🚀 Roteiro de Aceleração: Engenheiro Fullstack AI-Driven

Objetivo: Tornar-se um Engenheiro de Software Fullstack capaz de arquitetar, auditar e integrar sistemas gerados por IA. **Metodologia:** "Learning by Building" (Aprender construindo). **Projeto Principal:** "ServiceMarket" - Marketplace de Serviços Profissionais. **Stack:** Java 17+, Spring Boot 3, PostgreSQL, React, Tailwind CSS.

🧭 Como usar este Guia

1. **Copie os Prompts:** Use os textos sugeridos para guiar a IA.
2. **Audite o Código:** Nunca aceite o primeiro resultado. Use a seção "Auditoria" para verificar falhas.
3. **Marque o Progresso:** Utilize as caixas de seleção `[ ]` para gamificar o seu avanço.

🏆 O Projeto: "ServiceMarket"

Uma plataforma onde prestadores (encanadores, eletricistas) oferecem serviços e clientes agendam visitas.

- **Diferencial:** Complexidade real de regras de negócio (conflito de agenda, cálculo de valores) e segurança (perfis de acesso).

📅 Fase 1: Fundação e Dados (Semana 1)

Meta: Ter o banco de dados modelado e a lógica de negócios desenhada, garantindo integridade dos dados.

1.1 Conceitos Essenciais (Estudo Teórico)

- [ ] **POO Aplicada:** Encapsulamento (`private`), Polimorfismo (Interfaces) e Herança vs. Composição.
- [ ] **Modelagem ER:** Cardinalidade (1:N, N:N) e Normalização de tabelas.
- [ ] **SQL Moderno:** Constraints (`UNIQUE` , `CHECK`) e Índices.

1.2 Tarefas Práticas (Mão na Massa)

- [ ] **Setup do Projeto:**
  - Criar projeto no Spring Initializr (Web, JPA, Postgres, Validation, Lombok).
  - Configurar `application.properties` com as credenciais do banco.
- [ ] **Modelagem de Entidades:**
  - Criar entidade `User` com Enum `UserProfile` (ADMIN, CLIENT, PROVIDER).
  - Criar entidade `Service` (Preço com `BigDecimal` , não Double).
  - Criar entidade `Booking` (Agendamento) com status e data.

- [ ] **Relacionamentos:**
  - Mapear `@OneToMany` (Um Prestador -> Vários Serviços).
  - Mapear `@ManyToOne` (Um Agendamento -> Um Cliente).

🛡️ Auditoria de IA (Onde a IA falha)

- **Erro de Precisão:** A IA sugeriu `double` para dinheiro?
  - *Correção:* Mude para `BigDecimal`.
- **Erro de Integridade:** A IA esqueceu `@Column(nullable = false)`?
  - *Correção:* Adicione restrições para evitar dados nulos onde não deve.

👹 Prompt da Fase 1

> "Atue como um Arquiteto de Software Java Sênior. Crie o código das entidades JPA para um sistema de agendamento. Regras: 1. Use BigDecimal para preços. 2. Use UUID para IDs. 3. Mapeie o relacionamento onde um Prestador tem vários Serviços. Explique as anotações usadas."

📅 Fase 2: API e Arquitetura (Semana 2)

Meta: Construir uma API REST segura, organizada em camadas e validada.

2.1 Conceitos Essenciais

- [ ] **DTO Pattern:** Por que separar Entidade de Banco vs. Objeto de Transferência.
- [ ] **Camadas:** Controller (HTTP) -> Service (Regras) -> Repository (Dados).
- [ ] **Verbos HTTP:** Uso semântico de GET, POST, PUT, DELETE, PATCH.

2.2 Tarefas Práticas

- [ ] **CRUD de Serviços:**
  - Criar `ServiceDTO` com validações (`@NotNull`, `@Positive`).
  - Implementar `ServiceController` e `ServiceService`.
- [ ] **Regras de Agendamento:**
  - Validar: "Data de agendamento deve ser futura".
  - Validar: "Prestador não pode agendar consigo mesmo".
- [ ] **Tratamento de Erros:**
  - Implementar `@ControllerAdvice` para capturar exceções e retornar JSON limpo.

