





**UNIP**  
**UNIVERSIDADE PAULISTA**

## **Roteiros**

**Desenvolvimento para Dispositivos Móveis**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Introdução ao Desenvolvimento Android</p>	<p><b>ROTEIRO 1</b></p>
---	--	-------------------------

## Roteiro da Aula Prática – Introdução ao Desenvolvimento Android

### 1. Objetivos da Aula

- Compreender a evolução dos dispositivos móveis e sua relevância na sociedade atual.
- Explorar as principais características e aplicações dos dispositivos móveis.
- Demonstrar, na prática, o impacto dos dispositivos móveis em diferentes setores.
- Desenvolver o primeiro projeto Android: **"Hello, Android!"**.

### 2. Recursos Necessários

- Computadores com **Android Studio** ou **App Inventor** (<https://appinventor.mit.edu/>) instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Dispositivos Android (opcional, para testes práticos).
- Material de apoio.

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Boas-vindas aos alunos e apresentação do tema e dos objetivos da aula.
- Discussão inicial: **"Como os dispositivos móveis impactam o nosso cotidiano?"**
- Exibição de imagens ou vídeo sobre a evolução dos dispositivos móveis.

### 3.2. Revisão Conceitual (20 minutos)

- Definição de dispositivos móveis e suas características principais.
- Principais sistemas operacionais móveis (Android, iOS, HarmonyOS).
- Aplicações práticas em setores como:
  - saúde (Apps de monitoramento);
  - educação (e-learning);
  - finanças (internet banking e fintechs).
- Discussão sobre a evolução do Android e sua importância no mercado global.

### 3.3. Demonstração Prática (30 minutos)

- Apresentação do ambiente **Android**:
  - Explicação do processo inicial de Sync do gradle e Build.
  - Interface do usuário.
  - Explicação das principais seções (Editor, Logcat, Console).
  - Mostrar a tela de código e Split.
- Criação do primeiro projeto: **"Hello, Android!"**.
  - Configuração do projeto Android Empty Activity.
  - Explicação da estrutura de diretórios (MainActivity, res, manifest.xml).
  - Execução do App no emulador Android.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Cada aluno deverá desenvolver um App básico que exiba:
  - O nome do aluno.
  - Um botão que, ao ser clicado, exiba uma mensagem personalizada (ex.: "Bem-vindo ao meu primeiro App!").

## Ampliação do Desafio

- Personalizar o design usando diferentes layouts e cores.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades enfrentadas pelos alunos durante a prática.
- Dicas para aprofundamento no desenvolvimento Android.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo teórico:** Explicação sobre a evolução dos dispositivos móveis e o impacto atual.
- **Código-fonte comentado:** Explicação do funcionamento de cada trecho de código implementado no projeto.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Criatividade na personalização do App.
	3,0	Funcionamento correto do aplicativo.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## **8. Conclusão**

Nesta aula, os alunos compreenderam a evolução dos dispositivos móveis e sua relevância na sociedade atual, explorando suas principais características e aplicações em setores como saúde, educação e finanças. No aspecto técnico, adquiriram conhecimentos iniciais sobre o ambiente Android, configurando e desenvolvendo seu primeiro aplicativo básico, o "Hello, World!". A prática permitiu o entendimento da estrutura de um App, com foco em atividades, layouts e interações simples. Além disso, os alunos desenvolveram habilidades em personalização de interface e usabilidade, preparando-se para aprofundar conceitos mais avançados, como programação orientada a objetos e o desenvolvimento de interfaces mais complexas nas próximas aulas.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Variáveis, Operadores e Estruturas Condicionais</p>	<p><b>ROTEIRO 2</b></p>
---	--	-------------------------

## Roteiro da Aula Prática – Variáveis

### 1. Objetivos da Aula

- Compreender o conceito de variáveis e seus tipos de dados.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android/Kotlin Playground**.
- Conexão com a internet.
- Material de apoio.

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula.
- Discussão inicial: **"Onde vemos lógica de programação no nosso cotidiano?"** (exemplos práticos, como assistentes virtuais, apps de calculadora etc.)
- Exposição breve sobre a importância de uma lógica bem estruturada no desenvolvimento de aplicativos.

### 3.2. Revisão Conceitual (20 minutos)

- Definição de **variáveis** e tipos de dados em Kotlin (Int, double, String, boolean, Array, List e Pair).
- Explicação dos **operadores aritméticos** (+, -, \*, /, %), **relacionais** (>, <, ==, !=) e **lógicos** (&&, ||, !).

### 3.3. Demonstração Prática (30 minutos)

- Conceito de variável
  - Demonstração da criação de String
  - Demonstração de criação de tipo numérico
  - Demonstração da criação de um Array
  - Demonstração de criação de um List (mutável)
  - Demonstração da criação de um Pair

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Criar uma lista com nomes de pessoas:
  - Inclusão de nomes na lista.
  - Remoção de nomes da lista.

### 5. Ampliação do Desafio

- Alterar as informações para conter um conjunto nome, idade.

### 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as dificuldades enfrentadas durante a atividade prática.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Definir e explicar os conceitos de variáveis, operadores e estruturas condicionais.
- **Código-Fonte Comentado:** Explicação de cada parte do código desenvolvido, destacando o uso das estruturas condicionais.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Criatividade na personalização do App.
	3,0	Uso adequado de operadores e estruturas condicionais.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.



**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos entenderam os conceitos fundamentais de variáveis, operadores e estruturas condicionais, aplicando-os na criação de aplicativos básicos no ambiente Android. Além de consolidar os princípios de lógica de programação, a prática possibilitou a construção de soluções que envolvem decisões automáticas no código, preparando os alunos para desafios mais complexos.

**Bom estudo e boa prática!**



  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Funções e Modularização de Código</p>	<p><b>ROTEIRO 3</b></p>
---	--	-------------------------

## Roteiro da Aula Prática – Funções e Modularização de Código

### 1. Objetivos da Aula

- Compreender o conceito de **funções** em programação e sua importância na modularização do código.
- Aprender a criar e chamar funções (usando o ambiente Android).
- Explorar o uso de **parâmetros** e **valores de retorno** para otimizar o funcionamento dos aplicativos.
- Explorar o uso de **Modelos de Funções de Extensão, Anônimas, Lambdas**.
- Desenvolver aplicativos mais organizados e fáceis de manter, por meio da reutilização de código.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android/Kotlin Playground**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e discussão sobre o papel das funções na programação.
- Reflexão: **"Por que é importante dividir um código em partes menores?"** (exemplos práticos no cotidiano: funções de um smartphone como chamada, envio de mensagens e execução de aplicativos).
- Breve explicação sobre os benefícios da modularização de código (reutilização, legibilidade e manutenção).

#### 3.2. Revisão Conceitual (20 minutos)

- Definição de **função** e sua estrutura básica em Java:
  - **Declaração:** Tipo de retorno, nome da função, parâmetros (se houver).
  - **Chamada:** Como e quando utilizar a função no código principal.
  - **Retorno:** O que a função devolve ao ser chamada.
- Explicação sobre as vantagens de usar funções em um projeto de desenvolvimento Android.
- Exemplos simples de funções (ex.: funções matemáticas e de exibição de mensagens).

