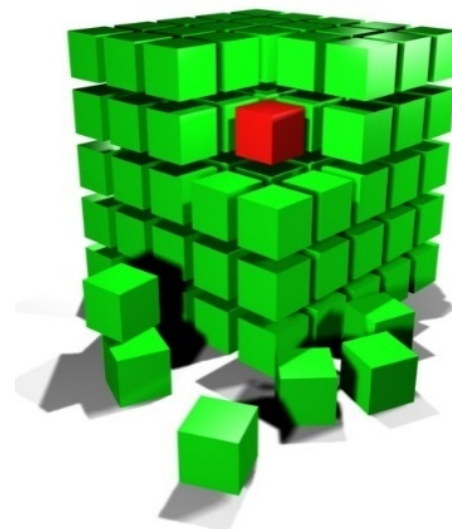




■ Aula 06

- Introdução
- Modelo geral do processo de projeto
- Requisitos de sistema e suas características
- Requisitos funcionais
- Requisitos não funcionais
- Regras de negócio
- Levantamento de informações
- Gerenciamento de requisitos





■ Introdução

- A descrição dos requisitos de um sistema pode ser uma tarefa difícil. Geralmente, essa tarefa exige que o analista possua os seguintes conhecimentos:
 1. Conhecimentos sobre o negócio;
 2. Conhecimento das tarefas e operações que serão realizadas pelo sistema;
 3. Conhecimento das necessidades dos usuários;
 4. Conhecimento das informações sobre o contexto do negócio;
 5. Conhecimento para identificar problemas técnicos.





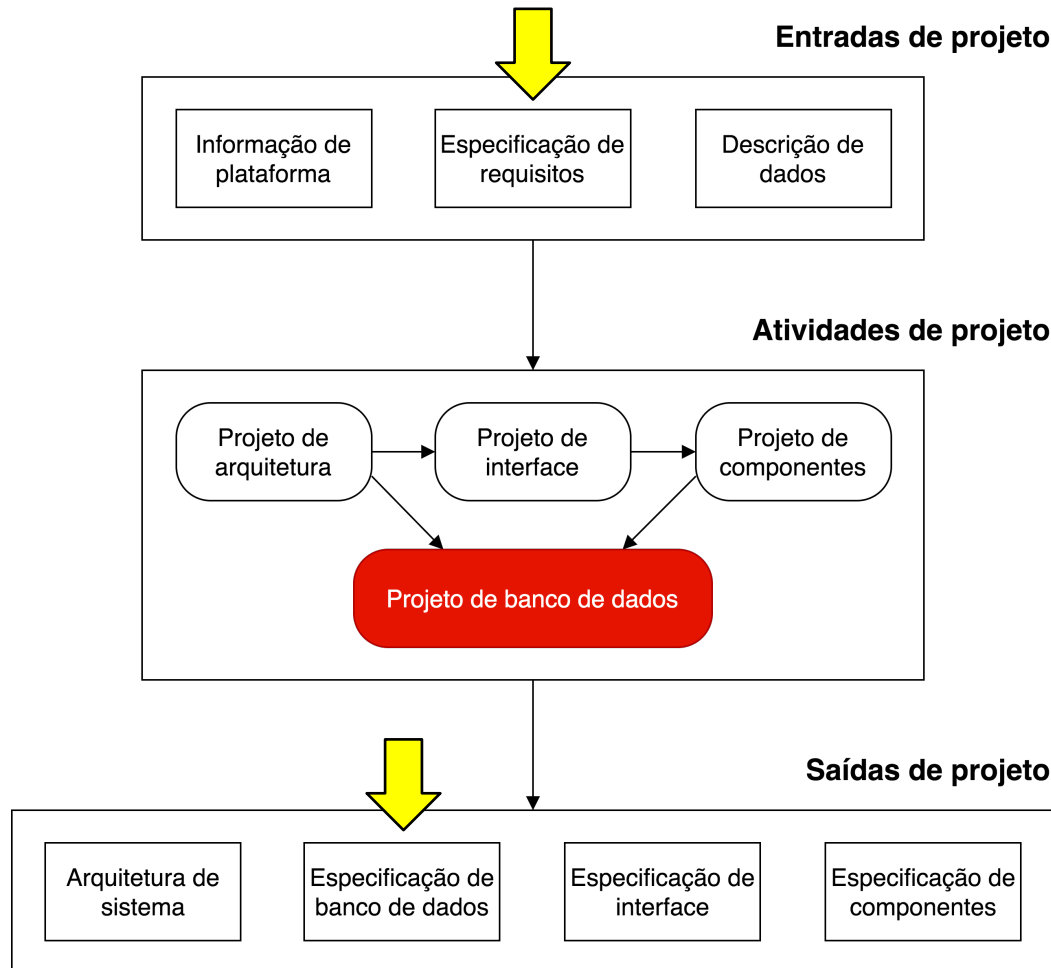
■ Introdução

- Entender os requisitos do sistema é importante para o sucesso de um projeto.
- O **banco de dados** representa uma parte do sistema, sendo que ele será responsável pelo armazenamento e recuperação das informações.
- Dessa forma, o **projeto de banco de dados** deve estar alinhado ao **projeto de desenvolvimento do sistema**, permitindo o armazenamento e a segurança dos dados, conforme os requisitos que foram coletados durante a fase de **Análise de Requisitos**.





■ Modelo geral do processo de projeto





■ Requisitos de sistema

- O Instituto de Eletricistas, Eletrônicos e Engenheiros (IEEE), define requisito como: *“uma condição ou capacidade requerida por um usuário, para resolver um problema ou atingir um objetivo, que deve ser atingida ou possuída por um sistema, ou partes de um sistema, para satisfazer um contrato, padrão, especificação ou outros documentos”*.
- Na prática, requisito é um processo que envolve o estudo das necessidades do usuário para encontrar a definição correta do sistema, determinando assim, o **sucesso** ou o **fracasso** de um projeto.





■ Requisitos de sistema

- Segundo Sommerville¹, os requisitos são descrições dos serviços fornecidos pelo sistema e suas restrições operacionais, sendo que os processos de aquisição, refinamento e verificação do sistema descrevem o que o sistema deve fazer, mas não a forma como será implementado.
- 1. **SOMMERVILLE, Ian.** *Engenharia de Software*. 10ª ed. São Paulo: Pearson Universidades, 2019.





■ Requisitos de sistema

- Sommerville classifica os requisitos da seguinte maneira:
 - **Requisitos funcionais:** representam as declarações dos serviços que o sistema deve fornecer, do modo como o sistema deve reagir a determinadas entradas e de como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem declarar explicitamente o que o sistema não deve fazer.
 - **Requisitos não funcionais:** representam as restrições sobre os serviços ou funções oferecidas pelo sistema. Eles incluem restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões. Os requisitos não funcionais se aplicam, frequentemente, ao sistema como um todo, em vez de às características individuais ou aos serviços.





- **Características importantes de um requisito**
 1. **Clareza**: os requisitos devem ser expressos de maneira clara e fácil de entender, evitando ambiguidades ou linguagem que possa levar a interpretações diferentes.
 2. **Especificidade**: os requisitos devem ser específicos, detalhados e concretos. Eles devem responder às perguntas do que, como, quando e por que uma determinada funcionalidade ou restrição é necessária.
 3. **Não ambiguidade**: deve-se evitar qualquer tipo de ambiguidade nos requisitos. Os termos e conceitos devem ser definidos com precisão, de forma que não haja espaço para múltiplas interpretações.





- **Características importantes de um requisito**
 4. **Mensurabilidade**: os requisitos devem ser formulados de maneira que, caso necessário, possam ser medidos ou verificados.
 5. **Relevância**: cada requisito deve ser relevante para o objetivo geral do projeto. Requisitos irrelevantes podem adicionar uma complexidade desnecessária.
 6. **Consistência**: os requisitos devem ser consistentes entre si e com outros documentos relacionados ao projeto do sistema.
 7. **Rastreabilidade**: cada requisito deve ser rastreável a uma fonte ou necessidade específica. Isso ajuda a entender por que ele foi incluído e facilita possíveis revisões.