🛡️ Auditoria de IA

- **Mass Assignment:** A IA usou a Entidade direto no Controller?
  - *Correção:* Obrigue o uso de DTOs.

- **N+1 Selects:** A busca de serviços está fazendo um SELECT extra para cada usuário?
  - *Correção:* Use `JOIN FETCH` na query JPQL.

👹 Prompt da Fase 2

> "Atue como um Desenvolvedor Backend Especialista. Crie um Service para agendamento que valide: 1. Se a data é futura. 2. Se já existe agendamento naquele horário (conflito). Use DTOs na entrada e saída. Lance exceções customizadas para erros de validação."

📅 Fase 3: Segurança e Testes (Semana 3)

Meta: Blindar o sistema com Login (JWT) e garantir estabilidade com testes.

3.1 Conceitos Essenciais

- [ ] **Spring Security:** Autenticação Stateless (JWT) e Filter Chain.
- [ ] **RBAC:** Controle de acesso baseado em Roles (`@PreAuthorize`).
- [ ] **Testes:** Unitários (Mockito) vs Integração.

3.2 Tarefas Práticas

- [ ] **Autenticação:**
  - Implementar endpoint `/login` que retorna Token JWT.
  - Criar Filtro para validar o Token em cada requisição.
- [ ] **Proteção de Rotas:**
  - Apenas ADMIN pode deletar usuários.
  - Apenas PROVIDER pode criar serviços.
- [ ] **Testes Automatizados:**
  - Criar teste unitário para a regra de "Conflito de Horário".

🛡️ Auditoria de IA

- **Segredos Expostos:** A IA colocou a chave JWT no código?
  - *Correção:* Mova para variáveis de ambiente (`System.getenv()`).
- **Testes Vazios:** O teste gerado tem `assert` ?
  - *Correção:* Verifique se o teste falha quando você quebra a regra propositalmente.

👹 Prompt da Fase 3

> "Atue como um Engenheiro de Segurança (AppSec). Analise este código de Controller. Identifique rotas que deveriam ser protegidas e sugira a implementação de `@PreAuthorize` . Procure também por vulnerabilidades de IDOR (Insecure Direct Object Reference)."

📅 Fase 4: Integração Frontend (Semana 4)

Meta: Criar uma interface React que consome a API de forma eficiente.

4.1 Conceitos Essenciais

- [ ] **React Hooks:** `useEffect` (efeitos colaterais) e `useState` (memória).
- [ ] **Integração API:** Configurar Axios com Interceptors para enviar o Token.
- [ ] **UX:** Feedback de carregamento (Skeleton) e Erros (Toasts).

4.2 Tarefas Práticas

- [ ] **Autenticação no Front:**
  - Tela de Login salvando Token no LocalStorage.
  - Redirecionar usuário se o Token expirar.
- [ ] **Dashboard:**
  - Listar serviços disponíveis.
  - Modal de Agendamento.
- [ ] **Meus Agendamentos:**
  - Lista com status colorido (Verde/Vermelho).

🛡️ Auditoria de IA

- **Loop Infinito:** O `useEffect` está rodando sem parar?
  - *Correção:* Verifique o array de dependências `[]`.
- **XSS:** A IA usou `dangerouslySetInnerHTML`?
  - *Correção:* Remova imediatamente.

🤖 Prompt da Fase 4

> "Atue como um Desenvolvedor Frontend React Sênior. Crie um hook customizado `useAuth` que gerencia o login, logout e verifica se o token JWT ainda é válido. Use Context API para compartilhar o estado do usuário."

✅ Checklist Final do Recrutador

Antes de colocar no currículo, garanta:

1. [ ] **README Incrível:** Com diagrama de arquitetura e instruções de instalação.
2. [ ] **Código Limpo:** Sem comentários inúteis ou `System.out.println`.
3. [ ] **Histórico Git:** Commits semânticos (`feat:` , `fix:` , `refactor:`).
4. [ ] **Segurança:** Nenhuma chave de API ou senha no código.
5. [ ] **Demo:** (Opcional) Um vídeo curto (Loom) mostrando o sistema funcionando.