Estudo Dirigido 007

Etapas 0 e 1 — Do Problema à Modelagem e Escolha de Framework (Django ou FastAPI)

Objetivo Geral
Consolidar a compreensão sobre como transformar um problema real (descrito em um briefing) em requisitos, fluxos e escolhas técnicas iniciais de um projeto backend.
Contexto: Projeto PATRI-TECH

O PATRI-TECH é um sistema de inventário e controle patrimonial destinado à Secretaria Municipal de Educação. Ele deve permitir identificação, contabilização e classificação de bens com o uso de tecnologia RFID, garantindo agilidade e precisão nas informações.

PARTE 1 - Leitura e Interpretação Crítica

Análise do briefing

1. Identifique e descreva, com suas palavras, o problema central que o sistema PATRI-TECH precisa resolver.

O problema central que o PATRI-TECH precisa resolver é a gestão ineficiente e manual dos bens patrimoniais da Secretaria Municipal de Educação. A "dor real" reside na dificuldade de saber exatamente quais bens existem, onde estão localizados, seu estado de conservação e quem é responsável por eles.

2. Liste três consequências para a Secretaria se esse problema não for solucionado.

1. Perda financeira e extravio de bens: Dificuldade em identificar e recuperar itens perdidos ou roubados devido à falta de controle e visibilidade em tempo real.
2. Problemas na prestação de contas e auditorias: Risco de não conformidade com a legislação e órgãos de controle (como o Tribunal de Contas), podendo gerar sanções.
3. Ineficiência operacional e tomada de decisão ruim: Tempo desperdiçado em inventários manuais demorados e dificuldade em planejar a substituição ou aquisição de novos equipamentos por não se saber o que já se tem.

3. Analise os trechos do briefing e destaque três verbos de ação e três adjetivos ou expressões qualitativas. Depois, classifique:

Requisitos Funcionais	Requisitos Não Funcionais
Cadastrar bens	Planejar bem aquisições
Cadastrar novos usuários	Desempenho
Crud	Segurança da integridade dos dados
Localizar equipamentos e contabilizar	Disponibilidade do sistema
Verificar status do equipamento	Escalabilidade

Vincular as unidades	Compatibilidade

● Verbos → possíveis requisitos funcionais
● Adjetivos → possíveis requisitos não funcionais

4. Reflita: quais são as restrições reais (de infraestrutura, tempo, equipe, orçamento) que podem impactar o desenvolvimento?

- Infraestrutura: Necessidade de aquisição e instalação de leitores RFID e etiquetas compatíveis em todas as unidades escolares ou locais de armazenamento. A infraestrutura de rede local (Wi-Fi/cabeada) precisa ser estável o suficiente para a comunicação dos dispositivos.
- Tempo: Prazos apertados para implementação, talvez coincidindo com o ano letivo, o que dificulta o treinamento e a coleta inicial de dados.
- Equipe: Equipe de desenvolvimento enxuta ou com pouca experiência em tecnologias específicas (como integração com hardware RFID).
- Orçamento: Recursos financeiros limitados para a compra da tecnologia RFID (hardware e etiquetas), que pode ser cara dependendo da escala.

PARTE 2 - Raciocínio Técnico: Requisitos e Atores

5. Usando a técnica MoSCoW, organize os seguintes requisitos em prioridades: (adicione outros se desejar)
MosCow é um método ou ferramenta de priorização usada na análise de negócio e gestão de projeto de softwares com o objetivo de encontrar um entendimento comum entre as partes interessadas sobre determinada questão.

Must Have (Prioridade Absoluta - Sem isso o sistema não funciona):

  - Fazer login seguro (Segurança básica).
  - Cadastrar bens (Função central do inventário).
  - Ler etiquetas RFID (Tecnologia chave do projeto).
  - Atualizar status dos bens (Manutenção da integridade dos dados).
- **Should Have (Importante, mas o sistema pode operar sem temporariamente):**
  - Gerar relatórios (Essencial para a gestão, mas pode ser feito manualmente no início).
- **Could Have (Desejável, "nice to have"):**
  - Exportar relatórios em PDF (Uma conveniência de formato).

6. Diferencie com suas palavras:

- **O que caracteriza um requisito funcional (RF)?**

    - Descreve *o que* o sistema deve fazer. São as funcionalidades, as ações, os comportamentos e as interações diretas do usuário com o sistema para atingir um objetivo de negócio. (Ex: "O sistema deve permitir o cadastro de um novo item").
- **O que caracteriza um requisito não funcional (RNF)?**
    - Descreve *como* o sistema deve ser. São atributos de qualidade, restrições e critérios de performance que limitam as escolhas de design e implementação. (Ex: "O sistema deve carregar a página em menos de 3 segundos", "O sistema deve ser seguro").

