Minha Agenda Simples

Um projeto prático completo para aplicar conceitos fundamentais do desenvolvimento Android. Neste estudo de caso, você construirá um aplicativo de agenda de compromissos do zero, cobrindo desde a interface do usuário até o consumo de APIs externas.

O Contexto do Projeto

O Desafio

Manter o controle de compromissos e tarefas é essencial no dia a dia moderno. Você foi encarregado de desenvolver a primeira versão de um aplicativo Android simples para ajudar os usuários a visualizarem suas próximas tarefas ou compromissos de forma organizada e intuitiva.

Este projeto combina os conceitos fundamentais que você aprendeu nas aulas anteriores em uma aplicação real e funcional, preparando você para desafios mais complexos no futuro.

Objetivo Principal

Aplicar conhecimentos de construção de UI com XML, manipulação de eventos de clique, modelagem de dados usando programação orientada a objetos, navegação entre telas, arquitetura MVVM e consumo de APIs REST usando Retrofit.

O resultado final será um aplicativo completo que busca dados de uma fonte externa e permite aos usuários visualizar e gerenciar suas tarefas de forma eficiente.

Funcionalidades Essenciais

Tela Principal

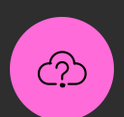
Lista rolável de tarefas usando RecyclerView

- Exibe descrição e data de cada tarefa
- Busca dados automaticamente da API
- Layout responsivo e organizado

Modelagem de Dados

Data class estruturada com Kotlin

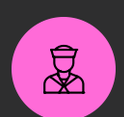
- ID único para cada tarefa
- Descrição detalhada
- Data de entrega
- Status de conclusão



API Simulada

Integração com Retrofit e Gson

- Chamadas assíncronas com Coroutines
- Conversão automática de JSON
- Tratamento de erros de rede



Navegação

Fluxo entre telas com Activities

- Clique em item abre detalhes
- Passagem de dados via Intent
- Experiência fluida do usuário

Tela Principal: Lista de Tarefas

A tela principal é o coração do aplicativo. Ao abrir o app, o usuário verá imediatamente uma lista rolável e organizada de todas as suas tarefas e compromissos. Esta implementação utiliza o poderoso componente RecyclerView, que oferece performance otimizada mesmo com grandes quantidades de dados.

Cada item na lista deve apresentar informações essenciais de forma clara: a descrição completa da tarefa e sua respectiva data de entrega ou realização. O layout de cada item será criado em um arquivo XML separado, promovendo a reutilização de código e facilitando a manutenção futura do projeto.

01

RecyclerView Setup

Configure o RecyclerView no layout principal com LayoutManager apropriado

02

Item Layout XML

Crie arquivo XML separado para o design de cada item da lista

03

Adapter Implementation

Implemente o Adapter e ViewHolder para vincular dados aos itens

04

Data Binding

Conecte a lista de tarefas ao RecyclerView através do Adapter

A implementação correta do RecyclerView é fundamental para o sucesso do aplicativo. Este componente gerencia eficientemente a criação e reciclagem de views, garantindo uma experiência suave mesmo em dispositivos com recursos limitados. Dedique tempo especial para entender o padrão ViewHolder e como ele otimiza o desempenho da lista.

Modelagem de Dados com Kotlin

Data Class: Tarefa

A modelagem correta dos dados é crucial para um aplicativo bem estruturado. Utilizando os recursos do Kotlin, você criará uma data class chamada `Tarefa` que encapsula todas as informações necessárias sobre cada compromisso.

Esta classe servirá como o modelo de dados fundamental do aplicativo, permitindo a conversão automática de JSON para objetos Kotlin através do Gson.

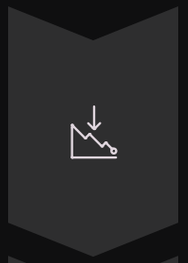
```
data class Tarefa(
    val id: Int,
    val descricao: String,
    val dataEntrega: String,
    val concluida: Boolean
)
```

Os principais campos incluem:

- **id:** Identificador único da tarefa
- **descricao:** Texto descritivo do compromisso
- **dataEntrega:** Data limite no formato ISO
- **concluida:** Status booleano de conclusão

As data classes do Kotlin oferecem funcionalidades automáticas extremamente úteis, como métodos equals(), hashCode(), toString() e copy(). Essas características tornam a manipulação de dados muito mais simples e segura, reduzindo significativamente a quantidade de código boilerplate necessário.

