Desenho de Solução

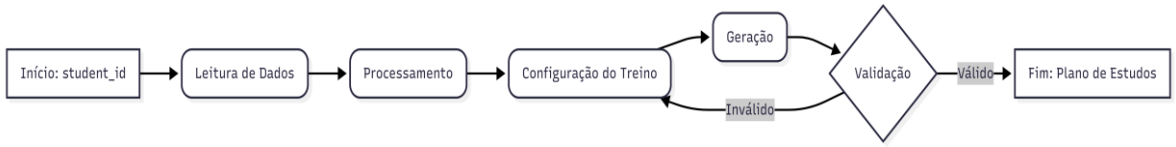
1. **Entendimento do problema**: Explique qual é o problema real, quais informações não estão claras e quais suposições seriam necessárias.
   Resposta:
   O problema central é a demora que o professor enfrenta para criar planos manuais para muitos alunos, o que justifica a criação dessa automação com IA. No entanto, notei que algumas informações importantes não ficaram claras no projeto, como a faixa etária dos alunos, algo que eu precisaria saber para ajustar melhor a linguagem da IA no prompt. Também não foi explicado como calcular o tempo total dos exercícios, o arquivo de política fala em um tempo máximo, mas como os exercícios não trazem o tempo específico de cada um, fica impossível saber se a combinação final respeita esse limite. Além disso, o README pedia 3 exercícios para notas baixas, mas o catálogo só oferecia 2 por categoria, enquanto a política permitia até 5, gerando uma confusão de regras. Para o código funcionar, presumi que todos os arquivos usam os mesmos nomes de campos, como o "skill", para que as informações se cruzassem corretamente. Também notei que existe um "ability_score" para cada aluno, mas como não foi especificado como usar essa nota geral, decidi ignorá-la e focar apenas na menor nota individual para definir o diagnóstico. Por causa dessas incertezas, tomei a iniciativa de criar uma lógica robusta no código que lida com a instabilidade ou falta de dados. Independente de quantos exercícios o catálogo tenha hoje, a regra de negócio sempre será respeitada: o sistema tenta entregar entre o mínimo de 2 e o máximo de 5 exercícios conforme a política, mas se o catálogo não tiver o mínimo necessário ou se as regras de validação não forem atendidas, a aula simplesmente não é validada para garantir a segurança da entrega. Assim, o programa se adapta sozinho e garante um resultado 100% confiável mesmo se o banco de dados mudar no futuro.

2. **Definição da saída**: Descreva que tipo de saída seria útil para o cliente e por quê.
   Resposta:
   A saída útil para o cliente é um Roteiro de Estudo Automatizado. Ele deve conter a habilidade foco, uma explicação resumida do conceito e a lista de exercícios recomendados. Essa saída é útil porque elimina o trabalho manual do professor de cruzar notas de CSV com catálogos de exercícios, garantindo que o aluno receba um material condizente com seu desempenho atual para a melhoria do seu desempenho, tudo isso de forma automática.

3. **Abordagem técnica**: Explique qual abordagem usaria (LLM, ML clássico, regras, combinação) e em que cenário ela deixaria de ser adequada.
   Resposta:
   A abordagem escolhida é híbrida, regras de combinação para a tomada de decisão lógica e IA Generativa (LLM) para a comunicação com o aluno. As regras garantem que o diagnóstico seja preciso e garanta que os exercícios sejam escolhidos corretamente bem como a quantidade sem variações indesejadas. A LLM é aplicada apenas onde a flexibilidade da linguagem é necessária: na criação da explicação personalizada.

4. **Pipeline da solução**: Descreva um fluxo simples da solução e justifique cada etapa. Lembre-se de pensar em onde guardar os dados na nuvem.

Resposta:
O pipeline da solução segue um fluxo de cinco etapas: Leitura de Dados, Processamento, Configuração de Treino, Geração e Validação. Cada etapa é justificada pela necessidade de transformar dados brutos em um planejamento pedagógico personalizada. Para o armazenamento na nuvem, os dados de treino, planejamento e perfis de alunos seriam armazenados para um banco de dados NoSQL para garantir escalabilidade e acesso rápido ao estado do agente, enquanto o catálogo de exercícios e notas de alunos poderia ser armazenado em um banco relacional, permitindo que o sistema suporte milhares de alunos simultâneos sem perda de performance. Segue o diagrama visual do pipeline:



5. **Riscos e limitações**: Explique onde a solução pode errar e como reduzir riscos, incluindo alucinação.
   Resposta:
   A solução apresenta alguns riscos críticos: a alucinação da LLM, que pode comprometer o ensino com informações falsas; o risco de loop infinito, causado pelo retorno da fase de validação porque as regras de decisão são fixas; a falta de dados, onde o catálogo dispõe de menos exercícios do que o intervalo de possibilidades; e a ambiguidade no empate de notas, que pode levar a escolhas erradas entre dificuldades iguais. Para mitigação, pode ser utilizada uma engenharia de prompt robusta para evitar alucinações, implementaremos contadores de recorrência no nó LessonState para evitar loops no grafo e aplicaremos uma lógica de seleção e critérios claros de desempate que adapta a entrega à disponibilidade real do catálogo de exercícios.

6. **Avaliação**: Como você avaliaria a solução?
   Resposta:
   Pode ser avaliada a solução através de métricas técnicas e pedagógicas. Tecnicamente, pode ser medito a acurácia do planejamento e treino, garantindo que o sistema identifique corretamente a maior lacuna do aluno, e a taxa de validação, para assegurar que os planos respeitem a "policy.json". A qualidade da IA será avaliada pela clareza e correção das explicações geradas. Por fim, o sucesso real será medido por uma comparação no desempenho do aluno antes e depois de seguir o roteiro automatizado, validando se a personalização em escala de fato gera resultados educacionais.

7. **Próximos passos**: O que faria antes de produção, o que deixaria para depois e o que não faria agora.
   Resposta:
   Como prioridade antes da produção, eu focaria em ajustar o conteúdo. Resolveria a falta de exercícios no catálogo para que o sistema não tente entregar algo que não existe e refinaria os prompts para evitar explicações erradas da IA. Deixaria para um segundo momento a criação de uma interface visual e o histórico de evolução do aluno. Não investiria tempo agora em treinar modelos próprios de IA focando em deixar a lógica de planejamento automático e a entrega de exercícios 100% confiáveis.