude=40.7128&longitu
```

Ответ от сервера будет содержать данные о достопримечательностях в формате JSON:

```
[ { "id": "1", "name": "Statue of Liberty", "description": "Famous American mon  
    "category": "Landmark", "latitude": 40.6892, "longitude": -74.0445 } ]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и UI. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в UI.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список достопримечательностей, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

При подключении к интернету данные должны обновляться. В случае отсутствия интернета — отображать данные из локальной базы данных. Соблюдать принципы архитектуры MVVM и использовать `Retrofit`, `Room` и `Jetpack Compose`.

Задание 15

Задание на разработку Android-клиента

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для получения информации о достопримечательностях. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/locations`. Используйте библиотеку `Retrofit` для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/locations?category=Natural
```

Ответ от сервера будет содержать данные о достопримечательностях в формате JSON:

```
[
  {
    "id": "1",
    "name": "Eiffel Tower",
    "description": "Iconic symbol of Paris",
    "category": "Landmark",
    "latitude": 48.8584,
    "longitude": 2.2945
  }
]
```

Локальное хранение данных

Для хранения результатов запросов используйте `Room`. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру MVVM для разделения бизнес-логики и UI. ViewModel будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в UI.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список достопримечательностей, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться. - В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры MVVM и использовать **Retrofit**, **Room** и **Jetpack Compose**.

Задание 16

Задание на разработку Android-клиента для работы с сервером поиска ресторанов

Разработать Android-клиент для взаимодействия с сервером, предоставляющим API для поиска ресторанов по названию.

Основные требования

- Сервер доступен по адресу `http://10.0.2.2:9080/search`.
- Использовать **Retrofit** для выполнения HTTP-запросов. Пример запроса:

```
GET http://10.0.2.2:9080/search?name=sushi
```

Ответ возвращается в формате JSON и содержит список ресторанов с полями `id`, `name`, `cuisine`, `location`, `is_open`.

- Если рестораны не найдены, сервер возвращает ошибку `404 Not Found`.

Хранение данных

Реализовать локальное кэширование данных с помощью Room для обеспечения офлайн-режима.

Архитектура

Использовать архитектуру MVVM для разделения ответственности:

- `ViewModel` управляет данными и выполняет запросы к серверу.
- `Repository` реализует логику взаимодействия с API и локальной базой данных.
- Пользовательский интерфейс разработан на базе Jetpack Compose.

Интерфейс

Реализовать экран с поисковой строкой для ввода части названия ресторана и отображения результатов в виде списка. Каждый элемент списка должен содержать информацию о названии, типе кухни, местоположении и статусе (`is_open`).

Дополнительные требования

- Реализовать обработку ошибок (например, отсутствие результатов или проблемы с подключением к серверу).
- Обновлять локальные данные при наличии подключения к интернету.
- Обеспечить поддержку тёмной темы интерфейса.

Задание 17

Задание на разработку Android-клиента для работы с сервером ресторанов

Разработать Android-клиент для взаимодействия с сервером, предоставляющим REST API для поиска ресторанов по типу кухни.

Основные требования

- Сервер доступен по адресу `http://10.0.2.2:9080/restaurants`.
- Для выполнения запросов используйте `Retrofit`. Пример запроса:

`GET http://10.0.2.2:9080/restaurants?cuisine=Japanese`

Ответ в формате JSON содержит список ресторанов с полями `id`, `name`, `cuisine`, `city` и `is_open`.

Хранение данных

Сохранять данные запросов в локальной базе с помощью `Room` для работы в офлайн-режиме.

Архитектура

Использовать архитектуру `MVVM` для разделения логики и интерфейса:

- `ViewModel` отвечает за запросы к API и управление данными.
- `Repository` для взаимодействия с сервером и базой данных.
- UI на основе `Jetpack Compose`.

Интерфейс

Реализовать экран для ввода типа кухни и отображения списка ресторанов, с поддержкой офлайн-режима.

Рекомендации

- Обновлять данные при подключении к интернету.
- Реализовать обработку ошибок (например, отсутствие параметра `cuisine`).

Задание 18

Задание на разработку Android-клиента для работы с сервером ресторанов

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для получения информации о ресторанах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/restaurants`. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/restaurants?city=New York
```

Ответ от сервера будет содержать данные о ресторанах в формате JSON:

```
[
  {
    "id": "1",
    "name": "Pasta House",
    "cuisine": "Italian",
    "city": "New York",
    "is_open": true
  }
]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных, чтобы обеспечить возможность работы в офлайн-режиме. Кэшированные данные должны быть доступны при отсутствии интернета.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и UI. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в UI.

Интерфейс пользователя

Используйте `Jetpack Compose` для создания интерфейса. Интерфейс должен отображать список ресторанов, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует. Рестораны должны быть отображены с информацией о названии, кухне, городе и статусе открытия.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры MVVM и использовать `Retrofit`, `Room` и `Jetpack Compose`.
- Реализовать обработку ошибок, например, при отсутствии параметра города в запросе.

Задание 19

Задание на разработку Android-клиента для работы с сервером квартир

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для получения информации о квартирах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/apartments`. Используйте библиотеку `Retrofit` для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/apartments?min_price=300000&max_price=600000
```