- **Características importantes de um requisito**
 8. **Priorização**: os requisitos devem ser priorizados de acordo com sua importância e impacto no projeto. Isso ajuda a equipe a concentrar seus esforços nas funcionalidades consideradas mais críticas.
 9. **Testabilidade**: os requisitos devem ser formulados de maneira que seja possível criar casos ou cenários de teste, para verificar sua implementação.
 10. **Validação**: os requisitos devem ser revisados e validados pelas partes interessadas, tais como usuários finais e clientes, para garantir que eles atendam suas necessidades e expectativas.





- **Características importantes de um requisito**
 11. **Documentação adequada:** os requisitos devem ser documentados de forma clara e organizada, muitas vezes utilizando uma linguagem padronizada e uma estrutura para documentação.
 12. **Flexibilidade:** os requisitos devem ser flexíveis o suficiente para permitir ajustes à medida que as necessidades do projeto mudam. No entanto, essas mudanças devem ser gerenciadas de forma controlada.
 13. **Aprovação:** os requisitos devem ser aprovados por todas as partes interessadas, para garantir que todos concordem com o que está sendo desenvolvido.





- **Características importantes de um requisito**
 - 14. **Conformidade com padrões e práticas recomendadas:** os requisitos devem seguir as melhores práticas da indústria e os padrões relevantes, quando aplicável.
 - 15. **Versionamento:** é importante manter um controle de versão dos requisitos para rastrear as mudanças ao longo do tempo e manter um histórico claro das decisões relacionadas ao projeto do sistema.





■ Requisitos funcionais

- Os **requisitos funcionais** são utilizados para descrever as ações que um sistema de *software* deve realizar. Eles dependem da natureza do sistema a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos.
- Quando expressos como requisitos de usuário, os requisitos funcionais geralmente são apresentados de maneira mais abstrata, para serem compreendidos pelos usuários do sistema.
- No entanto, requisitos funcionais mais detalhados oferecem uma descrição minuciosa das operações do sistema, incluindo suas entradas, saídas, tratamento de exceções, etc.





- **Exemplos de requisitos funcionais**
 - **Sistema de Reservas de Hotel**
 1. O sistema deve permitir que os clientes façam reservas de quartos de hotel.
 2. Deve ser possível cancelar uma reserva existente.
 3. O sistema deve calcular automaticamente o custo da estadia com base no tipo de quarto e nas datas selecionadas.





- **Exemplos de requisitos funcionais**
 - **Sistema de Gerenciamento de Biblioteca**
 1. Os usuários devem ser capazes de pesquisar e localizar livros no catálogo.
 2. O sistema deve permitir que os funcionários registrem a entrada e saída de livros.
 3. Deve ser possível renovar empréstimos de livros, desde que não haja reservas pendentes.





- **Exemplos de requisitos funcionais**
 - **Aplicativo de Rede Social**
 1. Os usuários devem poder criar perfis de usuário.
 2. Deve haver um sistema de autenticação seguro para proteger as contas dos usuários.
 3. Os usuários devem ser capazes de publicar mensagens, fotos e vídeos em seus perfis e interagir com as postagens de outros usuários.



- Exemplo de especificação de um requisito funcional

Identificador	RF001		
Nome	Cadastrar usuário		
Módulo	Cadastro		
Data de criação	10/09/2023	Autor	Paulo Giovani
Data da última alteração	10/09/2023	Autor	Paulo Giovani
Versão	1	Prioridade	Essencial
Descrição	<p>Quando o usuário realizar o primeiro acesso ao aplicativo de Rede Social, será necessário realizar o cadastro do seu nome completo, nome de usuário, e-mail, senha e telefone celular.</p> <p>Também existe a possibilidade do usuário se cadastrar utilizando a sua conta do Facebook ou do Google.</p> <p>Caso o usuário já possua cadastro, poderá acessar o aplicativo por meio da tela de login.</p>		





■ Requisitos não funcionais

- Os **requisitos não funcionais** não se referem diretamente às funções específicas que um sistema oferece aos seus usuários.
- Eles abordam características que afetam a qualidade e o desempenho do sistema como um todo, tais como confiabilidade, tempo de resposta e eficiência de uso de recursos, etc.
- Os requisitos não funcionais podem impor limitações sobre como o sistema deve ser implementado, como por exemplo, a definição da plataforma onde o sistema deverá ser executado, seus requisitos de *hardware*, sua tolerância a falhas, entre outras.





■ Requisitos não funcionais

- Os requisitos não funcionais, como desempenho, proteção ou disponibilidade, normalmente especificam ou restringem as características do sistema como um todo.
- Requisitos não funcionais são frequentemente mais críticos que requisitos funcionais individuais. Os usuários do sistema podem, geralmente, encontrar maneiras de contornar uma função do sistema que realmente não atenda a suas necessidades. No entanto, deixar de atender a um requisito não funcional pode significar a inutilização de todo o sistema.
- Por exemplo, se um sistema de aeronaves não cumprir seus requisitos de confiabilidade, não será certificado como um sistema seguro para operar.





- **Exemplos de requisitos não funcionais**
 - Para um **aplicativo de rede social**, os requisitos não funcionais desempenham um papel crítico na determinação de sua usabilidade, segurança e eficiência.
 - Vamos conferir alguns exemplos de requisitos não funcionais que desempenham um papel fundamental para assegurar que um aplicativo dessa categoria seja capaz de satisfazer as necessidades dos usuários, ao mesmo tempo em que mantém níveis adequados de segurança, confiabilidade, eficiência e aderência aos regulamentos e padrões de desenvolvimento.





- **RNFs para aplicativo de Rede Social**
 - **Desempenho**
 1. O aplicativo deve carregar o *feed de notícias* dos usuários em menos de 2 segundos.
 2. A capacidade de usuários simultâneos deve ser escalável para acomodar um aumento de 100% no tráfego durante um evento de pico.





- **RNFs para aplicativo de Rede Social**
 - **Segurança**
 1. Os dados dos usuários, incluindo informações de perfil e mensagens privadas, devem ser criptografados em repouso e em trânsito.
 2. O aplicativo deve possuir medidas eficazes de proteção contra ataques de negação de serviço (*Distributed Denial of Service* - DDoS) e tentativas de invasão.





- **RNFs para aplicativo de Rede Social**
 - **Disponibilidade**
 1. O aplicativo deve estar disponível e funcional 99,9% do tempo, excluindo manutenção programada.
 2. A recuperação após falhas deve ocorrer em menos de 15 minutos em caso de interrupção não planejada.





- **RNFs para aplicativo de Rede Social**
 - **Usabilidade**
 1. A interface do usuário deve ser intuitiva e seguir as diretrizes de usabilidade, com tempos mínimos de aprendizado para novos usuários.
 2. O aplicativo deve ser acessível para pessoas com deficiências, em conformidade com as diretrizes de acessibilidade WCAG (*Web Content Accessibility Guidelines*).





- **RNFs para aplicativo de Rede Social**
 - **Conformidade com padrões e regulamentos**
 1. O aplicativo deve estar em conformidade com as leis de proteção de dados, como a **Lei Geral de Proteção de Dados Pessoais** (LGPD).
 2. O aplicativo deve seguir padrões de privacidade reconhecidos e atualizados, além de estar em conformidade com regulamentos adequados para a proteção infantil.





- **RNFs para aplicativo de Rede Social**
 - **Interoperabilidade**
 1. O aplicativo deve ser capaz de se integrar com outras redes sociais, tais como *Facebook* e *Instagram*, permitindo o compartilhamento de conteúdos.
 2. O aplicativo deve ser compatível com diferentes sistemas operacionais móveis e navegadores da web.