#### 3.3. Demonstração Prática (30 minutos)

- Desenvolvimento de um ambiente básico com funções:
  - Criar uma função **sem parâmetros**.
  - Criar uma função **com parâmetros**.
  - Criar uma função **com parâmetros opcional**.
  - Criar uma função **com retorno**.
  - Criar uma função **de expressão única**.

- Uso do Operador **de segurança**.
- Uso do Operador **Elvis**.
- Criar funções **com assinatura**.
- Criar uma função **de extensão**.
- Criar uma função **Lambda e anônima**.
- Explicação detalhada do código, destacando o fluxo de chamada e retorno de funções.

#### 4. Atividade Prática (40 minutos)

##### Desafio Inicial

- Criar um aplicativo de **verifica a aprovação**:
  - Função que recebe duas notas como parâmetros e retorna se aprovado ou não.
  - Exibição de uma mensagem indicando se o aluno foi **aprovado** (média  $\geq 7$ ) ou **reprovado** (média  $< 7$ ).

##### Ampliação do Desafio

- Adicionar novas funcionalidades ao aplicativo:
  - Alterar a função para poder receber dois valores, inclusive nulo.
  - Em caso de duas notas iguais possibilitar a passagem de apenas um parâmetro.
  - Caso a nota passada seja nula, considerar o valor 5.

#### 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as dificuldades encontradas na implementação das funções.
- Dicas para escrever funções eficientes e evitar a duplicação de código.
- Orientação para a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação clara dos conceitos de função, parâmetros e valores de retorno.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido, com foco nas funções criadas e sua chamada no aplicativo.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta das funções e modularização do código.
	3,0	Uso adequado de parâmetros e valores de retorno.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos compreenderam a importância da modularização do código por meio da criação e chamada de funções no ambiente Android. A prática permitiu aplicar conceitos de parâmetros e valores de retorno em projetos reais, tornando o código mais organizado, reutilizável e fácil de manter.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Programação Estruturada, Sequência, Seleção e Repetição</p>	<p><b>ROTEIRO 4</b></p>
---	--	-------------------------

## Roteiro da Aula Prática – Programação estruturada

### 1. Objetivos da Aula

- Aplicar **estruturas condicionais** (if, else, else if, when) para tomada de decisões no código.
- Desenvolver algoritmos simples que envolvam cálculos e tomadas de decisão.
- Compreender o conceito e funcionamento das **estruturas de repetição**: for, while e do-while.
- Aprender quando e como utilizar cada tipo de laço de repetição em diferentes cenários.
- Aplicar as estruturas de repetição na criação de um aplicativo **Android** simples.
- Desenvolver algoritmos que automatizem tarefas repetitivas de forma eficiente.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android/Kotlin Playground**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula.
- Discussão inicial: **"Em quais situações do dia a dia usamos repetições?"** (exemplos práticos, como contagem de passos, listas de tarefas, cronômetros etc.)
- Introdução à importância das estruturas de repetição na automação de tarefas no desenvolvimento de aplicativos.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição das principais estruturas de seleção:
  - **if**: condicional simples;
  - **when**: Seleção múltipla.
- Definição das principais estruturas de repetição:
  - **for**: Quando o número de repetições é conhecido previamente;
  - **while**: Quando a condição de parada é avaliada antes da execução;
  - **do-while**: Quando a repetição deve ocorrer pelo menos uma vez antes da verificação da condição.
- Explicação sobre **inicialização, condição de parada e incremento/decremento** nos laços.
- Exemplos teóricos simples em Java para cada tipo de laço.

### 3.3. Demonstração Prática (30 minutos)

- Desenvolvimento de um aplicativo básico que:
  - Utilize a condicional **if** para mostrar se um número é par ou ímpar.
  - Utilize a condicional **when** para armazenar o texto do dia da semana na variável texto dia, definido por um inteiro dia (1-domingo, 7-sábado).
  - Utilize o laço **for** para exibir uma lista de países.
  - Utilize o laço **while** para exibir números pares até 20.
  - Utilize o laço **do-while** para repetir uma mensagem com nomes de países até o país ser "Alemanha".
- Explicação de cada parte do código e dos comportamentos observados no aplicativo.

### 4. Atividade Prática (40 minutos)

#### Desafio Inicial

- Criar um aplicativo que permita ao usuário inserir um número e, ao pressionar um botão, exiba a tabuada desse número (de 1 a 10) usando um laço **for**.

#### Ampliação do Desafio

- Adicionar as seguintes funcionalidades:
  - Usar um laço **while** para exibir apenas os números pares da tabuada.
  - Criar um botão que, ao ser pressionado, inicie um contador usando um laço **do-while**, que será interrompido por outro botão de parada.
  - Personalizar o layout do App para exibir as informações de maneira clara e organizada.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão das dificuldades enfrentadas pelos alunos durante o desenvolvimento do aplicativo.
- Dicas de boas práticas para o uso eficiente de laços de repetição.
- Orientação sobre a entrega do relatório final e revisão de conceitos que serão úteis em futuras aulas.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação conceitual dos laços de repetição (for, while, do-while).
- **Código-Fonte Comentado:** Explicação detalhada do código, destacando o funcionamento de cada estrutura de repetição utilizada.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Criatividade na personalização do App.
	3,0	Uso adequado das estruturas de repetição.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.



## **8. Conclusão**

Nesta aula, os alunos compreenderam os conceitos e aplicações das estruturas de repetição for, while e do-while, aplicando-os no desenvolvimento de um aplicativo Android funcional. Além de entender o comportamento de cada tipo de laço, os alunos desenvolveram habilidades práticas para resolver problemas com repetições automáticas e otimizar tarefas em seus códigos.

**Bom estudo e boa prática!**



Instituto de Ciências  
Exatas e Tecnologia

**Disciplina:** Desenvolvimento para  
Dispositivos Móveis

**Título da Aula:** Introdução à Programação  
Orientada a Objetos (POO)

**ROTEIRO 5**

## Roteiro da Aula Prática – Introdução à Programação Orientada a Objetos (POO)

### 1. Objetivos da Aula

- Compreender os conceitos de **classes**, **objetos** e **métodos** na Programação Orientada a Objetos (POO).
- Conceituação de **Herança**, **Polimorfismo**, **Encapsulamento**.
- Aplicar os conceitos de POO em um projeto **Android** simples.
- Desenvolver habilidades para estruturar um código mais organizado, reutilizável e modular.
- Criar um aplicativo funcional que utilize objetos e métodos para resolver problemas práticos.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android/Kotlin Playground**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de Programação Orientada a Objetos.
- Discussão inicial: **"Como objetos e classes aparecem no nosso cotidiano?"** (exemplos: carro, celular, conta bancária – cada um com atributos e comportamentos).
- Breve explicação sobre a importância da POO no desenvolvimento de aplicativos Android.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição de conceitos principais:
  - **Classe:** Modelo que define as características (atributos) e comportamentos (métodos) de um objeto.
  - **Objeto:** Instância de uma classe que contém dados específicos.
  - **Método:** Função definida dentro de uma classe, que realiza uma ação ou comportamento.
- Explicação sobre atributos (variáveis de instância) e métodos (ações).
- Instanciação.

#### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android/Kotlin**.
- Desenvolvimento de um aplicativo simples:
  - Criar uma **classe** chamada Aluno com os seguintes atributos: nome, curso, notaFinal.
  - Criar um método `exibirStatus()` que informe se o aluno foi **aprovado** ou **reprovado** com base na nota final (nota mínima de 7).
  - Instanciar objetos da classe Aluno e exibir os resultados em uma TextView.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Criar um aplicativo de **gestão de produtos**:
  - Definir uma **classe Produto** com os seguintes atributos: nome, preço, quantidade em estoque.
  - Criar um método `exibirInformacoes()` que mostre os detalhes do produto (nome, preço e estoque disponível).
  - Instanciar pelo menos três objetos de produtos diferentes e exibir as informações em uma lista.

### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Criar um método que **atualize o estoque** com base em uma venda simulada pelo usuário.
  - Implementar um método para calcular e exibir o **valor total em estoque** (preço x quantidade).
  - Exibir uma mensagem de alerta caso algum produto esteja com o estoque zerado (usando `AlertDialog`).

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas durante o desenvolvimento do projeto.
- Revisão dos conceitos de **classe**, **objeto** e **método** aplicados na prática.
- Dicas para modularizar o código de forma mais eficiente em projetos futuros.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação dos conceitos de classe, objeto e método com exemplos práticos.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido, destacando os atributos e métodos de cada classe.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta do aplicativo.
	3,0	Uso eficiente de objetos e modularização do código.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos compreenderam os conceitos centrais da Programação Orientada a Objetos (POO) aplicados ao desenvolvimento de aplicativos Android. A prática de criação de classes, objetos e métodos permitiu o desenvolvimento de projetos mais organizados, fáceis de manter e com melhor estrutura de código.

**Bom estudo e boa prática!**

  <b>Instituto de Ciências Exatas e Tecnologia</b>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Introdução ao Jetpack Compose e a Programação Declarativa</p>	<p><b>ROTEIRO 6</b></p>
---	--	-------------------------

## Roteiro da Aula Prática – Introdução ao Jetpack Compose e a Programação Declarativa

### 1. Objetivos da Aula

- Compreender a **técnica de desenvolvimento no ambiente Jetpack Compose** no desenvolvimento de aplicativos Android.
- Aprender a utilizar os principais componentes básicos do Material Design (Text, Image, Icon, Box, Surface).
- Aprender a utilizar os principais componentes de layout do Material Design (Column, Row, LazyColumn, LazyRow, Grid).
- Aprender a utilizar os principais componentes de interação do Material Design (Button, TextField, CheckBox, RadioButton).
- Criar widgets composables utilizando os componentes.
- Desenvolver um aplicativo simples que exiba mensagens visuais utilizando os componentes aprendidos.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **desenvolvimento de aplicativos com JetPack Compose**.
- Discussão inicial: **"Por que é importante entender a programação declarativa?"**
- Explicação de montagem de um widget.

#### 3.2. Revisão Conceitual (20 minutos)

- Explicação dos comandos executados na classe MainActivity.kt.
- Explicação do Widget Greeting.

#### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Activity**.
- Alteração do projeto inicial com:
  - Criar uma widget simples para ler o nome num TextField e, ao clicar no botão, mostrar na tela "Olá nome".

#### 4. Atividade Prática (40 minutos)

##### Desafio Inicial

- Criar um aplicativo que leia dois campos na tela e um botão:
  - Ao clicar no botão, realize a soma dos valores dos dois campos.

GreetingPreview

numero 1
numero 2
<div><div>+</div><div>Resultado:</div></div>

##### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Criar os botões de subtração, multiplicação, divisão e resto.

#### 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas durante a implementação.
- Orientação sobre a entrega do relatório final.

#### 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação do uso do JetPack Compose e Widgets.
- **Código-Fonte Comentado:** Explicação do código desenvolvido, destacando os métodos de ciclo de vida implementados.



## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta dos métodos de ciclo de vida.
	3,0	Uso adequado das transições entre os estados da Activity.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos entenderam o funcionamento do ciclo de vida das Activities em aplicativos Android. A prática proporcionou uma visão clara de como o sistema Android gerencia as Activities, permitindo um desenvolvimento mais eficiente e robusto.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Fundamentos dos Layouts</p>	<p><b>ROTEIRO 7</b></p>
---	--	-------------------------

## Roteiro da Aula Prática – Fundamentos dos Layouts

### 1. Objetivos da Aula

- Compreender os **Layouts no ambiente Jetpack Compose** no desenvolvimento de aplicativos Android.
- Compreender os conceitos de Composables, Layouts e Modificadores.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **Layouts no desenvolvimento de aplicativos com JetPack Compose**.
- Discussão inicial: **"Qual é a necessidade de Layouts flexíveis nos aplicativos?"**.
- Abertura do projeto iniciado no módulo anterior.

### 3.2. Revisão Conceitual (20 minutos)

- Revisão do estado atual do projeto.

### 3.3. Demonstração Prática (30 minutos)

- Demonstração de Column e Row utilizando textos.
- Demonstração de Box.
- Demonstração de LazyColumn.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Alterar o Layout para:

GreetingPreview

Calculadora	
numero 1	numero 2
<div><div>+</div><div>−</div><div>*</div><div>/</div><div>%</div></div>	
Resultado:	

### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Fazer o tratamento para evitar entradas nulas utilizando toDoubleOrNull e Elvis.
  - Fazer o tratamento da Divisão por zero.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas durante a implementação.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação do uso do JetPack Compose e Widgets.
- **Código-Fonte Comentado:** Explicação do código desenvolvido, destacando os métodos de ciclo de vida implementados.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta dos métodos de ciclo de vida.
	3,0	Uso adequado das transições entre os estados da Activity.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos entenderam o funcionamento do ciclo de vida das Activities em aplicativos Android. A prática proporcionou uma visão clara de como o sistema Android gerencia as Activities, permitindo um desenvolvimento mais eficiente e robusto.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Modificadores de Layout e de Interação</p>	<p><b>ROTEIRO 8</b></p>
---	---	-------------------------

## Roteiro da Aula Prática – Modificadores de Layout e de interação

### 1. Objetivos da Aula

- Compreender os **Modificadores ambiente Jetpack Compose** no desenvolvimento de aplicativos Android.
- Compreender como os Modificadores são utilizados para customizar a aparência, definir o comportamento, trabalhar com a acessibilidade, e gerenciar o layout.
- Compreender a estrutura básica de um modificador

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **Modificador**.
- Discussão inicial: **"Quais são as possibilidades de mudança da aparência e design de um componente?"**.
- Abertura do projeto iniciado no módulo anterior.

#### 3.2. Revisão Conceitual (20 minutos)

- Revisão do estado atual do projeto.

#### 3.3. Demonstração Prática (30 minutos)

- Um novo **Android Empty Activity**.
- Demonstração dos modificadores de layout: padding, fillMazSize, wrapContentSize, size.
- Demonstração no Box os modifcadores: offset, align e weight.
- Demonstração dos modificadores de Design: background/shape, border, clip, alpha, shadow.
- Explicação dos modificadores de Interação: clickable, toggleable, scrollable.