Ответ от сервера будет содержать данные о квартирах в формате JSON:

```
[
  {
    "id": "1",
    "location": "New York",
    "price": 500000,
    "area": 70,
    "floor": 10,
    "rooms": 2,
    "year_built": 2000
  }
]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных, чтобы обеспечить возможность работы в офлайн-режиме. Кэшированные данные должны быть доступны при отсутствии интернета.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и **UI**. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в **UI**.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список квартир, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры **MVVM** и использовать **Retrofit**, **Room** и **Jetpack Compose**.

Задание 20

Задание на разработку Android-клиента для работы с сервером недвижимости с фильтрацией по году постройки

Необходимо разработать Android-клиент для работы с сервером, который предоставляет REST API для получения информации о квартирах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Сервер доступен по адресу `http://10.0.2.2:9080/apartments`, и предоставляет данные о квартирах с параметрами для фильтрации по году постройки. Клиент должен использовать библиотеку `Retrofit` для выполнения запросов к серверу.

Пример запроса:

```
GET http://10.0.2.2:9080/apartments?min_year=2000&max_year=2020
```

Ответ от сервера будет содержать данные о квартирах в формате JSON, например:

```
[
  {
    "id": "1",
    "location": "New York",
    "price": 500000,
    "area": 70,
    "floor": 10,
    "rooms": 2,
    "year_built": 2000
  },
  ...
]
```

Локальное хранение данных

Для хранения результатов запросов используйте `Room`. После получения данных с сервера, они должны сохраняться в локальной базе данных,

чтобы обеспечить возможность работы в офлайн-режиме. При отсутствии интернет-соединения, клиент должен отображать данные, сохранённые локально.

Архитектура приложения

Используйте архитектуру MVVM для разделения бизнес-логики и UI. `ViewModel` будет отвечать за выполнение запросов через `Retrofit`, сохранение данных в `Room` и их передачу в UI.

Интерфейс пользователя

Используйте `Jetpack Compose` для создания интерфейса. Интерфейс должен отображать список квартир, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует. Также, пользователь должен иметь возможность вводить минимальный и максимальный год постройки через текстовые поля в интерфейсе.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры MVVM и использовать `Retrofit`, `Room` и `Jetpack Compose`.
- В запросах должны передаваться параметры `min_year` и `max_year`.
- При отсутствии одного из параметров или наличии некорректных значений отобразить ошибку.
- В случае отсутствия квартир, удовлетворяющих критериям, вывести сообщение о том, что подходящих вариантов не найдено.
- В случае ошибок при запросах, таких как неверные значения `min_year` или `max_year`, отображать соответствующие сообщения об ошибке.

Задание 21

Задание на разработку Android-клиента для работы с сервером недвижимости

Необходимо разработать Android-клиент для работы с сервером, который предоставляет REST API для получения информации о квартирах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Сервер доступен по адресу `http://10.0.2.2:9080/apartments`, и предоставляет данные о квартирах с параметрами для фильтрации по количеству комнат. Клиент должен использовать библиотеку `Retrofit` для выполнения запросов к серверу.

Пример запроса:

```
GET http://10.0.2.2:9080/apartments?min_rooms=2&max_rooms=3
```

Ответ от сервера будет содержать данные о квартирах в формате JSON, например:

```
[
  {
    "id": "1",
    "location": "New York",
    "price": 500000,
    "area": 70,
    "floor": 10,
    "rooms": 2,
    "year_built": 2000
  },
  ...
]
```

Локальное хранение данных

Для хранения результатов запросов используйте `Room`. После получения данных с сервера, они должны сохраняться в локальной базе данных, чтобы обеспечить возможность работы в офлайн-режиме. При отсутствии интернет-соединения, клиент должен отображать данные, сохранённые локально.

Архитектура приложения

Используйте архитектуру MVVM для разделения бизнес-логики и UI. `ViewModel` будет отвечать за выполнение запросов через `Retrofit`, сохранение данных в `Room` и их передачу в UI.

Интерфейс пользователя

Используйте `Jetpack Compose` для создания интерфейса. Интерфейс должен отображать список квартир, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует. Также, пользователь должен иметь возможность вводить минимальное и максимальное количество комнат через текстовые поля в интерфейсе.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры MVVM и использовать `Retrofit`, `Room` и `Jetpack Compose`.
- В запросах должны передаваться параметры `min_rooms` и `max_rooms`.
- При отсутствии одного из параметров или наличии некорректных значений отобразить ошибку.
- В случае отсутствия квартир, удовлетворяющих критериям, вывести сообщение о том, что подходящих вариантов не найдено.

Задание 22

Задание на разработку Android-клиента для работы с сервером недвижимости

Необходимо разработать Android-клиент для работы с сервером, который предоставляет REST API для получения информации о квартирах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/apartments`. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/apartments?min_area=50&max_area=100
```

