Detalhamento dos 5 Projetos

1. Sistema de Gestão de Biblioteca (CLI)

Objetivo: Criar um sistema em linha de comando (CLI) para gerenciar livros, membros e empréstimos.

- **Requisitos Mínimos:**
  - Classes: Livro, Membro, Emprestimo.
  - Funcionalidades (CLI): Adicionar, Listar, Buscar (por título/membro), Realizar Empréstimo, Devolver Livro.
- **Foco na Modelagem:**
  - **Relacionamento:** Um Membro pode ter vários Emprestimos. Um Livro pode estar em um Emprestimo.
  - **Herança (Opcional):** Criar LivroFisico e LivroDigital herdando de Livro.
  - **Persistência:** O estado do sistema deve ser salvo e carregado (inicialmente em arquivo JSON/CSV e, se o aluno já se sentir seguro, migrar para **SQLite**).

2. Simulador de Transações Bancárias

Objetivo: Desenvolver uma simulação de um sistema bancário simples focado em segurança, exceções e testes.

- **Requisitos Mínimos:**
  - Classes: Conta (Abstrata), ContaCorrente, ContaPoupanca.
  - Funcionalidades: Sacar, Depositar, Transferir, Ver Saldo.
  - **Regras de Negócio:**
    - Saques não podem exceder o saldo (exceto ContaCorrente com limite).
    - Transferências só entre contas existentes.
    - Uso obrigatório de **Tratamento de Exceções** (ex: SaldoInsuficienteError).
- **Foco em POO e Testes:**
  - Uso intensivo de **Encapsulamento** (getters/setters protegendo o saldo).
  - Criação de **Testes Unitários** para as regras de saque e depósito.
  - Aplicação do princípio **LSP** (Liskov Substitution Principle) na herança de Conta.

3. Gerador de Relatórios e Análise de Dados

Objetivo: Criar um sistema que carrega dados (simulados ou de um DB) e gera relatórios agregados.

- **Requisitos Mínimos:**
  - **Fonte de Dados:** Um arquivo CSV de vendas ou um DB **SQLite** preenchido (tabelas: Produtos, Vendas).

    - **Relatórios (Funções de Agregação):**
        - Total de vendas por produto (SUM).
        - Média de preço de produtos (AVG).
        - Produtos mais/menos vendidos (ORDER BY + LIMIT).
        - Vendas por categoria (GROUP BY).
- **Foco em SQL e Funções:**
    - Uso de **SELECT complexos** com WHERE, ORDER BY, GROUP BY e INNER JOIN.
    - Modularização do código (funções separadas para carregar dados, processar e imprimir relatório).

4. Sistema de Cadastro de Clientes (CRUD Básico)

Objetivo: Construir um CRUD (Create, Read, Update, Delete) de clientes com foco na integração robusta com Banco de Dados.

- **Requisitos Mínimos:**
    - **Modelagem ER:** Tabela Cliente (id, nome, email, telefone).
    - **Integração DB:** Uso de **SQLite** ou **PostgreSQL** para persistir os dados.
    - **Operações CRUD:** Funções separadas para Inserir (CREATE), Listar (READ), Atualizar (UPDATE) e Excluir (DELETE) clientes.
- **Foco em Padrão DAO:**
    - Implementação do **Padrão DAO (Data Access Object)**: Uma classe (ClienteDAO) responsável por toda a interação com o banco, separando a lógica de negócio do acesso aos dados.
    - Uso de *queries* SQL parametrizadas para evitar *SQL Injection* (segurança básica).

5. Sistema de Controle de Tarefas com Padrões de Projeto

Objetivo: Desenvolver um sistema de gestão de tarefas focado na aplicação prática de Padrões de Projeto.

- **Requisitos Mínimos:**
    - Classes: Tarefa, GerenciadorDeTarefas.
    - Funcionalidades: Adicionar, Listar, Concluir, Filtrar tarefas.
    - **Aplicação de Padrões:**
        - **Strategy:** Implementar diferentes **Estratégias de Ordenação** (por data de criação, por prioridade, por título) que o usuário possa escolher.
        - **Factory/Singleton:** Uso de um **Factory Method** para criar diferentes tipos de tarefas (ex: TarefaComum, TarefaUrgente) e/ou o **Singleton** para garantir uma única instância do GerenciadorDeTarefas (ou da conexão com o Logger/DB).
- **Foco em Modularidade:**
    - Estruturar o código em diferentes módulos (models.py, patterns.py, main.py) para demonstrar o conceito de modularização.

Roteiro das 4 Fases de Entrega a partir de 06/11

Entrega	Data	Foco Principal	Entregável Simplificado
Fase 1	Próxima quinta	Modelagem e Estrutura	Documentação: Requisitos (RFs) e Diagrama de Classes/ER. Código: Estrutura de Pastas e Classes vazias no Git.
Fase 2	Próxima terça	Lógica Essencial e POO	Código: Implementação da Lógica de Negócio central (métodos chave) e uso correto do Encapsulamento. Sistema funcional no terminal.
Fase 3	Quinta (Semana 2)	Persistência de Dados (CRUD)	Código: Funções de Criação (CREATE) e Leitura (READ) de dados que interagem com o DB ou arquivo. Testes: 2 a 3 Testes Unitários básicos.
Fase 4	Terça (Semana 2)	Conclusão e Qualidade	Código: Implementação das operações UPDATE e DELETE (CRUD completo) + Tratamento de Exceções. Finalização: README com instruções e código refatorado.