20 - 05 - 2024

Back- End+

Modelagem de Sistemas

- Elaborar documento de requisitos do projeto;
- Elaborar diagramas UML do projeto;
- Utilizar técnicas e ferramentas para registrar informações relevantes do projeto.

Requisitos não funcionais

Os requisitos não funcionais são aspectos do sistema que não estão relacionados diretamente com funcionalidades específicas, mas que são cruciais para garantir a qualidade e o desempenho do sistema como um todo.

Principais Categorias de Requisitos Não Funcionais:

Desempenho:

- **Tempo de resposta:** Define o tempo máximo permitido para que o sistema responda a uma solicitação.
- **Capacidade:** Relacionado à capacidade máxima de usuários, transações, etc., que o sistema pode suportar simultaneamente.
- **Escalabilidade:** Avalia a capacidade do sistema de se expandir para lidar com um aumento de carga.

Confiabilidade:

- **Disponibilidade:** Define o tempo durante o qual o sistema deve estar disponível para os usuários.
- **Confiança:** Indica a probabilidade de o sistema executar corretamente suas funções sem falhas.

Segurança:

- Autenticação e autorização: Métodos para garantir que apenas usuários autorizados tenham acesso ao sistema.
- Integridade: Protege contra alterações não autorizadas de dados.
- Privacidade: Garante que informações confidenciais sejam protegidas.

× × × ×

Usabilidade:

- **Interface do usuário:** Avalia a facilidade de uso e a satisfação do usuário.
- **Acessibilidade:** Garante que o sistema seja utilizável por pessoas com diferentes habilidades e necessidades.

Segurança:

- **Autenticação e autorização:** Métodos para garantir que apenas usuários autorizados tenham acesso ao sistema.
- Integridade: Protege contra alterações não autorizadas de dados.
- **Privacidade:** Garante que informações confidenciais sejam protegidas.

Manutenibilidade:

- **Facilidade de manutenção:** Refere-se à simplicidade com que o sistema pode ser mantido ou modificado.
- **Testabilidade:** Indica a facilidade com que o sistema pode ser testado para garantir sua integridade após alterações.

Portabilidade:

- **Adaptabilidade:** A capacidade do sistema de funcionar em diferentes ambientes.
- **Interoperabilidade:** Garante que o sistema pode interagir com outros sistemas.

Importância dos Requisitos Não Funcionais:

- **Garantia de Qualidade:** Os requisitos não funcionais ajudam a garantir que o sistema atenda aos padrões de qualidade desejados.
- **Experiência do Usuário:** Aspectos como usabilidade e desempenho afetam diretamente a satisfação do usuário.
- **Planejamento e Estimativa:** São cruciais para estimativas precisas de tempo e recursos durante o ciclo de vida do desenvolvimento do sistema.
- **Tomada de Decisões Arquiteturais:** Podem influenciar decisões relacionadas à arquitetura do sistema.

Conclusão:

Ao modelar sistemas, a consideração cuidadosa dos requisitos não funcionais é essencial para garantir que o sistema seja não apenas funcional, mas também eficiente, seguro, usável e fácil de manter. A identificação precoce desses requisitos contribui significativamente para o sucesso do projeto.

Exemplos:

Desempenho:

- Tempo de Resposta: O sistema deve responder a uma solicitação do usuário em menos de 2 segundos.
- Capacidade: O sistema deve suportar até 1000 usuários simultâneos.
- Escalabilidade: O sistema deve ser capaz de escalar horizontalmente para lidar com um aumento de 20% na carga de trabalho.

Exemplos:

Confiabilidade:

- Disponibilidade: O sistema deve estar disponível 99,9% do tempo.
- Confiança: A taxa de falhas críticas não deve exceder 0,1% das transações.

Exemplos:

Segurança:

- Autenticação e Autorização: A autenticação de dois fatores será implementada para todos os usuários administrativos.
- Integridade: Os dados críticos do sistema devem ser protegidos contra alterações não autorizadas.
- Privacidade: Informações pessoais dos usuários devem ser armazenadas de acordo com as regulamentações de privacidade.

Exemplos:

Usabilidade:

- Interface do Usuário: A interface do usuário deve ser intuitiva, seguindo as diretrizes de design de experiência do usuário (UX).
- Acessibilidade: O sistema deve ser acessível para usuários com deficiências visuais, seguindo as normas de acessibilidade WCAG.