- **RNFs para aplicativo de Rede Social**
 - **Manutenibilidade**
 1. O código-fonte do aplicativo deve ser bem documentado e modular para facilitar a manutenção e atualizações futuras.
 2. As correções de problemas críticos devem ser implementadas e distribuídas em um prazo máximo de 48 horas após a identificação.





- **RNFs para aplicativo de Rede Social**
 - **Eficiência de recursos**
 1. O aplicativo deve ser otimizado para poder utilizar de maneira eficiente os recursos disponíveis, minimizando o consumo de bateria em dispositivos móveis e a largura de banda de rede.





- RNFs para aplicativo de Rede Social
 - Tolerância a falhas
 1. O sistema deve ser projetado para continuar funcionando mesmo em caso de falhas de servidores ou serviços externos.
 2. Deve-se realizar *backups* regulares dos dados do usuário para evitar perda de informações críticas.





- **RNFs para aplicativo de Rede Social**
 - **Escalabilidade**
 1. O aplicativo deve ser capaz de lidar com um aumento significativo no número de usuários ativos.



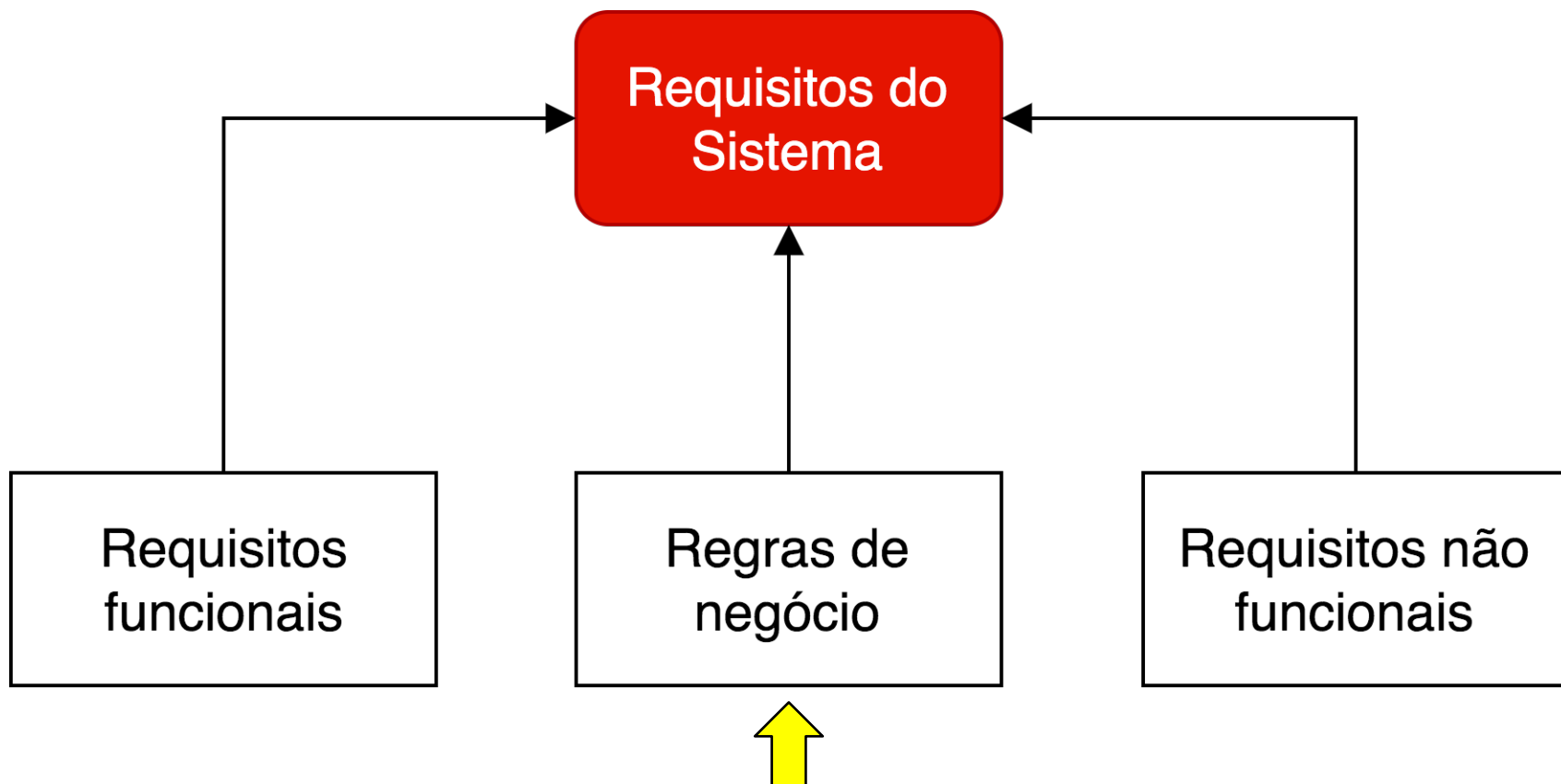


- **RNFs para aplicativo de Rede Social**
 - **Legal e regulamentação**
 1. O aplicativo deve cumprir as políticas de uso, diretrizes de conteúdo e regulamentos de segurança cibernética.
 2. O aplicativo deve permitir uma resposta rápida para ocorrências relacionadas às denúncias de violações de conteúdo.





■ Requisitos de sistema





■ Regras de negócio

- Em **Engenharia de Software**, as regras de negócio são premissas e restrições aplicadas a uma operação comercial de uma empresa, que precisam ser atendidas para que o negócio funcione da maneira esperada.
- As regras de negócio definem como o sistema poderá atender às necessidades e exigências que foram definidas, permitindo a implementação dos requisitos funcionais.





- **Regras de negócio**
 - Em um projeto de banco de dados, o modelo de dados somente adquire significado quando representa de maneira adequada as definições existentes nas regras de negócio.
 - **Regra de negócio**: descrição breve, precisa e sem ambiguidades, de uma política, um procedimento ou um princípio, utilizado em uma determinada organização.



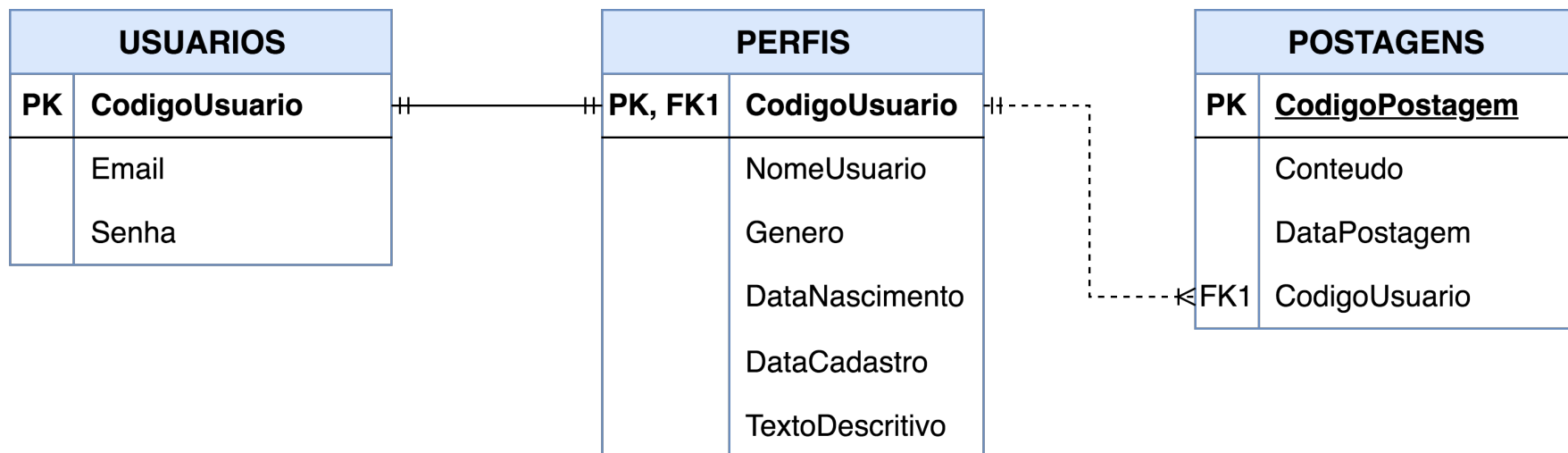


- **Exemplos de regras de negócio**
 - Um **usuário** deve possuir um único **perfil** e cada perfil deve conter o identificador, nome, gênero, data de nascimento, data de cadastro e texto descritivo do usuário. Para cada **usuário** devemos armazenar seu identificador, e-mail e senha.
 - Cada **perfil** pode realizar diversas **postagens** e cada postagem é realizada por um único perfil. Para cada **postagem**, precisamos armazenar seu identificador, conteúdo, data da postagem e a informação de quem realizou a postagem.