Consumo de API com Retrofit

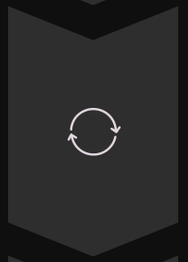


Configuração do Retrofit

Configure a instância do Retrofit com a URL base e o conversor Gson para transformar JSON em objetos Kotlin automaticamente

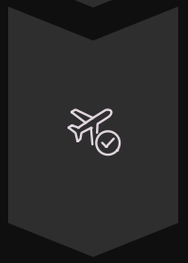
Interface de Serviço

Defina uma interface com os endpoints da API usando anotações do Retrofit para especificar o método HTTP e o caminho



Coroutines

Implemente chamadas assíncronas usando Coroutines para evitar bloquear a thread principal e garantir uma UI responsiva



Tratamento de Resposta

Processe a resposta da API, converta para List<Tarefa> e atualize a interface do usuário com os dados recebidos

O Retrofit é uma biblioteca poderosa que simplifica drasticamente o consumo de APIs REST em Android. Combinado com o Gson para conversão de JSON e Coroutines para operações assíncronas, você obtém uma solução elegante e eficiente para comunicação com servidores. A configuração inicial pode parecer complexa, mas uma vez estabelecida, adicionar novos endpoints se torna extremamente simples.

Estrutura da API e JSON

Endpoint de Exemplo

Para simular a busca de dados em produção, você utilizará uma API simulada que retorna um JSON com a lista de tarefas. Esta abordagem permite desenvolver e testar o aplicativo sem depender de um backend real inicialmente.

Endpoint: https://api.seudominio.com/minhas_tarefas

Substitua por um link real de um serviço de mock como Mocky.io ou JSONPlaceholder

Estrutura JSON Esperada


```
[
  {
    "id": 101,
    "descricao": "Preparar material da Aula 14",
    "dataEntrega": "2025-11-03",
    "concluida": false
  },
  {
    "id": 102,
    "descricao": "Revisar exercícios da Aula 13",
    "dataEntrega": "2025-10-28",
    "concluida": true
  },
  {
    "id": 103,
    "descricao": "Planejar reunião de projeto",
    "dataEntrega": "2025-11-05",
    "concluida": false
  }
]
```


Campos do JSON

- **id:** Número inteiro único
- **descricao:** String com texto da tarefa
- **dataEntrega:** String no formato YYYY-MM-DD
- **concluida:** Boolean indicando status

Cada objeto no array representa uma tarefa individual. O Gson converterá automaticamente estes objetos JSON em instâncias da data class Tarefa.

Arquitetura MVVM

A arquitetura Model-View-ViewModel (MVVM) é um padrão fundamental no desenvolvimento Android moderno. Ela promove a separação de responsabilidades, tornando o código mais testável, manutenível e escalável. Neste projeto, você implementará uma versão básica do MVVM para organizar adequadamente a lógica de negócio e a interface do usuário.

Model (Modelo)

Representa os dados do aplicativo através da data class Tarefa. Inclui também a lógica de acesso aos dados via Retrofit, encapsulando toda a comunicação com a API externa.

View (Visão)

Composta pelas Activities e layouts XML. Responsável apenas por exibir dados e capturar interações do usuário, sem conter lógica de negócio. Observa o ViewModel para atualizar a UI automaticamente.

ViewModel

Ponte entre Model e View. Busca dados da API, processa informações e as expõe através de LiveData. Sobrevive a mudanças de configuração, preservando o estado da UI durante rotações de tela.

Benefícios do MVVM

- Separação clara de responsabilidades
- Facilita testes unitários
- Código mais organizado e legível
- Reutilização de componentes
- Manutenção simplificada

LiveData e Observação

O LiveData é um componente chave do MVVM. Ele permite que a Activity observe mudanças nos dados do ViewModel e atualize automaticamente a interface quando novos dados chegam da API, sem acoplamento direto entre as camadas.