Exemplos:

Manutenibilidade:

- Facilidade de Manutenção: O código-fonte do sistema deve ser modular, permitindo fácil manutenção e extensibilidade.
- Testabilidade: Deve ser possível realizar testes automatizados em pelo menos 80% do código do sistema.

Exemplos:

Portabilidade:

- Adaptabilidade: O sistema deve ser capaz de se adaptar a diferentes sistemas operacionais, incluindo Windows, Linux e macOS.
- Interoperabilidade: O sistema deve ser capaz de trocar dados de forma eficiente com outros sistemas através de APIs padrão.

Esses exemplos ilustram a diversidade de requisitos não funcionais e como eles podem variar dependendo das necessidades específicas do sistema em questão. Ao elaborar requisitos não funcionais, é crucial que eles sejam mensuráveis e testáveis para garantir que possam ser verificados durante o desenvolvimento e a implementação do sistema.

Requisitos Funcionais

Definição de Requisitos Funcionais:

- São as funcionalidades que o sistema deve fornecer para atender às necessidades do usuário.
- Descrevem as ações específicas que o sistema deve executar.

Requisitos Funcionais

Importância dos Requisitos Funcionais:

- Formam a base para o design do sistema.
- Orientam o desenvolvimento e testes.
- Servem como critério de aceitação para os usuários.

Identificação de Requisitos Funcionais:

Elicitação de Requisitos:

- Entrevistas com stakeholders.
- Observação do ambiente de trabalho.
- Questionários e pesquisas.
- Análise de documentos existentes.

Identificação de Requisitos Funcionais:

Tipos de Requisitos Funcionais:

- Requisitos de Sistema: relacionados ao comportamento do sistema como um todo.
- Requisitos de Usuário: relacionados às ações específicas dos usuários.
- Requisitos de Produto: relacionados às características específicas do produto.

Identificação de Requisitos Funcionais:

Documentação de Requisitos Funcionais:

- Estrutura de Documentação:
  - Identificação e descrição do requisito.
  - Prioridade do requisito.
  - Critérios de aceitação.
- Modelagem de Requisitos:
  - Use casos.
  - Diagramas de sequência.
  - Fluxogramas.

Identificação de Requisitos Funcionais:

Exemplos de Requisitos Funcionais

RF001: Autenticação de Usuário

- Descrição: O sistema deve permitir que os usuários se autentiquem usando nome de usuário e senha.
- Prioridade: Alta.
- Critérios de Aceitação: O usuário deve ser redirecionado para a página principal após autenticação bem-sucedida.

Identificação de Requisitos Funcionais:

RF002: Cadastro de Cliente

- Descrição: O sistema deve possibilitar o cadastro de novos clientes, incluindo informações como nome, endereço e número de telefone.
- Prioridade: Média.
- Critérios de Aceitação: Deve ser possível visualizar e editar os dados do cliente após o cadastro.

Identificação de Requisitos Funcionais:

Conclusão:

- Os requisitos funcionais são fundamentais na construção de sistemas que atendam às necessidades dos usuários.
- A elicitação cuidadosa e a documentação clara são essenciais para o sucesso do projeto.
- A modelagem de requisitos ajuda a visualizar e comunicar as funcionalidades do sistema.

Documento de Requisitos

O Documento de Requisitos é um texto detalhado utilizado para guiar o desenvolvimento de um sistema. Ele contém uma descrição completa de como o sistema deve funcionar, incluindo os requisitos funcionais e não funcionais, restrições, casos de uso e cenários.

Importância

Serve como uma ponte entre os stakeholders (clientes, usuários, desenvolvedores, etc.) e o time de desenvolvimento.

Reduz ambiguidades, garantindo que todos tenham a mesma compreensão sobre o que será desenvolvido.

Ajuda na estimativa de recursos, tempo e custo do projeto.

Como Criar um Documento de Requisitos?

Identificação dos Stakeholders: Entender quem são os usuários, clientes e outras partes interessadas.

Coleta de Requisitos: Utilizar entrevistas, questionários, observações e outras técnicas para coletar requisitos.
Classificação dos Requisitos: Separar em requisitos funcionais (ações do sistema) e não funcionais (desempenho, segurança, etc.).

Como Criar um Documento de Requisitos?