### 4. Atividade Prática (40 minutos)

#### Desafio Inicial

- Carregar a atividade da aula anterior (Calculadora).
- Alterar o componente TextField por OutlinedTextField.
- Alterar o layout para:

### Calculadora

numero 1

numero 2

+

-

\*

/

%

Resultado:



## Ampliação do Desafio

- Fazer as seguintes alterações:
  - O fundo do aplicativo na cor Cyan.
  - Título com letra amarela e fundo azul ocupando toda a largura.
  - Botões e o resultado com sombra.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas durante a implementação.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação do uso do JetPack Compose e Modifiers.
- **Código-Fonte Comentado:** Explicação do código desenvolvido, destacando os métodos de ciclo de vida implementados.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta dos métodos de ciclo de vida.
	3,0	Uso adequado das transições entre os estados da Activity.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## **8. Conclusão**

Nesta aula, os alunos entenderam o funcionamento do ciclo de vida das Activities em aplicativos Android. A prática proporcionou uma visão clara de como o sistema Android gerencia as Activities, permitindo um desenvolvimento mais eficiente e robusto.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Parâmetros do text e gerenciamento de espaço e alinhamento</p>	<p><b>ROTEIRO 9</b></p>
---	---	-------------------------

## Roteiro da Aula Prática – Parâmetros do text e gerenciamento de espaço e alinhamento

### 1. Objetivos da Aula

- Conhecer os principais parâmetros do text: color, fontSize, fontWeight, fontStyle, letterSpacing, textAlign, overflow, maxLines, objeto TextStyle.
- Compreender **os gerenciamentos de espaço no ambiente Jetpack Compose** no desenvolvimento de aplicativos Android (verticalArrangement e horizontalArrangement).
- Expor os tipos mais comuns de implementação Start, End, Center, SpaceBetween, SpaceAround e SpaceEvenly.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de Arranjos.
- Discussão inicial: **"Como arranjar os componentes dentro dos espaços no Aplicativo?"**.
- Abertura do projeto iniciado no módulo anterior.

#### 3.2. Revisão Conceitual (20 minutos)

- Revisão do estado atual do projeto.

#### 3.3. Demonstração Prática (30 minutos)

- Um novo projeto vazio utilizando o widget Greeter.
- Demonstração parâmetros de text: color, textSize, textAlign, letterSpacing.

### 4. Atividade Prática (40 minutos)

#### Desafio Inicial

- Carregar a atividade da aula anterior (Calculadora).
- Alterar para obter:

GreetingPreview

Calculadora

numero 1

numero 2

+

-

\*

/

%

Resultado:

## Ampliação do Desafio

- Fazer as seguintes alterações:
  - Implementar o botão "C" vermelho para limpar os campos n1 e n2.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas durante a implementação.
- Orientação sobre a entrega do relatório final.


## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação do uso do JetPack Compose e Arranjos.
- **Código-Fonte Comentado:** Explicação do código desenvolvido, destacando os métodos de ciclo de vida implementados.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta dos métodos de ciclo de vida.
	3,0	Uso adequado das transições entre os estados da Activity.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.



**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## **8. Conclusão**

Nesta aula, os alunos entenderam o funcionamento do ciclo de vida das Activities em aplicativos Android. A prática proporcionou uma visão clara de como o sistema Android gerencia as Activities, permitindo um desenvolvimento mais eficiente e robusto.

**Bom estudo e boa prática!**





Instituto de Ciências  
Exatas e Tecnologia

**Disciplina:** Desenvolvimento para  
Dispositivos Móveis

**Título da Aula:** Ciclo de Vida  
de uma Activity no Android

**ROTEIRO 10**

## Roteiro da Aula Prática – Ciclo de Vida de uma Activity no Android

### 1. Objetivos da Aula

- Compreender o **ciclo de vida de uma Activity** no desenvolvimento de aplicativos Android.
- Entender o comportamento das principais fases do ciclo: **criação** (onCreate()), **pausa** (onPause()) e **destruição** (onDestroy()).
- Aplicar métodos de ciclo de vida para gerenciar corretamente os estados do aplicativo.
- Desenvolver um aplicativo simples que exiba mensagens visuais em cada fase do ciclo de vida.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).



### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **ciclo de vida de uma Activity**.
- Discussão inicial: **"Por que é importante entender o ciclo de vida de uma Activity?"** (exemplos: pausa de Apps ao receber uma chamada, destruição de atividades ao fechar o App).
- Explicação da importância do gerenciamento de estados no desenvolvimento de aplicativos Android.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição do **ciclo de vida da Activity** e suas principais fases:
  - **onCreate()**: Quando a Activity é criada pela primeira vez.
  - **onStart()**: Quando a Activity se torna visível ao usuário.
  - **onResume()**: Quando a Activity está em primeiro plano e o usuário pode interagir.
  - **onPause()**: Quando outra Activity entra em primeiro plano, mas a atual ainda está parcialmente visível.
  - **onStop()**: Quando a Activity não está mais visível.
  - **onDestroy()**: Quando a Activity é destruída, seja pelo usuário ou pelo sistema.
- Apresentação de um fluxograma do ciclo de vida das Activities.

### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Activity**.
- Desenvolvimento de um aplicativo simples que:
  - Exiba uma mensagem (usando Toast) em cada método de ciclo de vida (`onCreate()`, `onPause()`).
  - Adicione um registro de log (`Log.d()`) para monitorar as transições entre os estados no Logcat.
  - Mude para uma nova Activity com um botão e exiba mensagens quando as transições de Activity ocorrerem.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Criar um aplicativo que registre o tempo total em que o usuário interage com a Activity:
  - Iniciar a contagem de tempo em **`onResume()`**.
  - Pausar a contagem em **`onPause()`**.
  - Exibir o tempo total de interação ao destruir a Activity (no método **`onDestroy()`**).

### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Salvar o tempo de interação usando **`SharedPreferences`**, mesmo que o aplicativo seja fechado.
  - Exibir um Toast personalizado em cada evento do ciclo de vida, informando o estado atual da Activity.
  - Criar uma nova tela (Activity) que receba o tempo total e exiba um resumo da interação.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas durante a implementação.
- Dicas sobre como evitar **vazamentos de memória** e problemas comuns no ciclo de vida das Activities.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação detalhada das fases do ciclo de vida da Activity e suas funções.
- **Código-Fonte Comentado:** Explicação do código desenvolvido, destacando os métodos de ciclo de vida implementados.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta dos métodos de ciclo de vida.
	3,0	Uso adequado das transições entre os estados da Activity.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos entenderam o funcionamento do ciclo de vida das Activities em aplicativos Android. A prática proporcionou uma visão clara de como o sistema Android gerencia as Activities, permitindo um desenvolvimento mais eficiente e robusto.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> Intents e Comunicação entre Activities</p>	<p><b>ROTEIRO 11</b></p>
---	---	--------------------------