▪ Exemplo de DER baseado nas regras de negócio

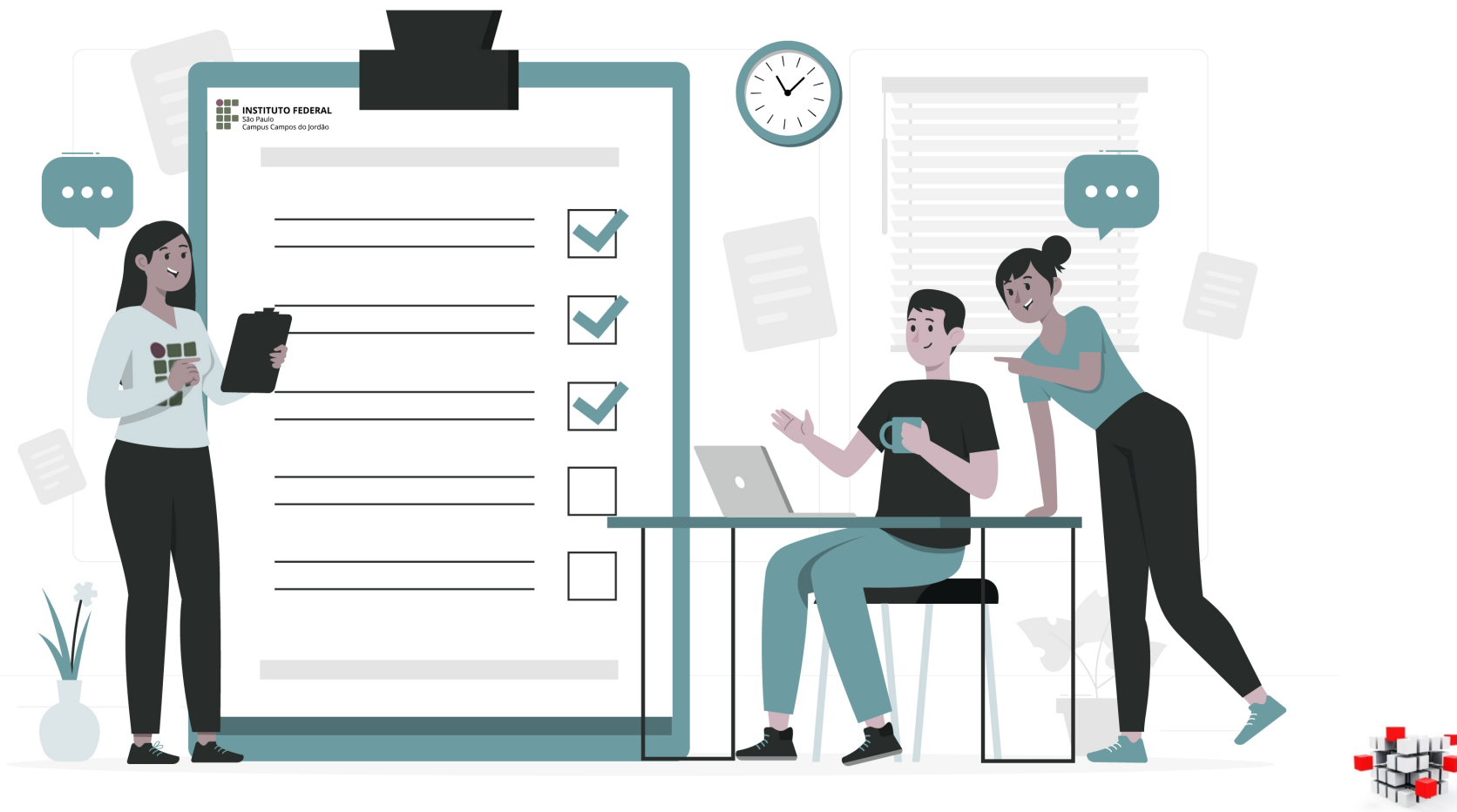


Exemplo de DER para representar as regras de negócio de uma aplicação de rede social. Em uma situação real, é necessário controlar o armazenamento de imagens, contadores de likes, comentários, mensagens privadas, entre outros recursos.





■ Principais fontes de regras de negócio





■ Levantamento de informações

- O propósito fundamental do processo de coleta de informações reside na análise minuciosa do problema, o que inclui a identificação das necessidades do usuário e a avaliação de sua situação atual.
- O **objetivo principal** é alcançar uma compreensão abrangente de todos os requisitos exigidos, permitindo, assim, a definição do escopo do projeto de sistemas que se planeja desenvolver.
- Por meio deste levantamento, também pretendemos obter uma percepção sobre como os usuários executam suas tarefas, assim como verificar o conhecimento que possuem em relação ao desenvolvimento de suas atividades.





■ Levantamento de informações

- Durante o levantamento de informações, pode-se utilizar notações que formam um sistema convencionado de representações. Essas representações possuem as seguintes características:
 - **Legibilidade**: todos os envolvidos no projeto conseguem facilmente entender o problema.
 - **Completa**: permite o crescimento e acréscimo gradativo das informações reunidas sobre o projeto.
 - **Facilidade de alteração**: utiliza ferramentas gráficas para auxiliar na manutenção das informações, tornando a documentação do projeto sempre atualizada.





■ Levantamento de informações

- A notação utilizada no levantamento das informações tem como finalidade eliminar ambiguidades que possam surgir durante o diálogo com os usuários.
- Quando o analista perceber a presença dessa situação, deve explorar a utilização de sinônimos, de forma a tornar mais claro o entendimento do problema.
- Como os requisitos são as principais entradas para iniciar um projeto de sistema, assim como desenvolver seu banco de dados, é importante evitar a presença de dúvidas decorrentes do entendimento do seu escopo.





■ Levantamento de informações

- As técnicas de levantamento de informações, no contexto do desenvolvimento de sistemas de banco de dados, possuem as seguintes finalidades:
 1. Definir os objetivos do projeto;
 2. Definir o escopo do projeto;
 3. Levantar os requisitos funcionais e não funcionais, que deverão ser atendidos pelo sistema;
 4. Obter os modelos preliminares de relacionados ao projeto do sistema, incluindo o modelo da aplicação e o **modelo de dados**.





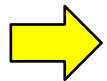
- **Levantamento de informações**
 - O **modelo da aplicação** é uma parte fundamental do processo de projeto e desenvolvimento de sistemas.
 - Ele é utilizado como uma representação abstrata, que descreve como a aplicação funcionará em termos de estrutura de dados, lógica de negócios, interface do usuário e arquitetura geral.
 - O modelo da aplicação serve como um guia para os desenvolvedores durante todo o ciclo utilizado para o desenvolvimento do *software*, desde o projeto até as fases de implementação e testes.