7. Identifique três atores que interagem com o sistema e explique suas funções. (Dica: um ator pode ser uma pessoa ou outro sistema.)

1. **Servidor/Funcionário de Inventário:** Responsável por realizar a leitura física dos bens usando o leitor RFID, cadastrar novos itens e atualizar localizações.
2. **Gestor/Secretário(a):** Utiliza o sistema para visualizar relatórios gerenciais, tomar decisões estratégicas, auditar dados e ter uma visão geral do patrimônio.
3. **Sistema de Auditoria Externa (Outro Sistema):** Um potencial sistema externo (como do Tribunal de Contas) que pode interagir via API para extrair dados de conformidade automaticamente.

8. Descreva, em uma sequência lógica, o fluxo de um inventário típico usando o sistema. (Quem faz o quê, e o que o sistema precisa fazer em cada etapa?)

Fluxo: Realização de Inventário Periódico

1. **Servidor** acessa o módulo de inventário no sistema (web ou app mobile).
2. **Sistema** solicita a identificação do local (ex: "Escola X, Sala 1").
3. **Servidor** caminha pela sala com o leitor RFID (conectado ao sistema).
4. **Leitor RFID** escaneia múltiplas etiquetas simultaneamente e envia os IDs para o sistema.
5. **Sistema** processa os IDs, compara com o banco de dados de bens esperados para aquela localização.
6. **Sistema** marca os bens encontrados como "presentes" no inventário atual e atualiza automaticamente a data da última localização.
7. **Sistema** exibe um relatório em tempo real na tela do servidor: "X itens encontrados, Y itens não encontrados/divergência".
8. **Servidor** finaliza a sessão de inventário para aquela sala.

PARTE 3 - Modelagem Conceitual e Decisões Técnicas

9. A partir dos atores e fluxos, elabore 3 casos de uso (em formato textual, não precisa diagramar). Exemplo:

1. **Cadastro de Novo Bem:** O Servidor acessa o formulário de cadastro, preenche detalhes do bem (descrição, categoria, valor), associa a etiqueta RFID (lendo-a uma vez), e o sistema persiste o novo registro no banco de dados.
2. **Visualização de Relatório de Auditoria:** O Gestor acessa o menu de relatórios, seleciona um período e uma unidade escolar, e o sistema gera um relatório detalhado de todos os bens inventariados naquele período.
3. **Atualização de Localização:** O Servidor move um bem de uma sala para outra, lê a etiqueta RFID do bem, seleciona a nova localização no sistema, e o sistema atualiza o registro do bem com o novo local.

10. Escolha um dos casos de uso e descreva:
● Entrada de dados
Os dados fornecidos pelo usuário (via formulário ou API).

- Descrição detalhada.
- Categoria do bem (ex.: eletrônico.)
- Número de série da TAG do equipamento.
- Data de aquisição.
- Localização do bem (ex.: laboratório, escola , sala.)
- Estado de conservação (novo, danificado etc.)

● Processamento
O sistema recebe os dados e valida: Se todos os campos obrigatórios foram preenchidos
se o equipamento foi cadastrado corretamente,Verifica se já existe um equipamento com
id: cadastrado,Em caso de inconsistências, o sistema gera erros específico, Se tudo estiver correto, os dados do usuários e equipamentos são salvos no banco de dados.
Retorna uma confirmação de sucesso.

● Saída esperada

{“message”:”equipamento cadastrado com sucesso”,}

11. Com base nos requisitos e nas necessidades do projeto, qual framework você escolheria: Django ou FastAPI? Justifique sua escolha com base em:

Django: aplicações completas com painel admin, templates e autenticação pronta..
● Tempo e curva de aprendizado é maior pois O Django é um framework “completo” pois ja vem com estruturas (views, models, templates, migrations, apps).

● Tipo de sistema (painel web x API)

- Se o sistema for um *painel web Melhor escolha Django,Templates, Admin, formulários, autenticação pronta*

- *Se o sistema for uma API , Exemplo: backend que fornecerá dados para,App mobile,Sistema externo,Página em React,Microsserviços. Melhor escolha: FastAPI.*

● Tempo e curva de aprendizado
- Django : Curva de aprendizado maior, pois é um framework.

● Necessidade de integração

FastAPI é superior quando:

- Você vai integrar com mobile (Flutter, React Native)

- Vai usar frontend separado (React, Vue, Next)

- Vai expor endpoints para outros sistemas

Django é melhor quando:

- Você quer tudo integrado (frontend + backend + admin)

- Precisa de um painel administrativo pronto imediatamente

● Requisitos de desempenho

- **FastAPI** → Desempenho muito superior, graças ao ASGI e suporte nativo a async. OBs.(ASGI serve para dar compatibilidade entre servidores web, frameworks e aplicações). Async é uma função que não para a execução do seu código.