## Roteiro da Aula Prática – Intents e Comunicação entre Activities

### 1. Objetivos da Aula

- Compreender o conceito de **Intents** e sua função na comunicação entre **Activities** no Android.
- Aprender a implementar a **passagem de dados** entre diferentes telas de um aplicativo.
- Aplicar os conceitos de **Intent explícita** e **Intent implícita** em projetos práticos.
- Desenvolver um aplicativo simples que permita a troca de informações entre Activities.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **Intents** no Android.
- Discussão inicial: **"Como os aplicativos se comunicam entre diferentes telas?"** (exemplos de troca de dados em Apps de redes sociais, e-commerce etc.).
- Explicação sobre a importância das Intents na navegação e troca de dados entre Activities.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição de **Intent**: Objeto responsável por iniciar outra Activity ou serviço.
- Tipos de Intents:
  - **Explícita**: Quando se conhece a classe da Activity de destino.
  - **Implícita**: Quando o sistema decide qual componente deve lidar com o Intent.
- Passagem de dados entre Activities com métodos:
  - **putExtra()**: Para enviar dados.
  - **getIntent().getStringExtra()**: Para recuperar os dados na Activity de destino.
- Exemplo simples:

java

```
// Enviando dados
val context = LocalContext.current
Column {
    Text("Tela principal")
    Button(onClick = {
        val intent = Intent(context, SegundaActivity::class.java)
        intent.putExtra("chave1", "valor 1")
        context.startActivity(intent)
    }) {
        Text("Ir para a segunda tela")
    }
}
```

```
// Recebendo dados
val valorRecebido = intent.getStringExtra("chave1")
Text( text = valorRecebido)
```

### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Activity**.
- Desenvolvimento de um aplicativo básico com duas telas (Activities):
  - Primeira tela (Activity principal): Campo de texto para o usuário inserir uma mensagem e um botão para enviar os dados.
  - Segunda tela: Recebe e exibe a mensagem enviada pela primeira tela em um text.
  - Implementação de uma **Intent explícita** para a troca de dados.
  - Utilização de um botão para retornar à tela inicial.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Criar um aplicativo de **cadastro de usuário**:
  - Primeira tela: Permitir que o usuário insira **nome, idade e e-mail**.
  - Segunda tela: Exibir as informações inseridas em um resumo formatado.
  - Utilizar uma **Intent explícita** para enviar os dados preenchidos.

### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Permitir a edição dos dados na segunda tela e enviá-los de volta para a primeira tela usando **startActivityResult()**.
  - Implementar uma **Intent implícita** para enviar um e-mail com as informações inseridas, usando um aplicativo de e-mail do dispositivo.
  - Adicionar validação de campos obrigatórios (ex.: nome e e-mail não podem estar vazios).

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades enfrentadas durante a implementação da comunicação entre Activities.
- Dicas sobre quando utilizar **Intents explícitas** ou **implícitas**.
- Orientação sobre a entrega do relatório final.



## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação dos conceitos de **Intents**, tipos de Intents e passagem de dados entre Activities.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido, com foco na lógica de envio e recebimento de dados.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta da passagem de dados.
	3,0	Uso adequado de <b>Intents explícitas</b> e <b>implícitas</b> .
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos compreenderam o funcionamento das Intents e sua aplicação na comunicação entre Activities em aplicativos Android. A prática demonstrou como passar dados entre diferentes telas e utilizar Intents explícitas e implícitas de forma eficiente.

**Bom estudo e boa prática!**



Instituto de Ciências  
Exatas e Tecnologia

**Disciplina:** Desenvolvimento para  
Dispositivos Móveis

**Título da Aula:** Introdução à  
Interface Gráfica (GUI) no Android

**ROTEIRO 12**

## Roteiro da Aula Prática – Introdução à Interface Gráfica (GUI) no Android

### 1. Objetivos da Aula

- Compreender os conceitos de **interface gráfica (GUI)** em aplicativos Android.
- Aprender a utilizar os principais tipos de **layouts: LinearLayout, RelativeLayout e ConstraintLayout**.
- Implementar diferentes tipos de layouts em um aplicativo simples e entender quando utilizar cada um.
- Desenvolver boas práticas de design responsivo para diferentes tamanhos de tela.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **Interface Gráfica (GUI)** no Android.
- Discussão inicial: **"Como o layout influencia a experiência do usuário em um aplicativo?"** (exemplos práticos como Apps de redes sociais, e-commerce etc.).
- Explicação da importância de criar interfaces organizadas e adaptáveis para diferentes dispositivos móveis.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição dos principais **layouts** no Android:
  - **LinearLayout:**
    - Organiza os elementos em uma única direção (vertical ou horizontal).
    - Exemplo de código:

xml

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button android:text="Botão 1" />
    <Button android:text="Botão 2" />
</LinearLayout>
```

- **RelativeLayout:**

- Posiciona os elementos em relação a outros componentes ou ao contêiner pai.
- Exemplo de código:

xml

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/botao1"
        android:text="Botão 1"
        android:layout_centerInParent="true" />

    <Button
        android:text="Botão 2"
        android:layout_below="@id/botao1"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

- **ConstraintLayout:**

- Permite a criação de layouts complexos com menos hierarquia de visualização.
- Exemplo de código:

xml

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/botao1"
        android:text="Botão 1"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Views Activity**.
- Desenvolvimento de um aplicativo simples utilizando os três tipos de layouts:
  - **Tela 1:** Interface usando **LinearLayout** para exibir uma lista de botões na vertical.
  - **Tela 2:** Interface com **RelativeLayout** posicionando os elementos em relação a outros widgets.
  - **Tela 3:** Interface com **ConstraintLayout** criando uma disposição mais flexível e responsiva.
- Explicação da diferença entre os layouts e quando usar cada um de acordo com a complexidade da interface.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Criar um aplicativo de **perfil de usuário**:
  - Utilizar **LinearLayout** para organizar as informações básicas (nome, e-mail, telefone).
  - Usar **RelativeLayout** para posicionar uma foto de perfil ao lado das informações.
  - Exibir um botão de "Editar Perfil" centralizado na parte inferior da tela.

### Ampliação do Desafio

- Melhorar a interface com as seguintes funcionalidades:
  - Usar **ConstraintLayout** para tornar o design mais flexível e responsivo para diferentes tamanhos de tela.
  - Adicionar campos de edição com **EditText** e um botão de salvar que atualize as informações exibidas na tela.
  - Implementar um recurso que mostre um alerta (AlertDialog) ao salvar as alterações com sucesso.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas na criação dos layouts.
- Dicas sobre boas práticas de design responsivo e acessibilidade.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação sobre os conceitos de **LinearLayout**, **RelativeLayout** e **ConstraintLayout**, destacando suas principais diferenças e usos.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido, destacando a estrutura dos layouts e suas propriedades.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta e funcional dos layouts.
	3,0	Usabilidade e responsividade do design.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos compreenderam os fundamentos da criação de **interfaces gráficas** usando os principais **layouts** do Android. A prática permitiu desenvolver habilidades em estruturar visualmente as aplicações e adaptar os layouts a diferentes tamanhos de tela.

**Bom estudo e boa prática!**



Instituto de Ciências  
Exatas e Tecnologia

**Disciplina:** Desenvolvimento para  
Dispositivos Móveis

**Título da Aula:** Widgets e Layouts no Android

**ROTEIRO 13**

## Roteiro da Aula Prática – Widgets e Layouts no Android

### 1. Objetivos da Aula

- Compreender os conceitos de **widgets** e **layouts** no Android.
- Aprender a utilizar **botões** (Button), **campos de texto** (EditText), e **TextView** no desenvolvimento de interfaces gráficas.
- Implementar a **manipulação de eventos** (cliques e entrada de texto) em um aplicativo Android.
- Desenvolver um aplicativo simples que responda às ações do usuário com interatividade e feedback visual.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).



### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **widgets** e **layouts**.
- Discussão inicial: **"Como os elementos de uma interface gráfica afetam a experiência do usuário?"** (exemplos: Apps de bancos, redes sociais, calculadoras).
- Explicação sobre a importância de criar interfaces intuitivas e responsivas.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição dos principais **widgets**:
  - **Button**: Botão que dispara uma ação ao ser clicado.
  - **EditText**: Campo de texto para inserção de dados pelo usuário.
  - **TextView**: Área de exibição de texto estático ou dinâmico.
- Explicação dos tipos de **layouts**:
  - **LinearLayout**: Organiza elementos de forma linear (vertical ou horizontal).
  - **RelativeLayout**: Posicionamento relativo entre os componentes.
  - **ConstraintLayout**: Layout avançado que permite o posicionamento flexível e eficiente.
- Manipulação de eventos de clique:
  - **setOnClickListener()** para capturar cliques em botões.
- Exemplo simples de manipulação de evento:

java