■ Levantamento de informações

- O modelo da aplicação pode conter os seguintes itens:




- **Estruturas de dados:** descreve como os dados serão organizados e armazenados pela aplicação. Isso pode incluir a definição de bancos de dados, tabelas, campos e relacionamentos entre eles.
- **Fluxo de controle:** define como o controle do programa é transferido entre diferentes partes da aplicação. Isso envolve a especificação de quais ações serão executadas em resposta a eventos ou ações do usuário.
- **Interfaces de usuário:** descreve como a interface de usuário da aplicação será projetada, incluindo *layouts* de tela, elementos de interface, interações e fluxos de usuário.





■ Levantamento de informações

- 
- **Regras de negócio:** especifica as regras e lógica que governam o comportamento da aplicação, como validações de entrada, cálculos, processamento de dados e tomada de decisões.
 - **Arquitetura:** define a arquitetura geral da aplicação, incluindo a escolha de tecnologias, componentes de *software*, padrões de *design* e como os diferentes módulos ou componentes interagem entre si.
 - **Diagramas e documentação:** O modelo da aplicação muitas vezes é documentado com diagramas, descrições textuais e outras formas de documentação que ajudam a comunicar a visão da aplicação para toda a equipe de desenvolvimento.





■ Levantamento de informações

- Para garantir a eficácia do processo de levantamento de informações sobre os requisitos, é necessário compreender plenamente o perfil do usuário que irá interagir com o sistema. Portanto, manter uma comunicação contínua com o usuário é de suma importância.
- Esse nível de entendimento terá um impacto direto no modo como o sistema se comportará diante do usuário, ao mesmo tempo em que possibilitará uma preparação mais eficaz para a coleta inicial de informações relativas ao sistema em desenvolvimento ou em processo de modificação.





■ Levantamento de informações

- Segundo Pressman², o processo de levantamento de informações é cíclico e compreende quatro fases:
 1. Entendimento do domínio;
 2. Extração e análise de requisitos;
 3. Especificação dos requisitos;
 4. Validação dos requisitos.
- 2. **PRESSMAN, R. S.; MAXIM, B. R.** *Engenharia de Software: uma abordagem profissional*. 9ª ed. Porto Alegre: AMGH, 2021.





■ Levantamento de informações

- Existem diversas técnicas que podem ser utilizadas para realizar o levantamento de informações:
 1. Reunião
 2. Entrevista
 3. JAD (*Joint Application Development*)
 4. Questionário
 5. Observação visual
 6. *Brainstorming*
 7. IDEF (*Integrated Definition Methods*)
 8. Cartões CRC (*Class, Responsibility, Collaboration*)





■ Reunião

- A **técnica de reunião** é uma abordagem essencial no processo de levantamento de requisitos no desenvolvimento de *software* e em outros projetos. Ela desempenha um papel fundamental na coleta de informações, na compreensão das necessidades dos clientes e usuários, e na definição clara dos requisitos do sistema ou projeto.
- As reuniões permitem uma comunicação direta entre as partes interessadas, incluindo clientes, usuários finais, gerentes de projeto e membros da equipe de desenvolvimento. Isso ajuda a esclarecer dúvidas, resolver mal-entendidos e garantir que todos tenham uma compreensão comum do que está sendo solicitado ou planejado.
- Nas reuniões, as partes interessadas podem compartilhar detalhes específicos sobre suas necessidades, desafios e expectativas. Isso é crucial para coletar informações precisas que servirão como base para a definição dos requisitos do projeto.





■ Reunião

- Durante as reuniões, os participantes podem fornecer *feedback* imediato, o que ajuda a ajustar e refinar as ideias e os requisitos, à medida que eles são discutidos.
- As reuniões permitem que os analistas de requisitos façam perguntas, esclareçam dúvidas e obtenham *insights* diretamente das partes interessadas, resultando em um entendimento mais profundo das necessidades.
- As reuniões também desempenham um papel importante na construção de relacionamentos e na construção de confiança entre as equipes de projeto e as partes interessadas.





■ Reunião

- Sugestão de melhores práticas para reuniões de levantamento de requisitos:
 1. **Defina uma agenda**: antes da reunião, estabeleça uma agenda clara que descreva os tópicos a serem discutidos e os objetivos a serem alcançados.
 2. **Selecione os participantes adequados**: convide as pessoas certas para a reunião, garantindo que todas as partes interessadas estejam presentes.
 3. **Mantenha o foco e a disciplina**: durante a reunião, mantenha o foco nos tópicos definidos na agenda e evite desvios irrelevantes.
 4. **Documente minuciosamente**: designe alguém para tomar notas detalhadas durante a reunião, registrando informações cruciais, decisões tomadas e ações a serem seguidas.
 5. **Siga um cronograma**: respeite o tempo alocado para a reunião e siga um cronograma para manter o encontro eficiente.
 6. **Solicite *feedback* e confirmação**: ao final da reunião, peça aos participantes que revisem e confirmem as informações discutidas, garantindo que tudo esteja documentado corretamente.





■ Entrevista

- A **técnica de entrevista** é um componente fundamental no processo de levantamento de requisitos no desenvolvimento de *software* e em muitos outros projetos. Ela desempenha um papel crucial na obtenção de informações valiosas, compreensão das necessidades dos *stakeholders* e na definição precisa dos requisitos do sistema.
- As entrevistas possibilitam uma comunicação direta entre os analistas de requisitos e as partes interessadas. Isso ajuda a esclarecer informações, resolver ambiguidades e garantir que todos tenham uma compreensão clara dos requisitos.
- Durante as entrevistas, os analistas podem coletar informações contextuais valiosas, como cenários de uso, restrições operacionais e preferências dos usuários. Essas informações ajudam a moldar os requisitos de forma mais precisa.





■ Entrevista

- Sugestão de melhores práticas para entrevistas de levantamento de requisitos:
 1. **Prepare-se adequadamente**: antes da entrevista, familiarize-se com o contexto do projeto e os tópicos a serem abordados. Prepare um conjunto de perguntas relevantes.
 2. **Estabeleça um ambiente confortável**: crie um ambiente de entrevista confortável e aberto, onde os *stakeholders* se sintam à vontade para compartilhar informações.
 3. **Escute atentamente**: durante a entrevista, escute atentamente as respostas, fazendo perguntas de acompanhamento para obter detalhes adicionais.
 4. **Seja flexível**: esteja disposto a adaptar a abordagem de acordo com as respostas e necessidades dos entrevistados.
 5. **Documente cuidadosamente**: tome notas detalhadas durante a entrevista e, depois, transforme essas notas em documentação clara e organizada.
 6. **Realize múltiplas entrevistas**: entreviste várias partes interessadas para obter uma visão completa das necessidades e expectativas.





- **JAD (*Joint Application Development*)**
 - A **técnica JAD** (*Joint Application Development*, ou Desenvolvimento Conjunto de Aplicações), é uma abordagem colaborativa utilizada no desenvolvimento de sistemas de *software* e no processo de levantamento de requisitos. Também conhecida como **Método de Projeto Interativo**, foi idealizada em 1977 pelo consultor Chuck Morris, da IBM.
 - Essa técnica envolve a participação ativa de diversas partes interessadas, incluindo clientes, usuários finais, gerentes de projeto e membros da equipe de desenvolvimento.
 - O JAD tem visa obter, de maneira mais rápida, informações de alta qualidade dos envolvidos, por meio de sessões de trabalho estruturadas que buscam chegar a decisões consensuais para definir os requisitos do sistema.