Ответ от сервера будет содержать данные о квартирах в формате JSON, например:

```
[
  {
    "id": "1",
    "location": "New York",
    "price": 500000,
    "area": 70,
    "floor": 10,
    "rooms": 2,
    "year_built": 2000
  }
]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных, чтобы обеспечить возможность работы в офлайн-режиме. Важно, чтобы при отсутствии интернет-соединения, клиент показывал данные, сохранённые локально.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и UI. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в UI.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список квартир, полученных с сервера или из локальной

базы данных, если интернет-соединение отсутствует. Также, пользователь должен иметь возможность вводить минимальную и максимальную площадь квартир через текстовые поля в интерфейсе.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры MVVM и использовать Retrofit, Room и Jetpack Compose.
- В запросах должны передаваться параметры `min_area` и `max_area`.
- При неправильных или отсутствующих параметрах отобразить ошибку.

Задание 23

Задание на разработку клиента для REST API сервера

Необходимо разработать клиент для взаимодействия с сервером, предоставляющим REST API для получения данных о маршрутах автобусов. Клиент должен выполнять следующие задачи:

Подключение к серверу

Для выполнения запросов к API необходимо использовать библиотеку Retrofit. Пример запроса:

```
GET http://10.0.2.2:9080/routes?min_length=10&max_length=20
```

Ответ от сервера возвращается в формате JSON и содержит данные о маршрутах автобусов:

```
[
  {
    "id": "1",
    "name": "Route A",
```

```
        "length": 12.5,  
        "number_of_stops": 8  
    }  
]
```

Локальное хранение данных

Для хранения данных о маршрутах автобусов необходимо использовать библиотеку **Room**. После получения данных с сервера они должны быть сохранены в локальной базе данных, чтобы быть доступными для работы в офлайн-режиме.

Архитектура приложения

Использовать архитектуру **MVVM** (Model-View-ViewModel) для разделения бизнес-логики и пользовательского интерфейса. Компонент **ViewModel** должен выполнять следующие функции:

- Выполнение запросов к серверу через библиотеку **Retrofit**.
- Сохранение данных в локальной базе данных с использованием **Room**.
- Передача данных в интерфейс приложения.

Интерфейс пользователя

Использовать библиотеку **Jetpack Compose** для создания пользовательского интерфейса.

При отсутствии интернет-соединения необходимо отображать данные из локальной базы данных.

Задание 24

Задание на разработку Android-клиента

Необходимо разработать Android-клиент для работы с сервером, предоставляющим **REST API** для получения информации о продуктах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/products`. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/products?  
min_price=50&max_price=200
```

Ответ от сервера будет содержать данные о продуктах в формате JSON:

```
[  
  {  
    "id": "2",  
    "name": "Груша",  
    "price": 110.75,  
    "shelf_life": 25,  
    "is_organic": false  
  }  
]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных, чтобы обеспечить возможность работы в офлайн-режиме. Кэшированные данные должны быть доступны при отсутствии интернета.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и UI. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в UI.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список продуктов, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться.
- В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры MVVM и использовать **Retrofit**, **Room** и **Jetpack Compose**.

Задание 25

Задание на разработку Android-клиента

Необходимо разработать Android-клиент для работы с сервером, предоставляющим REST API для получения информации о продуктах. Клиент должен выполнять следующие задачи:

Подключение к серверу

Клиент должен взаимодействовать с сервером по адресу `http://10.0.2.2:9080/products`. Используйте библиотеку **Retrofit** для выполнения запросов к API. Пример запроса:

```
GET http://10.0.2.2:9080/products?  
min_shelf_life=2023-01-01&max_shelf_life=2024-01-01
```

Ответ от сервера будет содержать данные о продуктах в формате JSON:

```
[  
  {  
    "id": 1,  
    "name": "Молоко",  
    "category": "Молочные продукты",  
    "shelf_life": "2024-01-01",  
    "price": 100.50,  
    "is_organic": true  
  }  
]
```

Локальное хранение данных

Для хранения результатов запросов используйте **Room**. После получения данных с сервера они должны сохраняться в локальной базе данных для работы в офлайн-режиме.

Архитектура приложения

Используйте архитектуру **MVVM** для разделения бизнес-логики и **UI**. **ViewModel** будет отвечать за выполнение запросов через **Retrofit**, сохранение данных в **Room** и их передачу в **UI**.

Интерфейс пользователя

Используйте **Jetpack Compose** для создания интерфейса. Интерфейс должен отображать список продуктов, полученных с сервера или из локальной базы данных, если интернет-соединение отсутствует.

Рекомендации

- При подключении к интернету данные должны обновляться. - В случае отсутствия интернета — отображать данные из локальной базы данных.
- Соблюдать принципы архитектуры **MVVM** и использовать **Retrofit**, **Room** и **Jetpack Compose**.

Ссылки

<https://developer.android.com/>

<https://kotlinlang.ru/docs/kotlin-doc.html>