```
val botao = findViewById<Button>(R.id.meuBotao)
botao.setOnClickListener {
    Toast.makeText(this, "Botao clicado!", Toast.LENGTH_SHORT).show()
}
```

### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Views Activity**.
- Desenvolvimento de um aplicativo básico:
  - Layout usando **LinearLayout** (modo vertical).
  - Um **EditText** para entrada de texto, um **Button** para confirmar a ação e uma **TextView** para exibir a mensagem inserida pelo usuário.
  - Manipulação de evento: Exibir um Toast ao clicar no botão com o texto inserido.

### 4. Atividade Prática (40 minutos)

#### Desafio Inicial

- Criar um aplicativo de **calculadora simples**:
  - Campos de texto para inserir dois números.
  - Quatro botões: **Somar**, **Subtrair**, **Multiplicar**, **Dividir**.
  - Exibição do resultado em uma **TextView**.
  - Implementação de tratamento de exceções para evitar erros (ex.: divisão por zero).

#### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Limitar a entrada dos campos de texto a apenas números usando a propriedade `inputType="number"`.
  - Utilizar **RelativeLayout** ou **ConstraintLayout** para criar um design mais organizado e adaptável.
  - Implementar uma função que registre e exiba o **histórico de operações** em uma lista (**ListView** ou **RecyclerView**).

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas na implementação dos widgets e eventos.
- Dicas para melhorar a usabilidade da interface e as boas práticas no design de layouts.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação sobre os conceitos de **widgets**, **layouts** e manipulação de eventos.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido, com foco na lógica de interação entre os componentes.

## 7. Critérios de Avaliação



Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta das funcionalidades do aplicativo.
	3,0	Usabilidade e organização da interface gráfica.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## **8. Conclusão**

Nesta aula, os alunos entenderam como utilizar widgets e layouts no desenvolvimento de aplicativos Android. A prática envolveu a criação de interfaces interativas, manipulando eventos de clique e inserção de dados, proporcionando uma melhor experiência ao usuário.

**Bom estudo e boa prática!**

  <p><b>Instituto de Ciências Exatas e Tecnologia</b></p>	<p><b>Disciplina:</b> Desenvolvimento para Dispositivos Móveis</p> <p><b>Título da Aula:</b> ListView e Adapters – Criando Listagens Dinâmicas</p>	<p><b>ROTEIRO 14</b></p>
---	--	--------------------------

## Roteiro da Aula Prática – ListView e Adapters – Criando Listagens Dinâmicas

### 1. Objetivos da Aula

- Compreender o conceito de **ListView** e **Adapters** no Android para a criação de listagens dinâmicas.
- Aprender a implementar **listagens personalizadas** usando **Adapters** personalizados.
- Explorar a interação do usuário com itens da lista por meio de eventos de clique.
- Desenvolver um aplicativo que exiba e atualize uma lista de dados em tempo real.

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula e introdução ao conceito de **listas dinâmicas** no Android.
- Discussão inicial: **"Onde encontramos listas dinâmicas em aplicativos do dia a dia?"** (exemplos: lista de contatos, histórico de chamadas, feed de redes sociais).
- Explicação sobre a importância de trabalhar com dados dinâmicos e atualizáveis.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição de **ListView**: Um componente de interface usado para exibir uma lista de itens roláveis.
- Definição de **Adapter**:
  - **ArrayAdapter**: Usado para listas simples com dados diretos de arrays ou listas.
  - **Custom Adapter**: Permite a personalização do layout dos itens da lista.
- Eventos de clique em itens do ListView usando **setOnItemClickListener()**.
- Exemplo básico:

```
val listView = findViewById<ListView>(R.id.listView)
val nomes = arrayOf("João", "Maria", "Carlos")
val adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, nomes)
listView.adapter = adapter
listView.setOnItemClickListener { parent, view, position, id ->
    val itemSelecionado = nomes[position]
    Toast.makeText(applicationContext, "Você selecionou: $itemSelecionado",
    Toast.LENGTH_SHORT).show()
}
```

### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Views Activity**.
- Desenvolvimento de um aplicativo básico que:
  - Exiba uma lista de itens usando **ListView** com um **ArrayAdapter**.
  - Permita a interação com os itens da lista, exibindo uma mensagem personalizada usando um **Toast**.
  - Implemente um **Custom Adapter** para exibir um layout personalizado com ícones ao lado de cada item.

## 4. Atividade Prática (40 minutos)

### Desafio Inicial

- Criar um aplicativo de **lista de tarefas (To-Do List)**:
  - Exibir uma lista de tarefas utilizando um **ArrayAdapter**.
  - Permitir a adição de novas tarefas com um campo de texto e um botão.
  - Implementar um evento de clique que, ao clicar em uma tarefa, exiba uma mensagem de **"Tarefa concluída"**.

### Ampliação do Desafio

- Adicionar funcionalidades extras ao aplicativo:
  - Implementar um **Custom Adapter** que permita exibir uma marcação visual de tarefa concluída (ex.: texto riscado ou ícone de check).
  - Adicionar um botão de remoção de tarefa ao lado de cada item da lista.
  - Implementar um recurso de pesquisa (filtro) para buscar tarefas na lista.

## 5. Encerramento e Orientações Finais (20 minutos)

- Discussão sobre as principais dificuldades encontradas na implementação de listagens dinâmicas.
- Dicas sobre otimização de listas com muitos dados, usando **RecyclerView** em projetos mais avançados.
- Orientação sobre a entrega do relatório final.

## 6. Orientações para o Relatório Final


Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação sobre os conceitos de **ListView**, **Adapters** e eventos de interação.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido, destacando a criação e personalização da lista.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Implementação correta e funcional das listagens.
	3,0	Personalização e responsividade da interface gráfica
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.





**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## **8. Conclusão**

Nesta aula, os alunos compreenderam como criar e personalizar listagens dinâmicas usando ListView e Adapters no Android. A prática permitiu desenvolver habilidades em interações com o usuário, permitindo ações como adicionar, excluir e atualizar itens dinamicamente.

**Bom estudo e boa prática!**





Instituto de Ciências  
Exatas e Tecnologia

**Disciplina:** Desenvolvimento para  
Dispositivos Móveis

**Título da Aula:** Miniprojeto:  
Desenvolvendo um Aplicativo Completo

**ROTEIRO 15**

## Roteiro da Aula Prática – Miniprojeto: Desenvolvendo um Aplicativo Completo

### 1. Objetivos da Aula

- Construir um aplicativo completo utilizando a técnica do Views.
- Compreender o conceito de **API**.
- Conhecer novos componentes **Calendar** e **FAB**.
- Desenvolver um aplicativo para gerar um roteiro turístico utilizando API de inteligência. Artificial

### 2. Recursos Necessários

- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android** e acesso ao site: <https://ai.google.dev/gemini-api/docs/get-started/tutorial?lang=android&hl=pt-br>.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).

### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula conhecer o aplicativo a ser desenvolvido.
- Discussão inicial: **"Qual a importância das APIs e da IA?"**
- Explicação sobre a integração do App com a IA Gemini.

#### 3.2. Revisão Conceitual (20 minutos)

- Definição de **Calendar** e **DatePickerDialog**: Um componente de interface usado para exibir um Calendário e escolher uma data.
- Definição de **Spinner**: Versão prática do ListView em janela suspensa.
- Processo de Inclusão de um botão flutuante.
- Processo de obtenção da chave da API Gemini.

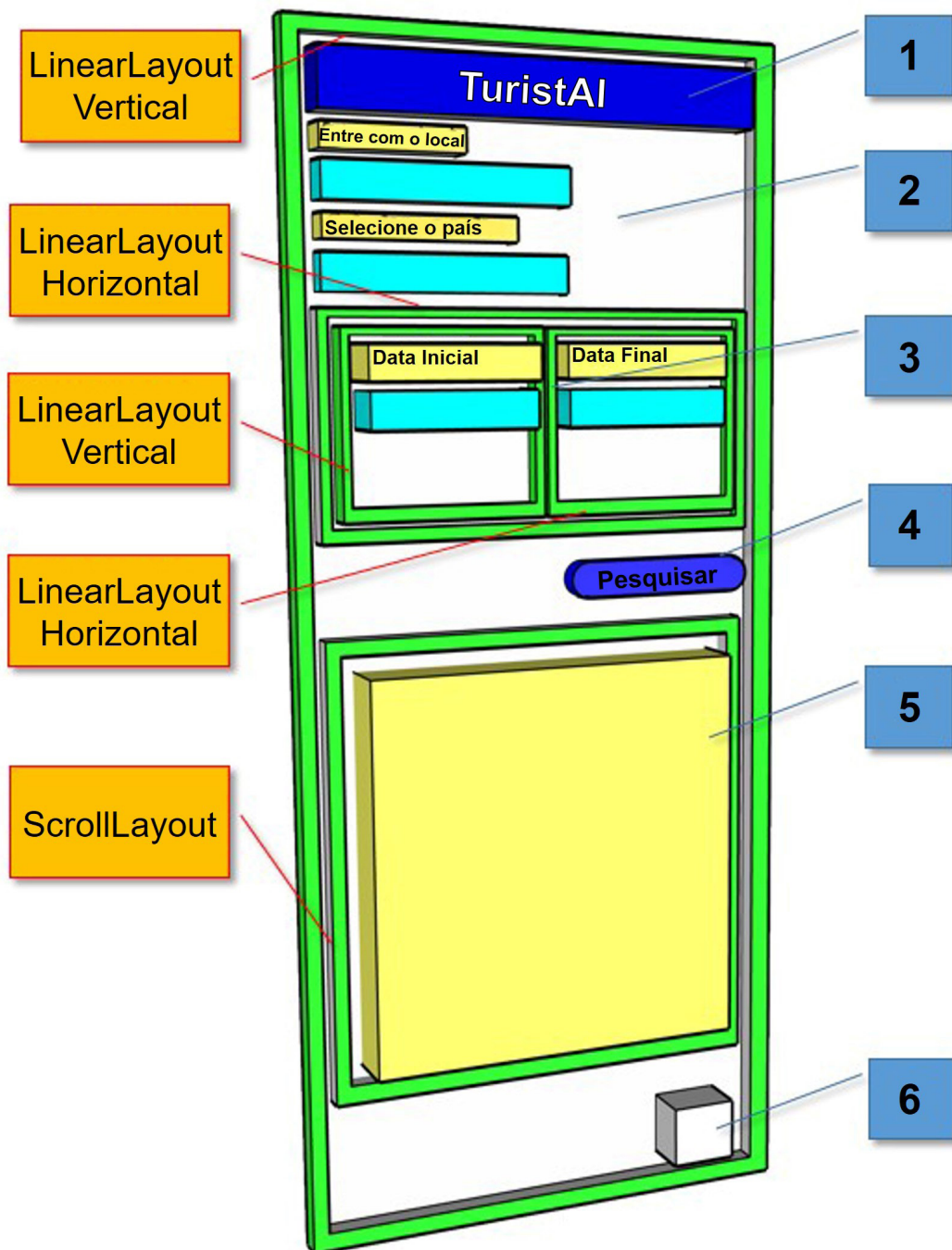
#### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto Android Empty Views Activity.
- No site <https://ai.google.dev/gemini-api/docs/get-started/tutorial?lang=android&hl=pt-br>, obter a chave da API.
- Adicionar a dependência do SDK ao seu projeto.
- Inicializar o modelo generativo utilizando a chave obtida.

### 4. Atividade Prática (40 minutos)

#### Desafio Inicial

- Criar a ActivityMain.xml para o layout:



## 1. Cabeçalho:

- 1.1. Um banner azul na parte superior com o logo do aplicativo "TuristAI" centralizado e escrito em branco.

## 2. Local e País

- 2.1. Um campo de texto onde o usuário pode digitar o local desejado para a viagem.
- 2.2. Um texto explicativo "Entre com o local:".
- 2.3. Um spinner (caixa de seleção) onde o usuário pode escolher o país de destino.
- 2.4. Um texto explicativo "Selecione o país:".

## 3. Datas da viagem:

- 3.1. Dois campos de texto onde o usuário pode selecionar as datas de início e fim da viagem.
- 3.2. Os campos não permitem digitação direta, mas abrem um calendário para escolha da data.
- 3.3. Um texto explicativo "Escolha a data Inicial:" para o primeiro campo.
- 3.4. Um texto explicativo "Escolha a data Final:" para o segundo campo.

## 4. Botão de pesquisa:

- 4.1. Um botão Azul com o texto "Pesquisar" localizado no canto direito da tela.

## 5. Área de resultados:

- 5.1. Uma caixa de rolagem vertical que deve exibir o roteiro sugerido pelo aplicativo após a pesquisa.