Validação: Revisar os requisitos com os stakeholders para garantir precisão e completude.

Documentação: Escrever os requisitos de forma clara e detalhada, incluindo casos de uso e cenários.

Regras de Negócios:

Regras de negócio são diretrizes, condições e lógicas específicas que orientam o comportamento, operações, decisões e transações de um sistema empresarial. Elas refletem as políticas, procedimentos e critérios operacionais da organização.

Importância

- Definem como o negócio opera e quais são seus critérios de decisão.
- Garantem que o sistema esteja alinhado com os objetivos e estratégias da empresa.
- Facilitam a adaptação do sistema a mudanças nas estratégias de negócio.

Identificação e Documentação de Regras de Negócio

- Identificação: Envolve o entendimento profundo dos processos de negócio através de entrevistas com especialistas de domínio, análise de documentos etc.

- Documentação: As regras devem ser documentadas de forma clara e acessível, preferencialmente em uma linguagem próxima da natural, para fácil entendimento por todos os stakeholders.

Modelagem de Regras de Negócio

Modelagem de regras de negócio é o processo de representar as regras de negócio de forma estruturada, usando notações ou ferramentas específicas, para facilitar a implementação, gestão e alteração dessas regras no sistema.

Técnicas de Modelagem

Diagramas de Atividade UML: Representam os processos de negócio e onde as regras se aplicam.

Tabelas de Decisão: Tabelas que mapeiam condições a ações, facilitando a visualização de regras complexas.

Business Rule Management Systems (BRMS): Ferramentas especializadas que permitem a definição, armazenamento, gestão e execução de regras de negócio.

Implementação

Deve-se escolher a técnica de modelagem mais adequada ao contexto do projeto e às capacidades da equipe.

A implementação das regras de negócio deve ser feita de forma que permita fácil manutenção e atualização.

Conclusão

Entender e documentar corretamente os requisitos e as regras de negócio é fundamental para o sucesso de um projeto de desenvolvimento de software. A modelagem de regras de negócio, por sua vez, facilita a implementação, gestão e adaptação das regras ao longo do tempo, garantindo que o sistema permaneça alinhado com os objetivos do negócio.

Escopo do Projeto de Software

Hoje, vamos explorar um aspecto crucial no desenvolvimento de software: o escopo do projeto. Compreender o escopo é vital para o sucesso de qualquer projeto de software, pois define os limites e os objetivos do que será entregue. Vamos detalhar os componentes chave do escopo: premissas, restrições, riscos, papéis e a utilização de Diagramas UML.

Premissas

Premissas são suposições aceitas como verdadeiras para o planejamento do projeto. Elas formam a base sobre a qual o projeto é construído e influenciam diretamente seu planejamento e execução.

Importância

Estabelecem um entendimento comum entre todos os envolvidos no projeto.

Ajudam na definição do escopo, identificando o que está incluído ou excluído do projeto.

Restrições

Restrições são limitações que o projeto deve respeitar. Podem incluir limites de tempo, orçamento, recursos disponíveis, tecnologias específicas, etc.

Importância

- Identificar as restrições é crucial para o planejamento realista do projeto.
- Auxiliam na gestão de expectativas dos stakeholders.

Riscos

Riscos são eventos ou condições incertas que, se ocorrerem, terão um efeito positivo ou negativo nos objetivos do projeto.

Gestão de Riscos

- Identificação: Reconhecer os riscos potenciais.
- Análise: Avaliar a probabilidade e o impacto dos riscos.
- Planejamento de Resposta: Desenvolver ações para mitigar ou aproveitar os riscos.
- Monitoramento: Acompanhar os riscos identificados e ajustar planos conforme necessário.

Papéis

A definição clara de papéis e responsabilidades é fundamental para o sucesso do projeto. Isso inclui papéis como gestor de projeto, analista de negócios, desenvolvedor, testador, entre outros.

Gestor de Projeto: Responsável pela coordenação do projeto.
Analista de Negócios: Define requisitos e necessidades de negócios.
Desenvolvedor: Constrói a solução de software.
Testador: Verifica a qualidade e a conformidade do software.

Diagramas UML

O que é UML?

A Linguagem de Modelagem Unificada (UML) é uma linguagem padrão para especificar, visualizar, construir e documentar os artefatos de um sistema de software.