■ JAD (*Joint Application Development*)

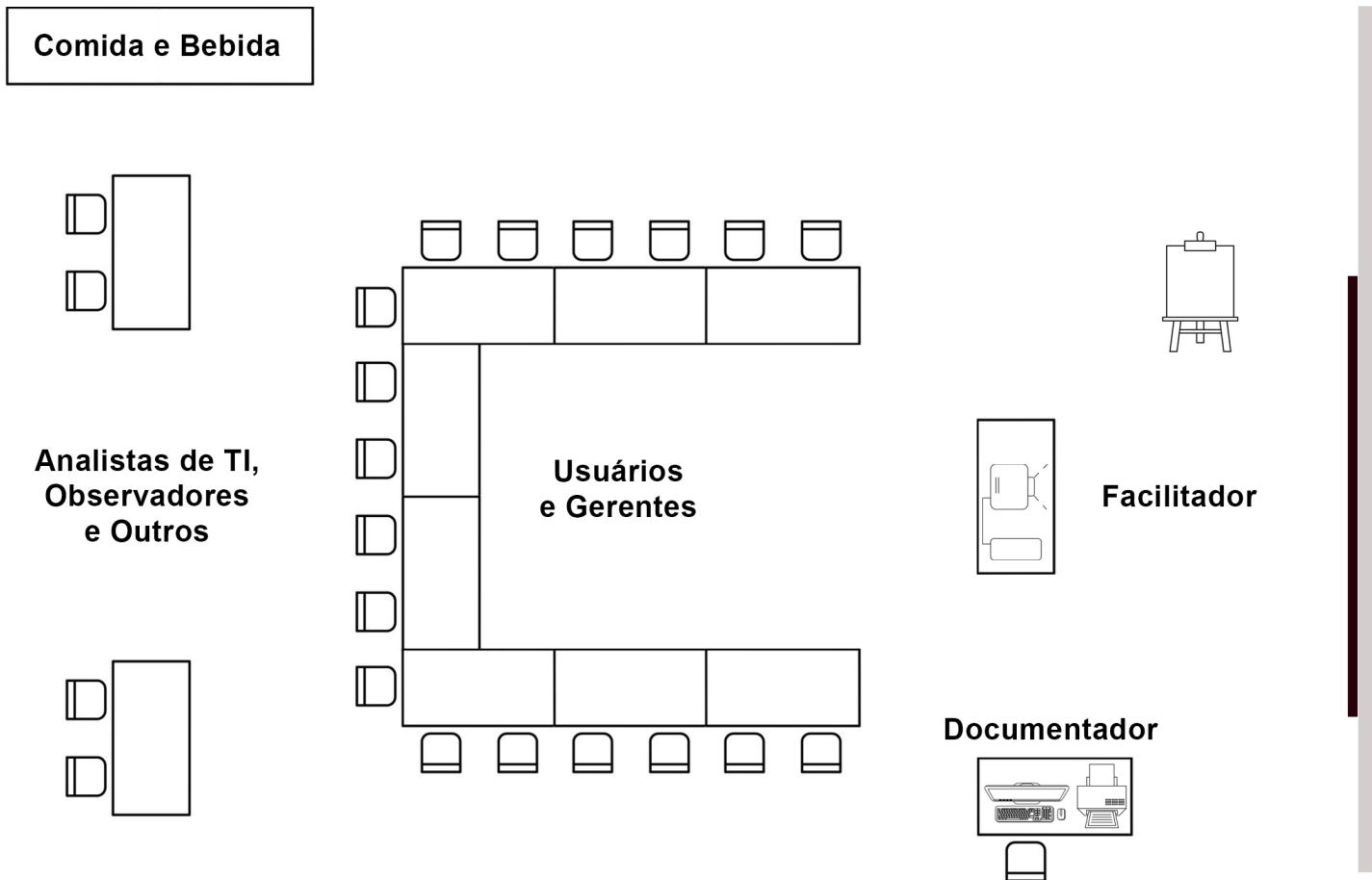
■ Principais características do JAD:

1. **Colaboração intensiva:** o JAD promove a colaboração intensiva entre as partes interessadas, reunindo-as em sessões de trabalho presenciais ou virtuais para discutir, analisar e documentar os requisitos do sistema. Essa colaboração ajuda a garantir que todas as perspectivas sejam consideradas.
2. **Facilitador ou moderador:** uma figura chamada de facilitador ou moderador geralmente lidera as sessões de JAD. Esse facilitador é responsável por manter o foco, gerenciar o tempo e garantir que a discussão seja produtiva.
3. **Metodologia estruturada:** o JAD segue uma metodologia estruturada, com uma agenda definida e objetivos claros para cada sessão. As sessões são frequentemente divididas em partes, como coleta de informações, análise de requisitos, identificação de prioridades e tomada de decisões.
4. **Feedback em tempo real:** durante as sessões de JAD, os participantes podem fornecer feedback em tempo real sobre as ideias e propostas, o que permite ajustes imediatos e ajuda a evitar mal-entendidos.
5. **Documentação detalhada:** o resultado das sessões de JAD é uma documentação detalhada dos requisitos do sistema, que serve como base para o desenvolvimento posterior do *software*.





■ JAD (*Joint Application Development*)





■ Questionário

- A **técnica de questionário** é uma abordagem essencial no processo de levantamento de requisitos no desenvolvimento de *software* e em outros projetos. Essa técnica envolve a criação e a distribuição de questionários estruturados para as partes interessadas, com o objetivo de coletar informações específicas sobre suas necessidades, preferências e expectativas.
- Um questionário deve ser bem elaborado, contendo perguntas claras, abertas e fechadas, encadeadas e complementares.
- Os benefícios da aplicação dessa técnica incluem a eficiência, a economia de tempo e de recursos, o anonimato, a coleta de dados quantitativos e a facilidade de distribuição.





■ Questionário

- Sugestão de melhores práticas para elaboração de questionários:
 1. **Projete perguntas relevantes:** elabore perguntas claras, relevantes e específicas que estejam alinhadas com os objetivos do levantamento de requisitos.
 2. **Ofereça opções de resposta:** forneça opções de resposta, como escolha múltipla ou escalas de classificação, para facilitar o processo de resposta dos participantes.
 3. **Teste e valide:** antes de distribuir o questionário em larga escala, teste-o com um grupo piloto para identificar quaisquer problemas ou ambiguidades nas perguntas.
 4. **Estabeleça um prazo:** defina um prazo claro para a devolução dos questionários e siga-o rigorosamente.
 5. **Analise os resultados:** após a coleta de dados, analise cuidadosamente as respostas dos questionários e identifique tendências, padrões e áreas de preocupação.
 6. **Combinação com outras técnicas:** considere combinar a técnica de questionário com outras, como entrevistas, para obter uma compreensão mais completa das necessidades.





■ Observação visual

- A **técnica de observação visual** é uma abordagem poderosa no processo de levantamento de requisitos, especialmente quando se trata de entender como as pessoas interagem com sistemas, processos ou ambientes específicos.
- Essa técnica envolve a observação direta e sistemática de usuários e partes interessadas em suas atividades cotidianas para identificar necessidades, desafios e oportunidades de melhoria.
- Além da compreensão contextual, essa técnica também permite que o analista possa realizar a identificação de comportamentos e a validação das necessidades reais dos usuários.





■ ***Brainstorming***

- A **técnica de *brainstorming***, também conhecida como **tempestade de ideias**, é uma abordagem criativa e colaborativa frequentemente utilizada no processo de levantamento de requisitos.
- Essa técnica tem como objetivo gerar um grande número de ideias, sugestões e conceitos relacionados a um projeto ou problema específico. Embora muitas vezes associada à geração de soluções, o *brainstorming* também pode ser aplicado com eficácia na identificação de requisitos essenciais.
- A reunião de *brainstorming* deve ser descontraída. Os participantes devem procurar enriquecer as ideias alheias, evitando discussões desnecessárias.