- **Django** → Mais pesado, orientado a WSGI. Funciona bem, mas não é tão rápido para requests muito volumosos. OBS(requests : refere-se a uma solicitação feita por um cliente ou usuário a um servidor)

12. Quais seriam os riscos de uma escolha equivocada de framework neste estágio do projeto?
- pode ter que refazer partes inteiras do projeto caso o framework não atenda às necessidades.
- no caso Django para algo que precisa de desempenho muito alto, pode sofrer com lentidão e escalabilidade.
- Escolher Django quando só precisa de uma API pode deixar o projeto mais complicado do que deveria.
- Um framework inadequado pode limitar integração com mobile ou outros serviços.

- Curva de aprendizado maior sem necessidade pode atrasar entregas.

- A escolha errada pode dificultar a expansão ou adição de novos módulos.

PARTE 4 - Síntese e Aplicação

13. Crie uma frase-síntese (1 linha) que resuma a visão do produto PATRI-TECH.

O PATRI-TECH é um sistema que simplifica o inventário de bens públicos, garantindo agilidade e precisão na gestão patrimonial da Secretaria de Educação.

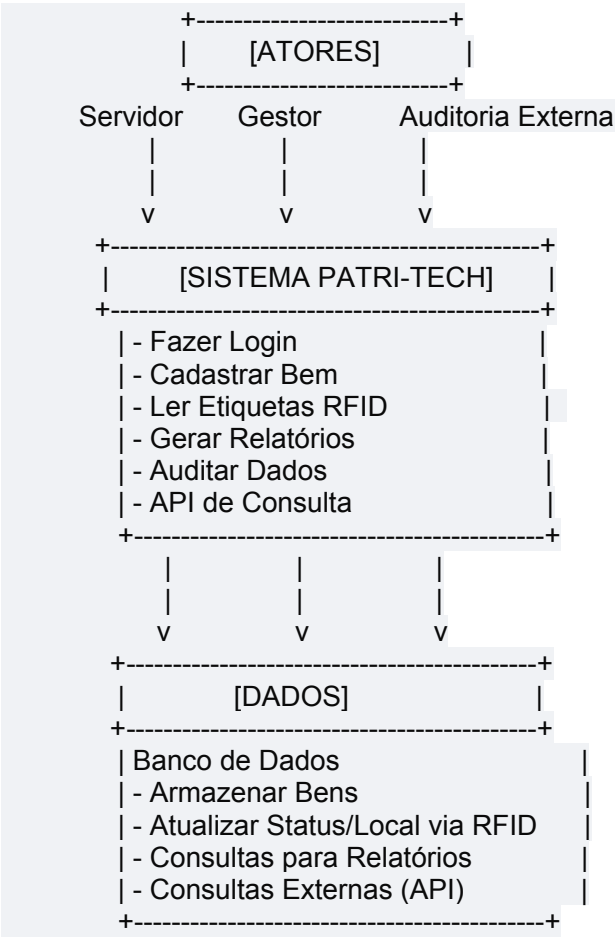
14. Esboce (em texto ou rascunho) um mini-diagrama de casos de uso, listando quem se conecta a quem. (Você pode usar setas textuais, como “Servidor → Sistema → Relatório”.)

[ATORES] [SISTEMA PATRI-TECH] [RELATÓRIOS/DADOS]

Servidor --> Fazer Login
Servidor --> Cadastrar Bem --> Banco de Dados
Servidor --> Ler Etiquetas RFID --> Banco de Dados (Atualização de Status/Local)

Gestor --> Fazer Login
Gestor --> Gerar Relatórios --> Relatório em PDF/Tela
Gestor --> Auditar Dados --> Banco de Dados (Leitura)

Auditoria Externa --> (API de Consulta) --> Banco de Dados (Leitura)



15. Por que é importante alinhar requisitos e arquitetura antes de codar? (Responda como se estivesse orientando outro desenvolvedor.)

Entregáveis.

- Quando você começa a programar sem ter requisitos claros você pode acabar tendo que refazendo partes do sistema, criando soluções improvisadas ou até desenvolvendo algo que não atende completamente o que o usuário realmente precisa alinhar requisitos e arquitetura antes do código traz benefícios diretos como:
- 1. Evita retrabalho
- 2. Facilita decisões técnicas
- 3. Melhora a comunicação entre a equipe
- 4. Aumenta a qualidade do código e do sistema
- 5. Ajuda a prever esforço, custo e prazo

Entregáveis:

- Requisitos (Funcionais e Não Funcionais)
- Diagramas de fluxo UML
- Definição da Arquitetura
- Critérios de Aceitação (ex. que precisa acontecer para considerar uma funcionalidade pronta)
- Plano de Testes.

- Documento .md ou .pdf com as respostas.

- Esboço do diagrama de casos de uso (imagem, Draw.io Miro, TLDRAW, Paint).