- 5.2. Inicialmente, essa área exibe apenas o texto "Seu roteiro".

## 6. Botão flutuante de ação:

- 6.1. Um botão circular flutuante na parte inferior direita da tela com uma lixeira como ícone.
- 6.2. A função desse botão servindo para limpar os dados inseridos pelo usuário.

- No MainActivity fazer as seguintes operações:
  1. Criar as variáveis de trabalho.
  2. Configurar a entrada do local.
  3. Configurar a lista de países:
    - 3.1. Criar o spinner
    - 3.2. Criar e configurar o adapter.
    - 3.3. Associar o adapter ao spinner.
    - 3.4. Tratar a seleção do usuário.
    - 3.5. Tratar do item não selecionado.
  4. Fazer o tratamento de data inicial e final: Selecionando datas com DatePickerDialog.
  5. Declarar a saída de texto.
  6. Implementar a API Gemini utilizando o modelo generativo já obtido.
  7. Implementar a Lógica de Consulta.
  8. Configurar o botão de pesquisa.
  9. Configurar o botão de limpar.

## **Ampliação do Desafio**

### **Colocar Imagem de fundo**

## **5. Encerramento e Orientações Finais (20 minutos)**

- Discussão sobre as principais dificuldades encontradas na implementação de listagens dinâmicas.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- Discussão sobre as principais dificuldades encontradas na implementação de listagens dinâmicas.
- **Resumo Teórico:** Explicação sobre os componentes **Calendar** e **FAB**.

**Código-Fonte Comentado:** Explicação detalhada do código desenvolvido.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Organização do código comentado.
	3,0	Funcionamento da solução.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos desenvolveram um aplicativo completo utilizando os conhecimentos adquiridos na disciplina.

**Bom estudo e boa prática!**



Instituto de Ciências  
Exatas e Tecnologia

**Disciplina:** Desenvolvimento para  
Dispositivos Móveis

**Título da Aula:** Protótipo de  
Jogo Simples no Android

**ROTEIRO 16**

## Roteiro da Aula Prática – Protótipo de Jogo Simples no Android

### 1. Objetivos da Aula

- Construir um jogo simples utilizando o Jetpack Compose.
- Compreender a mecânica e a evolução do jogo.
- Conhecer a estrutura de arquivos do jogo.
- Aprender o uso de corrotinas.

### 2. Recursos Necessários

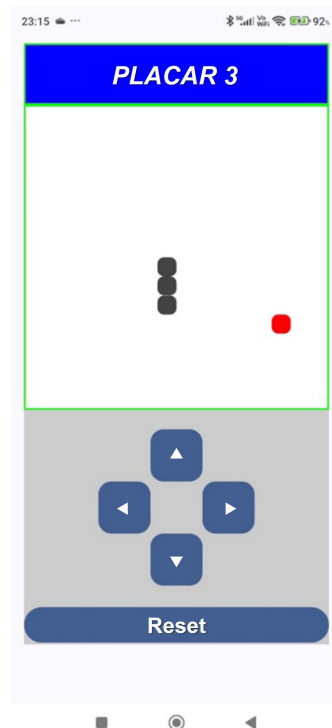
- Computadores com **Android Studio** instalado e/ou acesso a plataformas interativas para o desenvolvimento de aplicativos **Android**.
- Conexão com a internet.
- Material de apoio.
- Emulador Android ou dispositivo físico para testes (opcional).



### 3. Estrutura da Aula

#### 3.1. Abertura (10 minutos)

- Apresentação dos objetivos da aula conhecer o jogo da cobrinha 2.



- Discussão inicial: "Quais são as dificuldades envolvendo a criação de um jogo, jogos simples podem fazer sucesso?"
- Demonstração do jogo pronto.

#### 3.2. Revisão Conceitual (20 minutos)

- Processo de funcionamento de corrotinas.
- Uso de **Box** e **BoxWithConstraints** e o offset.
- Entendimento do processo matemático envolvendo o movimento.
- Entendimento do processo matemático envolvendo as coordenadas e limites.

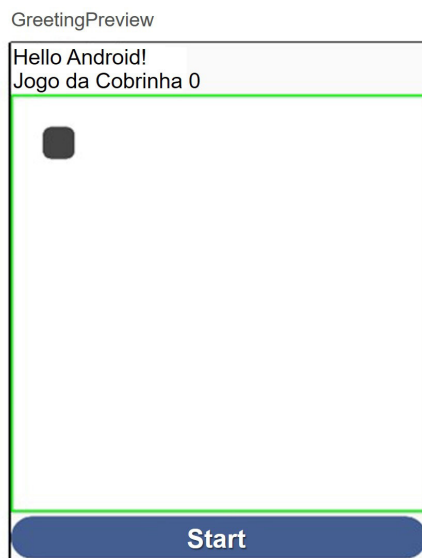
### 3.3. Demonstração Prática (30 minutos)

- Configuração de um novo projeto **Android Empty Activity**.
- Criar uma nova classe em pasta separada (Logica. kt).
- Implementar os MutableState, Suspend fun/ delay e LaunchedEffect.
- Implementar o botão para Start e Pause.

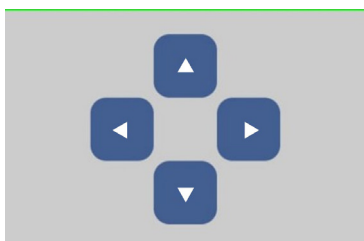
### 4. Atividade Prática (40 minutos)

#### Desafio Inicial

1. Desenhar o Objeto principal (Tabuleiro 16x16 quadrado com largura e altura baseadas na largura da tela do dispositivo).
2. Desenhar o quadro correspondente à tela da cobra:



3. Implementar o movimento.
  - a. Fazer o tratamento dos limites do tabuleiro.
4. Movimentar o objeto por meio do comando do usuário.
  - a. Criar o layout dos Botões direcionais:



5. Colocar o corpo da cobra e exibi-lo na tela.
6. Criar a comida.
7. Implementar a colisão com a comida.
  - a. Aumentar o comprimento da cobra.
  - b. Gerar nova comida.
8. Implementar a colisão com o próprio corpo:
  - a. Tratamento do Game Over.
9. Adicionar o placar.

### **Ampliação do Desafio**

- Aumento de dificuldade:
  - Aumentar a velocidade a cada x comidas.
  - Geração de comida tóxica.

### **5. Encerramento e Orientações Finais (20 minutos)**

- Discussão sobre as principais dificuldades encontradas na implementação de listagens dinâmicas.

## 6. Orientações para o Relatório Final

Cada aluno deve produzir um relatório curto (1 a 2 páginas) contendo:

- **Resumo Teórico:** Explicação sobre a mecânica e a evolução do jogo. Explicar a estrutura de arquivos do jogo.
- **Código-Fonte Comentado:** Explicação detalhada do código desenvolvido.

## 7. Critérios de Avaliação

Critério	Peso descrição	Descrição
Qualidade do Resumo Teórico	2,0	Clareza, objetividade e demonstração de entendimento sobre a teoria abordada na aula.
Estrutura e Organização do Código Funcionamento da Solução	3,0	Organização do código comentado.
	3,0	Funcionamento da Solução.
Criatividade e Aprimoramentos	2,0	Inclusão de melhorias que demonstrem domínio do conteúdo.

**Nota Final:** Será a soma dos valores obtidos em cada critério. Alunos ou equipes que não cumprirem os requisitos mínimos de funcionamento do código ou não entregarem o relatório dentro do prazo terão sua nota diminuída proporcionalmente.

## 8. Conclusão

Nesta aula, os alunos construirão um jogo utilizando todo o conhecimento aprendido na disciplina.

**Bom estudo e boa prática!**