■ **Brainstorming**

- Sugestão de melhores práticas para utilizar a técnica de *brainstorming*:
 1. **Estabeleça um objetivo**: antes de começar, defina claramente o objetivo do *brainstorming*, como a identificação de requisitos específicos ou a geração de ideias para um determinado aspecto do projeto.
 2. **Defina um líder ou moderador**: nomeie um líder ou moderador para facilitar a sessão de *brainstorming* e garantir que as regras sejam seguidas.
 3. **Estabeleça regras básicas**: estabeleça algumas regras básicas, como adiar julgamentos críticos, incentivar a livre expressão de ideias e evitar interrupções.
 4. **Incentive a participação de todos**: encoraje todas as partes interessadas a contribuírem, independentemente do cargo ou experiência. Incentive a construção de ideias uns sobre os outros.
 5. **Registre todas as ideias**: registre todas as ideias geradas, mesmo que pareçam inicialmente irrelevantes. Às vezes, ideias aparentemente “fora da caixa” podem levar a soluções criativas.
 6. **Avaliação e priorização**: após a sessão de *brainstorming*, avalie e priorize as ideias geradas para determinar quais são mais relevantes para os requisitos do projeto.





- **IDEF (*Integrated Definition Methods*)**
 - A **técnica IDEF** (*Integrated Definition Methods*) representa um conjunto de métodos de modelagem gráfica, utilizados para descrever todo o ciclo de vida do desenvolvimento de um sistema.
 - Composto por 16 métodos, essa técnica foi elaborada durante os anos 70, por meio do *Program for Integrated Computer Aided Manufacturing* (ICAM), da Força Aérea Norte-americana
 - Seu objetivo é exibir o fluxo de informações, auxiliando no processo de modelagem e desenvolvimento de sistemas.





■ IDEF (*Integrated Definition Methods*)

Método IDEF	Nome	Descrição
IDEF0	<i>Function modeling</i>	Método de modelagem funcional de processos que descreve as funções de um sistema.
→ IDEF1	<i>Information modeling</i>	Derivado do modelo relacional de Ted Codd, representa o método para o levantamento das informações dos objetos conceituais ou físicos existentes na empresa, para a modelagem da informação.
→ IDEF1X	<i>Data modeling</i>	Notação projetada para representar graficamente a estrutura de informação existente em um determinado negócio.
IDEF2	<i>Simulation model design</i>	Método dinâmico, que descreve o comportamento de um sistema.
IDEF3	<i>Process description capture</i>	Método para coleta e documentação dos processos atuais e futuros, expressando conhecimento sobre como um sistema, processo ou empresa funciona.





■ IDEF (*Integrated Definition Methods*)

Método IDEF	Nome	Descrição
IDEF4	<i>Object-oriented design</i>	Método que auxilia no projeto de um sistema orientado a objetos.
IDEF5	<i>Ontology description capture</i>	Método que auxilia na identificação de ontologias associadas aos processos e informações.
IDEF6	<i>Design rationale capture</i>	Método para facilitar a aquisição, representação e manipulação da lógica de <i>design</i> utilizada no desenvolvimento de sistemas corporativos.
IDEF7	<i>Information system auditing</i>	Método para auditoria de sistemas de informação.
IDEF8	<i>User interface modeling</i>	Método para produzir projetos de alta qualidade das interações que ocorrem entre os usuários e os sistemas que eles operam.
IDEF9	<i>Business constraint discovery</i>	Método projetado para auxiliar na descoberta e análise de restrições em um sistema de negócios.





■ IDEF (*Integrated Definition Methods*)

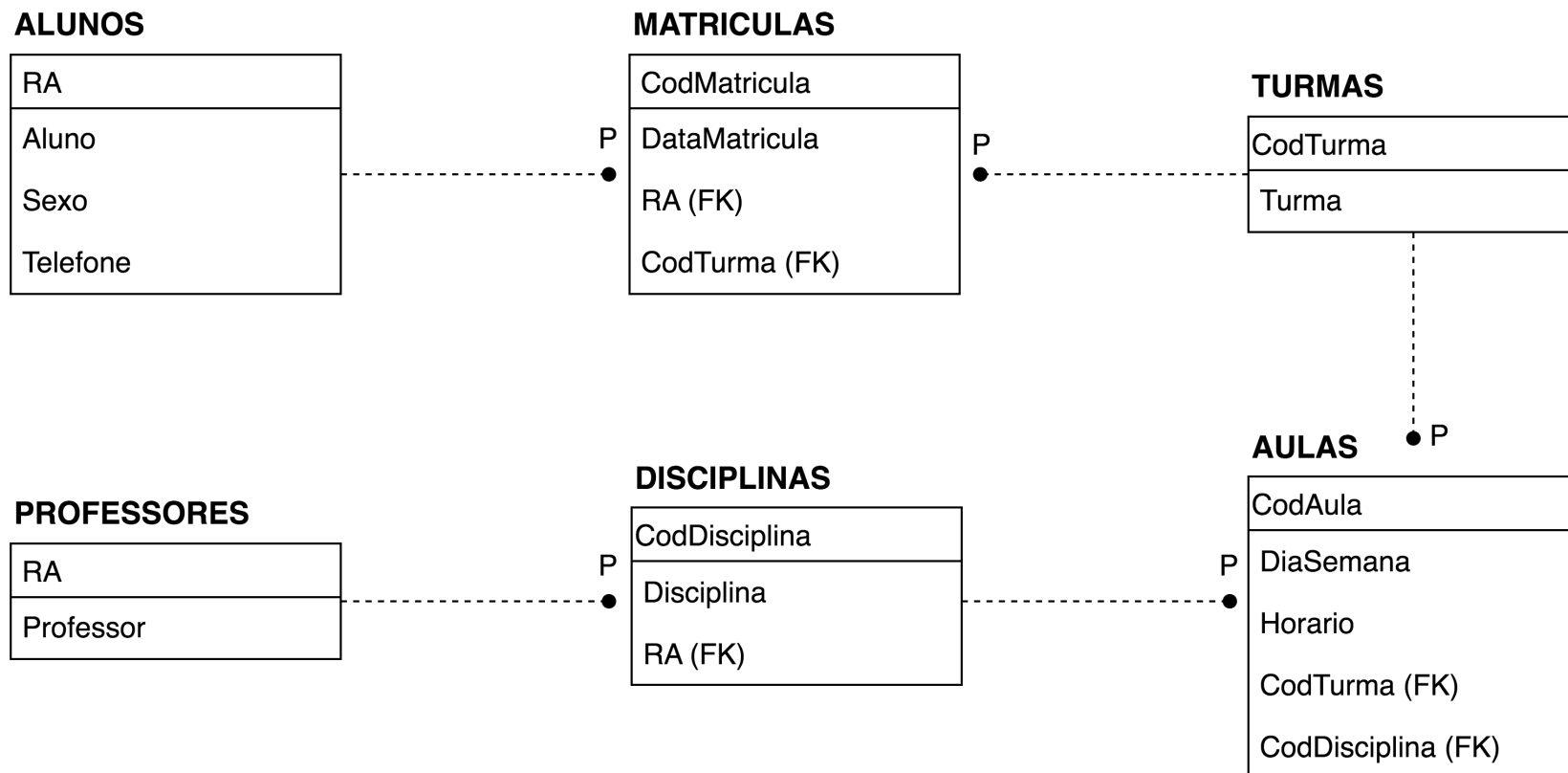
Método IDEF	Nome	Descrição
IDEF10	<i>Implementation architecture modeling</i>	Método para modelagem de arquitetura de implementação.
IDEF11	<i>Information artifact modeling</i>	Método para modelagem de artefato de informação.
IDEF12	<i>Organization modeling</i>	Método para auxiliar na modelagem organizacional.
IDEF13	<i>Three schema mapping design</i>	Método para auxiliar no projeto de mapeamento em três esquemas.
IDEF14	<i>Network design</i>	Método que visa modelar e projetar redes de computadores e comunicações.

- Considerando o projeto de banco de dados, é comum a utilização dos métodos **IDEF1** e **IDEF1X**.





