Plano Integrado de Projeto
SimuBlock

Visão Geral

Desenvolver um **mini-Simulink** para **Android (Kotlin no Android Studio)** com:

- **Editor de blocos** (drag-and-drop, conexões entre portas).
- **Motor de simulação** contínua (SISO) com integração numérica (Euler/Runge–Kutta).
- **Scope** para visualização da resposta (degrau no Mínimo Produto Viável).
- **Banco de dados** para persistir diagramas, parâmetros e resultados (foco em BD I).

Disciplinas atendidas

- **Banco de Dados I**: Requisitos, MER, modelo físico, normalização, SQL, telas no APEX (ou script + mock se o professor preferir).
- **Programação Mobile**: App Android com editor de blocos, simulação e gráficos.

Escopo por Fase (Evolutivo)

Fase A — MVP (Editor simples + simulação 1ª/2ª ordem)

- Blocos: *Step, Gain, Sum(+-), Integrator, Scope.*
- Conexões: 1 entrada, 1 saída (SISO), grafo acíclico.
- Simulação: resposta ao degrau, método de Euler; passo fixo Δt.
- Salvamento: diagrama e parâmetros (sem resultados cronometrados).

Fase B — Robustez (motor e editor)

- Runge–Kutta 4 (RK4) e detecção simples de ciclos inválidos.
- Zoom/pan no canvas, snapping e grade.
- Tipagem de portas (evitar conexões inválidas).
- CRUD completo no BD (diagramas, blocos, conexões, simulações, resultados).

Fase C — Extensões (opcional)

- Entradas: impulso, rampa; bloco *Transfer Function* (2ª ordem geral).
- Exportar/importar diagramas (JSON).
- Histórico de simulações e comparação de curvas.

Arquitetura do App (Android Studio)

Camadas

- **UI**: Canvas customizado (View/Compose) para blocos e conexões; tela de parâmetros; Scope.
- **Core**: Grafo de blocos → sistema de equações; *scheduler* de simulação.
- **Sim**: Integradores numéricos (Euler, RK4); gerador de entrada (degrau).
- **Dados**: Repositório com persistência (Room/SQLite) e/ou API futura.

Modelo de Blocos (conceito)

Cada bloco implementa:

- `evaluate(dt)` para avançar o estado.
- Portas de entrada/saída tipadas (double).
- Parametrização (ganho, sinais de soma, condição inicial).

Pipeline de simulação

1. Validar grafo (sem ciclos não permitidos no MVP).
2. Ordenar topologicamente os blocos.
3. Laço de tempo: aplicar entrada, propagar sinais, integrar estados.
4. Registrar saída no *Scope*.

Banco de Dados I

Entidades (MER)

- **Diagrama**(<u>id</u>, nome, criado_em).
- **Bloco**(<u>id</u>, diagrama_id, tipo, x, y, params_json).
- **Conexao**(<u>id</u>, diagrama_id, bloco_origem, porta_origem, bloco_destino, porta_destino).
- **Simulacao**(<u>id</u>, diagrama_id, dt, t_final, metodo).
- **Resultado**(<u>id</u>, simulacao_id, t, y).

Consultas SQL sugeridas

1. Listar blocos e parâmetros de um diagrama.
2. Obter conexões (origem→destino) de um diagrama.
3. Recuperar séries (t,y) de uma simulação para plot.
4. Filtrar diagramas por data e tipo de bloco presente.
5. Contar simulações por método (Euler vs RK4).

Conjunto de Blocos (versão inicial)

Bloco	Portas	Parâmetros
Step	out:1	amplitude, atraso, offset
Gain	in/out:1	K
Sum	in:2, out:1	sinais (`+ +`, `+ -`, etc.)
Integrator	in/out:1	condição inicial $x(0)$
Scope	in:1	buffer de exibição (tamanho N)

Cronograma

Mês 1	Requisitos finais; protótipo do canvas (arrastar blocos e posicionar); modelo de dados (MER); script inicial SQL.
Mês 2	Conexões entre blocos; serialização do diagrama (JSON/DB); motor Euler (1ª/2ª ordem); Scope simples.
Mês 3	RK4; validações (portas, tipos, ciclos); UI de parâmetros; persistência (Room/SQLite) e/ou APEX para BD I.
Mês 4	Telas CRUD no APEX; consultas SQL; comparação de simulações; otimizações de UI (zoom/pan, snapping).
Mês 5	Refinos; testes; exportar/importar diagramas; documentação; ensaio de apresentação.

Entregas por Disciplina

Banco de Dados I

- **Entrega 02**: MER, modelo relacional (3FN), script DDL.
- **Entrega 03**: 5+ consultas SQL, telas APEX (ou mock), carga de exemplo e um gráfico (APEX) com dados de *Resultado*.

Programação Mobile

- App Android (Kotlin) com: editor de blocos, conexões, parâmetros, simulação (Euler/RK4) e Scope.
- **Demonstração**: montar Step→Gain→Integrator→Scope; simular degrau e exibir curva.

Riscos & Mitigações

- **Canvas e conexões**: começar com MVP (sem múltiplas saídas, sem loops). Evoluir após validação.
- **Motor de simulação**: começar por Euler e validar; depois RK4 para estabilidade.
- **Tempo**: fixar escopo (blocos mínimos) e adiar features extras (impulso, rampa, TF genérica) para a Fase C.

Critérios de Conclusão (DoR/DoD)

- Editor permite inserir, mover e conectar blocos mínimos.
- Simulação executa e o Scope exibe $y(t)$ estável com degrau.
- Banco salva diagrama, simulação e pelo menos uma série (t,y).
- Consultas SQL e telas APEX demonstráveis.

Anexos Úteis

Exemplo de função de transferência 2ª ordem

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Pseudo-estrutura JSON de diagrama

```
{
 "diagramId": 1,
 "blocks": [
   {"id": "b1","type": "Step","x": 40,"y": 80,"params":{"amp":1}},
   {"id": "b2","type": "Gain","x": 160,"y": 80,"params":{"K":2}},
   {"id": "b3","type": "Integrator","x": 280,"y": 80,"params":{"x0":0}},
   {"id": "b4","type": "Scope","x": 420,"y": 80}
 ],
 "connections": [
   {"from":"b1:out1","to":"b2:in1"},
   {"from":"b2:out1","to":"b3:in1"},
   {"from":"b3:out1","to":"b4:in1"}
 ]
}
```