🎮 Engenharia de Requisitos

Por que não começamos programando?

🔥 Provocação Inicial

“Desenvolva um Jogo da Velha.”

- Não dei regras.
- Não dei requisitos.
- Não expliquei o contexto.
- Não detalhei expectativas.

👉 O que você faz?

🤓 A reação natural do desenvolvedor

A maioria dos alunos:

- Abre o VS Code
- Cria uma matriz 3×3
- Implementa turno X e O
- Faz verificação de vitória
- Monta uma interface básica
- Usa o **próprio entendimento** do jogo
- Gera o código usando a IA!

⚠️ E então surge um problema...

Pergunto ao aluno:

“Como você sabe se o cliente vai aceitar a sua solução?”

As respostas normalmente são:

❌ “Não sei.”
❌ “Acho que sim.”
❌ “O jogo funciona, então está certo.”

🧠 A grande reflexão

Como você sabe se o cliente queria:

- Jogo no navegador, celular ou terminal?
- Modo Jogador vs Jogador?
- Modo contra o Computador?
- IA fácil, média ou difícil?
- Registrar pontuação?
- Sons e animações?
- Cores, tema, acessibilidade?
- Botão de reiniciar?
- Evitar jogadas inválidas?

👉 Como você sabe?

🎯 O aluno descobre sozinho:

Sem requisitos...

- Você faz suposições
- Cada desenvolvedor entrega algo diferente
- Não há critérios de aceitação
- Não há limites de escopo
- Não há como testar corretamente
- O cliente talvez rejeite a solução

😮 Lição poderosa

Programar primeiro parece rápido...

...mas quase sempre leva a retrabalho.

💡 Pergunta-chave da Engenharia de Requisitos

“Como você sabe se o cliente vai aceitar a sua solução?”

✓ Resposta:

O cliente só aceita quando a solução atende aos requisitos acordados.

Sem requisitos:

❌ Não há acordo
❌ Não há clareza
❌ Não há garantia de aceitação

🟦 Agora sim: A Teoria

Engenharia de Requisitos

🟩 02.01 – Elicitação de Requisitos

Processo de **descobrir** necessidades, expectativas e restrições.

Técnicas:

- Entrevistas
- Questionários
- Observação
- Brainstorming
- Workshops
- Protótipos

Objetivo:

👉 Saber exatamente o que o cliente quer.

📙 02.02 – Análise de Requisitos

Processo de **refinar, organizar e validar** o que foi levantado.

Atividades:

- Classificação (RF, RNF, regras)
- Priorização (MoSCoW)
- Identificação de conflitos
- Verificação e validação
- Modelagem (UML, BPMN, domínio)

🟦 02.03 – Gerência de Requisitos

Garantir que os requisitos permaneçam **claros, rastreáveis e controlados**.

Inclui:

- Controle de mudanças
- Rastreabilidade
- Versionamento
- Revisões e auditorias

🟧 02.04 – Modelos e Especificação

Modelos usados para representar requisitos:

- Diagramas UML
- Modelo CRC
- Documento SRS/ERS
- Protótipos
- Casos de Uso

Uma boa especificação deve ser:

✓ Clara
✓ Completa
✓ Consistente
✓ Verificável

🎮 Aplicação da Teoria

Engenharia de Requisitos no Jogo da Velha

📝 Elicitação do Jogo da Velha

Técnicas usadas:

- Entrevista com cliente
- Observação de usuários
- Brainstorming
- Protótipo de baixa fidelidade

Descobertas:

- Jogo deve rodar no navegador
- Jogador vs Jogador
- Jogador vs Computador
- IA simples

🔧 Requisitos Funcionais (RF)

RF01: Iniciar um novo jogo com tabuleiro vazio

RF02: Jogador vs Jogador

RF03: Jogador vs Computador

RF04: Registrar e exibir jogada atual

RF05: Verificar vitória

RF06: Verificar empate

RF07: Exibir mensagens de resultado

RF08: Reiniciar jogo

🔒 Requisitos Não Funcionais (RNF)