Diagramas UML no Escopo do Projeto

Os Diagramas UML ajudam a visualizar o escopo do projeto, oferecendo uma representação gráfica dos requisitos, da arquitetura e do design do sistema.

Diagramas UML

Diagrama de Casos de Uso: Mostra as funcionalidades do sistema e suas interações com os usuários.

Diagrama de Classes: Representa a estrutura e o relacionamento entre as classes no sistema.

Diagrama de Sequência: Ilustra como os objetos interagem em termos de uma sequência de mensagens.

Caso de Uso

É uma representação de uma interação possível entre um sistema e seus atores (usuários ou outros sistemas). Ele descreve as funcionalidades do sistema do ponto de vista do usuário.

Elementos-Chave em um Diagrama de Caso de Uso:

- Ator Principal: Identificação do ator principal, representando o usuário primário ou sistema externo.
- Caso de Uso Principal: Destaque para o caso de uso principal, muitas vezes relacionado à funcionalidade principal do sistema.
- Ativadores Secundários: Identificação de outros atores que interagem com o sistema, mas não são os principais.

Relações em Diagramas de Caso de Uso:

- Inclusão: Representa a inclusão de um caso de uso em outro. Útil para evitar duplicação de ações comuns.
- Extensão: Indica que um caso de uso pode ser estendido por outro em certas condições. Útil para representar comportamentos opcionais ou alternativos.

Elaboração de Descrições de Caso de Uso:

- Pré-condições e Pós-condições: Detalhes sobre o estado inicial necessário e o estado final esperado após a execução do caso de uso.
- Fluxo Principal: Descrição passo a passo do fluxo normal de execução do caso de uso.
- Fluxos Alternativos e Exceções: Destaque para cenários alternativos e exceções que podem ocorrer durante a execução do caso de uso.

Boas Práticas na Modelagem de Casos de Uso:

- Nomes Significativos: Utilizar nomes claros e significativos para atores e casos de uso.
- Granularidade Adequada: Evitar casos de uso muito grandes ou muito pequenos. Buscar uma granularidade que seja compreensível e gerenciável.
- Manter Simplicidade: Evitar detalhes desnecessários. Casos de uso devem ser compreensíveis para diferentes partes interessadas.

Ferramentas e Notações Específicas:

- Uso de Estereótipos: Explorar estereótipos UML específicos para casos de uso, como <<extend>> e <<include>>.
- Identificação Visual: Utilizar cores, notas e ícones para destacar informações importantes ou fornecer detalhes adicionais.

Atividade Prática:

Os alunos podem criar um diagrama de caso de uso para um sistema real ou fictício, seguindo as boas práticas aprendidas na aula. Podem também elaborar descrições detalhadas para alguns casos de uso específicos.

Conclusão:

A modelagem de casos de uso é uma parte crucial do desenvolvimento de software, proporcionando uma compreensão clara das interações entre usuários e o sistema. Dominar os detalhes, relações e práticas recomendadas na elaboração de diagramas de caso de uso contribui significativamente para o sucesso na comunicação e no desenvolvimento de sistemas robustos e orientados ao usuário.

Classe:

Classes são a base da orientação a objetos, representando entidades no sistema e suas propriedades e comportamentos. Aqui estão alguns pontos importantes:

Classe:

- **Atributos:** São as propriedades da classe, representando características das entidades que a classe modela.

- **Métodos:** Representam os comportamentos ou ações que a classe pode executar. Métodos geralmente manipulam os atributos da classe.

Classe:

Relações entre Classes: Relações como associação, composição e herança descrevem como as classes se relacionam entre si. Associação indica que as classes estão conectadas de alguma forma, composição indica uma relação mais forte, e herança representa uma relação de especialização/generalização.

Sequência:

Diagramas de Sequência ajudam a visualizar a interação entre objetos ao longo do tempo. Eles são especialmente úteis para entender como objetos colaboram em um cenário específico. Alguns pontos importantes:

Sequência:

- **Objeto:** Uma instância de uma classe que participa da interação. Pode ser um ator, uma classe, ou mesmo um sistema externo.
- **Mensagens:** Representam as interações entre objetos. Existem mensagens síncronas (o remetente aguarda a resposta) e mensagens assíncronas (o remetente continua sem esperar pela resposta).
- **Ativação e Desativação:** Indicam quando um objeto está ativo e envolvido na interação e quando ele se desativa.