■ Exemplo de notação IDEF1X





- **Cartões CRC (*Class, Responsibility, Collaboration*)**
 - A **técnica de Cartões CRC** representa uma abordagem eficaz utilizada no levantamento de requisitos, especialmente no contexto de desenvolvimento ágil de *software*.
 - Esta técnica é uma abordagem colaborativa que ajuda a definir e organizar os requisitos de um sistema de maneira interativa e centrada nos usuários.
 - O método dos Cartões CRC é particularmente valioso para equipes ágeis, pois promove a comunicação entre os membros da equipe e permite a rápida adaptação às mudanças nos requisitos.





■ Cartões CRC (*Class, Responsibility, Collaboration*)

■ Componentes de um Cartão CRC:

1. **Classe (Class)**: a classe é uma representação de um objeto ou entidade no sistema. Ela descreve um conjunto de atributos e comportamentos que são relevantes para o sistema. As classes são frequentemente nomeadas de acordo com o que representam, como **Usuário** ou **Pedido**.
2. **Responsabilidade (Responsability)**: as responsabilidades são as ações ou comportamentos associados a uma classe. Elas indicam o que a classe pode fazer ou como ela interage com outras classes ou objetos no sistema. As responsabilidades são descritas de maneira sucinta e focam nas funcionalidades essenciais.
3. **Colaboração (Collaboration)**: a colaboração descreve como as classes interagem umas com as outras para cumprir suas responsabilidades. Isso inclui a identificação das dependências entre as classes, os métodos de comunicação e como as classes contribuem para alcançar os objetivos do sistema.





■ Cartões CRC (*Class, Responsibility, Collaboration*)

- Processos para a utilização de um Cartão CRC:
 1. **Identificação das classes:** a equipe começa identificando as classes ou objetos relevantes para o sistema em discussão. Isso geralmente envolve a revisão de documentos de requisitos, discussões com as partes interessadas e análise do domínio do problema.
 2. **Atribuição de responsabilidades:** para cada classe identificada, a equipe atribui responsabilidades, ou seja, quais ações ou comportamentos essa classe deve executar. Isso ajuda a definir o comportamento esperado do sistema.
 3. **Mapeamento de colaborações:** em seguida, a equipe mapeia as colaborações entre as classes, determinando como elas interagem para cumprir suas responsabilidades. Isso ajuda a entender como as partes do sistema se encaixam.
 4. **Revisão e refinamento:** o processo de criação de cartões CRC é iterativo. A equipe revisa e refina continuamente os cartões à medida que ganha mais compreensão sobre o sistema ou à medida que os requisitos evoluem.



■ Exemplo de Cartão CRC

Nome da classe	
Responsabilidades	Colaboração

Modelo de
Cartão CRC

Nome da classe: Conta Corrente	
Responsabilidades	Colaboração
Saber seu saldo Saber seu cliente Manter histórico de transações Realizar saques e depósitos	Cliente Histórico de Transações

Exemplo
preenchido





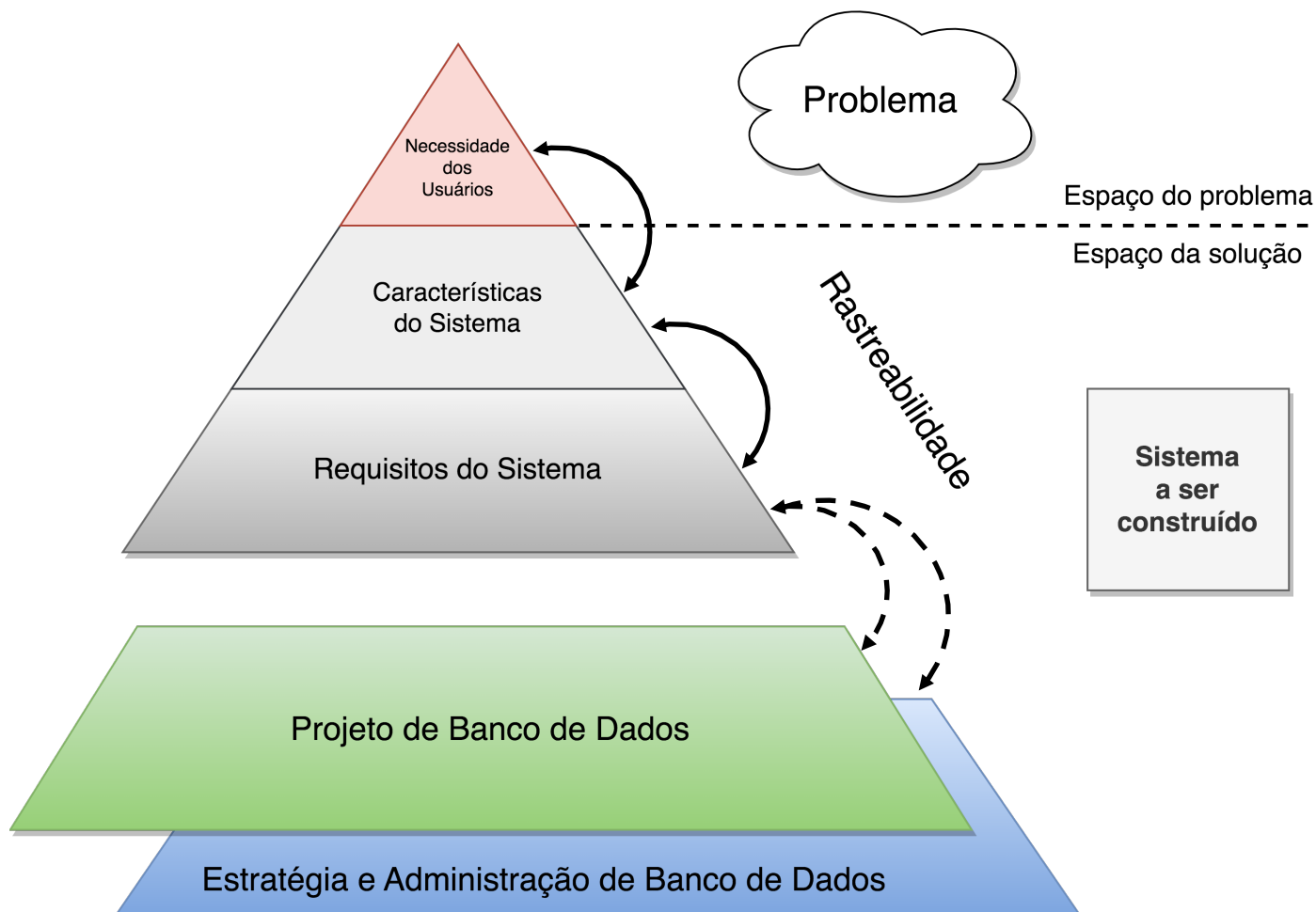
■ Gerenciamento de requisitos

- O levantamento de informações e a identificação de requisitos podem parecer duas atividades bem simples. Porém, na prática, essas atividades costumam apresentar várias dificuldades.
- O **gerenciamento de requisitos** é uma maneira sistemática de descobrir, documentar, organizar e rastrear os requisitos de um sistema.
- A **rastreabilidade** é a capacidade de relacionar os elementos existentes dentro de um projeto de sistema, de forma que haja sentido na relação entre eles.





■ Gerenciamento de requisitos





■ Gerenciamento de requisitos

- O estabelecimento da rastreabilidade entre os elementos do projeto pode auxiliar nos seguintes aspectos:
 - Entender a origem dos requisitos.
 - Gerenciar o escopo do projeto, por meio do controle dos requisitos que foram coletados.
 - Gerenciar mudanças nos requisitos.
 - Avaliar o impacto da mudança de um requisito dentro do projeto.
 - Avaliar o impacto da falha de um teste para um requisito.
 - Verificar se todos os requisitos do sistema foram implementados.





- **Na próxima aula veremos...**
 - Modelagem Entidade-Relacionamento.