RNF01 – Usabilidade: interface simples e clara

RNF02 – Performance: resposta imediata

RNF03 – Portabilidade: rodar em navegadores modernos

RNF04 – Confiabilidade: impedir jogadas inválidas

RNF05 – Manutenibilidade: facilitar evolução da IA

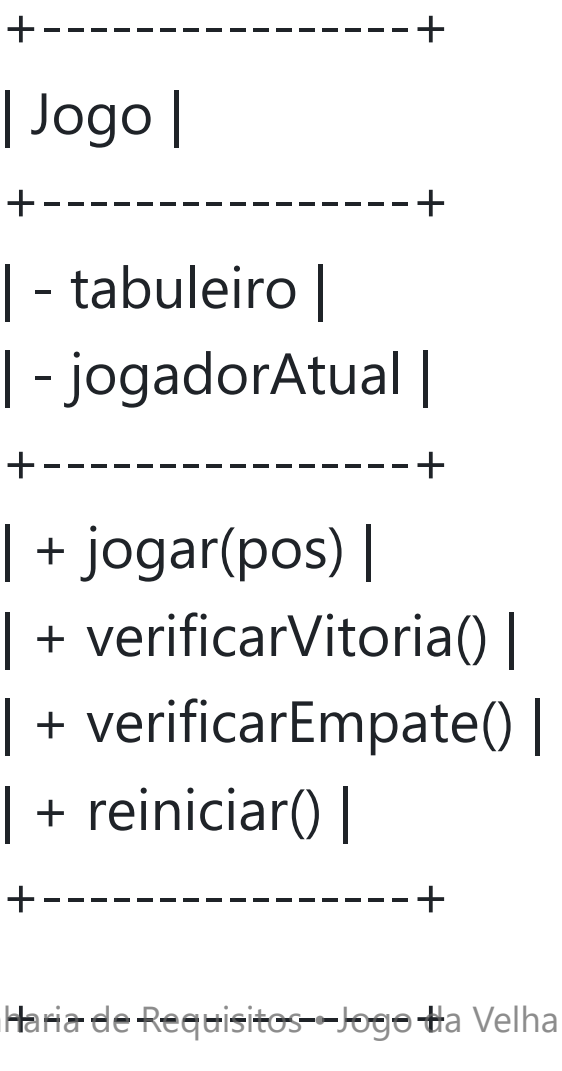
Regras de Negócio (RN)

RN01: Jogador 1 é “X”; jogador 2/IA é “O”

RN02: “X” sempre começa

RN03: Após vitória/empate, jogadas são bloqueadas

🧱 Diagrama de Classes (Simplificado)



🪪 Modelo CRC

Classe: Jogo

- Responsabilidades: controlar turnos, validar jogadas, verificar fim da partida
- Colabora com: Tabuleiro, Jogador, IA

Classe: Tabuleiro

- Responsabilidade: armazenar estado do jogo
- Colabora com: Jogo

Classe: Jogador

- Responsabilidade: fornecer símbolo
- Colabora com: Jogo

Classe: IA

Especificação SRS (Resumo)

1. Introdução

Objetivo: definir requisitos do sistema Jogo da Velha.

Escopo: jogo simples com PvP e PvE.

2. Descrição Geral

Usuários: crianças/iniciantes.

Plataforma: navegador web.

3. RF – conforme listados

4. RNF – conforme listados

5. Regras de Negócio

6. Modelos (UML + CRC)

7. Critérios de Aceitação

🧪 Quiz – Engenharia de Requisitos

Vamos ver o que você aprendeu!

Responda as perguntas nos próximos slides.

❓ Pergunta 1

Por que não devemos começar um projeto de software diretamente pela programação?

A) Porque programar é difícil
B) Porque o cliente nunca sabe o que quer
C) Porque sem requisitos não sabemos o que entregar
D) Porque diagramas são mais importantes que código

? Pergunta 2

Qual é a pergunta-chave que revela a necessidade de requisitos?

A) “Qual linguagem você vai usar?”
B) “Quanto tempo você vai levar para programar?”
C) “Como você sabe se o cliente vai aceitar a sua solução?”
D) “Você prefere front-end ou back-end?”

❓ Pergunta 3

Qual dos itens abaixo é um **Requisito Funcional** do Jogo da Velha?

A) O sistema deve responder rapidamente às jogadas
B) Deve permitir que dois jogadores joguem alternadamente
C) A interface deve ser amigável para crianças
D) O jogo deve rodar no navegador

❓ Pergunta 4

Qual técnica NÃO é usada na elicitação de requisitos?

A) Entrevista

B) Brainstorming

C) Observação

D) Compilação de código

❓ Pergunta 5

O que representa o Modelo **CRC**?

A) Classe, Relacionamento e Código

B) Classe, Responsabilidade e Colaboração

C) Código, Regras e Casos de Uso

D) Cliente, Resultado e Critérios

📝 Gabarito do Quiz

Pergunta 1: C

Pergunta 2: C

Pergunta 3: B

Pergunta 4: D

Pergunta 5: B