Navegação e Tela de Detalhes

Clique no Item

Usuário toca em uma tarefa na lista principal

1

Intent Criado

Activity cria Intent com dados da tarefa selecionada

2

Navegação

Sistema inicia nova Activity de detalhes

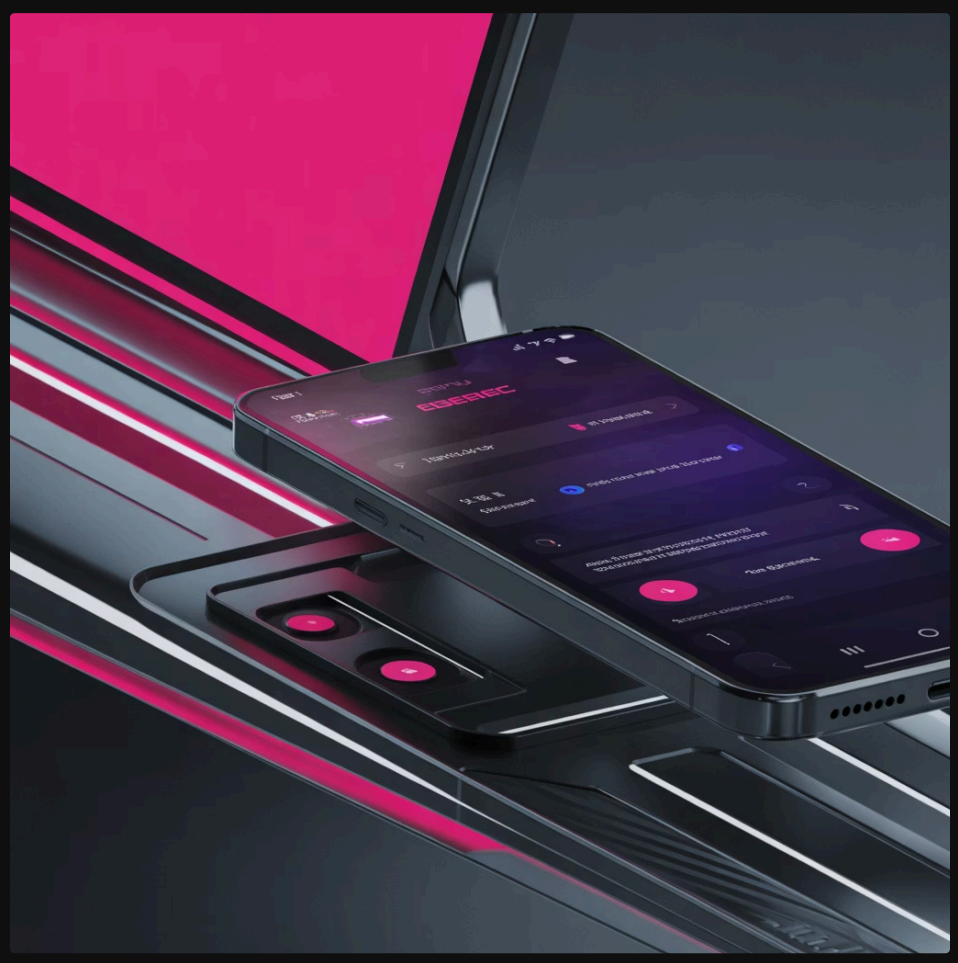
3

Exibição

Detalhes completos são mostrados ao usuário

4

A navegação entre telas é um aspecto fundamental da experiência do usuário em aplicativos Android. Quando o usuário clica em um item da lista na tela principal, o aplicativo deve navegar suavemente para uma segunda Activity que exibe todos os detalhes daquela tarefa específica.



Passagem de Dados

Utilize Intents para passar informações entre Activities. Você pode enviar o ID da tarefa ou, preferencialmente, serializar o objeto Tarefa completo usando Parcelable ou Serializable.

Tela de Detalhes

Esta tela deve receber os dados e exibir todas as informações: descrição completa, data de entrega formatada e status visual indicando se a tarefa está "Pendente" ou "Concluída".

A implementação correta da navegação garante uma experiência fluida e profissional. Considere adicionar animações de transição entre Activities para tornar o fluxo ainda mais agradável visualmente.

Requisitos e Avaliação

Requisitos Técnicos

Linguagem e Ambiente

Linguagem: Kotlin

IDE: Android Studio

Layouts: XML com ConstraintLayout

Componentes

Lista: RecyclerView com Adapter customizado

Rede: Retrofit + Gson + Coroutines

Navegação: Intent entre Activities

Arquitetura

Padrão: MVVM (Model-View-ViewModel)

Observação: ViewModel com LiveData

Separação: Camadas bem definidas

Critérios de Avaliação

Critério	Descrição
Estrutura do Projeto	Organização dos arquivos, uso de pacotes, nomenclatura consistente
Qualidade da UI	Layouts bem construídos, uso correto de ConstraintLayout e componentes
Modelagem de Dados	Data class corretamente definida com tipos apropriados
RecyclerView	Implementação completa de Adapter, ViewHolder e binding de dados
Arquitetura MVVM	Separação de responsabilidades, uso de ViewModel e LiveData
Consumo de API	Retrofit configurado, chamadas assíncronas, tratamento de dados
Funcionalidade	Aplicativo cumpre todos os requisitos essenciais descritos
Qualidade do Código	Legibilidade, comentários, boas práticas de programação

Este projeto representa uma oportunidade excepcional para consolidar os conhecimentos adquiridos nas aulas anteriores. Ao completá-lo com sucesso, você terá desenvolvido um aplicativo Android funcional e bem arquitetado, preparando-se para desafios mais complexos e projetos profissionais no futuro. Dedique-se, explore as possibilidades e aproveite o processo de aprendizado!