Análise de Sistemas e Padrões de Projeto

Prof: Wagner

BeTech. Integrado

Processo de Desenvolvimento de Software

Um projeto de desenvolvimento de software deve passar por alguns processos até a entrega do produto de software:

- Levantamento de Requisitos
- Análise e Especificação de Requisitos
- Especificação de Sistema
- Implementação / Testes
- Implantação
- Manutenção

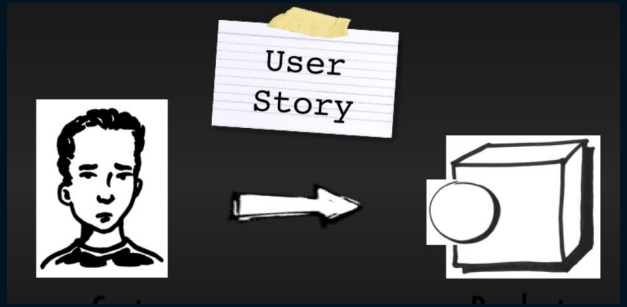
Processo de Desenvolvimento de Software

Um projeto de desenvolvimento de software deve passar por alguns processos até a entrega do produto de software:

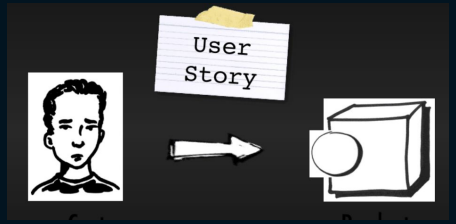
- Levantamento de Requisitos
- Análise e Especificação de Requisitos
- Especificação de Sistema
- Implementação / Testes
- Implantação
- Manutenção

O que são Requisitos (do Sistema)?

Requisitos em um projeto de software são descrições detalhadas das funcionalidades, características e restrições que o sistema deve atender para satisfazer as necessidades dos usuários e partes interessadas.



Representação dos Requisitos



Eu, enquanto QUEM, necessito O QUE, porque EXPLICAÇÃO.

Eu, enquanto caixa de banco, necessito realizar depósito para uma conta bancária porque faz parte das minhas atribuições.

Importância dos Requisitos

São a base sobre a qual todo o processo de desenvolvimento é construído, orientando as atividades desde o design até a implementação e testes.

- Estabelecem as bases para o desenvolvimento.
- Reduzem a ambiguidade e o risco de interpretações equivocadas.
- Orientam o processo de teste e validação.
- Contribuem para a satisfação do cliente ao entregar um produto alinhado com suas expectativas.

Tipos de Requisitos

Requisitos de Negócios

Requisitos de Usuário

Requisitos Funcionais

Requisitos Não Funcionais

Estudo de Caso

SISTEMA DE RESERVAS PARA COMPANHIA AÉREA

Uma companhia aérea está enfrentando desafios na eficiência do processo de reservas, resultando em atrasos, erros e insatisfação dos clientes.

A empresa decidiu desenvolver um novo sistema de reservas para melhorar a experiência do cliente e otimizar suas operações internas.

Requisitos de Negócios

Refletem as necessidades e objetivos do negócio que o software pretende atender.

Para o estudo de caso, os requisitos de negócio são:

- **Eficiência operacional**: O sistema deve otimizar o processo de reservas para reduzir o tempo necessário para completar uma transação.
- **Integração com outros sistemas**: Deve ser capaz de integrar-se de forma eficiente com sistemas de controle de tráfego aéreo, sistemas financeiros e de gerenciamento de passageiros.
- **Satisfação do cliente**: o novo sistema deve garantir que o processo de reserva seja fácil e intuitivo para os clientes, aumentando a satisfação geral.

Requisitos de Usuário

Representam as expectativas e necessidades dos usuários finais do sistema.

Para o estudo de caso, os requisitos de usuário são:

- **Interface amigável**: Os usuários (agentes de reservas, clientes online) devem ter acesso a uma interface amigável e fácil de usar para efetuar reservas.
- **Suporte móvel**: O sistema deve ser acessível a partir de dispositivos móveis para permitir que os clientes realizem reservas enquanto estão em movimento.
- **Comunicação em tempo real**: Oferecer atualizações em tempo real aos clientes sobre mudanças de horário, cancelamentos de voos ou atualizações relevantes.

Requisitos Funcionais

Descrevem as funções específicas que o sistema deve realizar.

Para o estudo de caso, os requisitos funcionais são:

- **Reservas Online:** Os usuários devem ser capazes de efetuar reservas de passagens online, selecionando voos, assentos e adicionando serviços adicionais.
- **Gestão de Assentos:** O sistema deve permitir a seleção e gestão eficaz de assentos, considerando preferências dos passageiros e restrições operacionais.
- **Atualizações de Voo:** Fornecer informações em tempo real sobre o status dos voos, incluindo atrasos, cancelamentos e mudanças de gate.

Requisitos Não Funcionais

Estabelecem as características do sistema que não estão diretamente relacionadas às funcionalidades, mas impactam a qualidade e o desempenho.

Para o estudo de caso, os requisitos não funcionais são:

- **Desempenho rápido**: Garantir tempos de resposta rápidos durante picos de atividade, como períodos de alta demanda de reservas.
- **Segurança de dados**: Implementar protocolos robustos de segurança para proteger informações sensíveis dos clientes, como dados pessoais e detalhes de pagamento.
- **Disponibilidade contínua**: Assegurar que o sistema esteja disponível 24/7, minimizando tempo de inatividade e interrupções no processo de reserva.

Conhecimento adquirido ...

Até agora você compreendeu o significado, a importância e os tipos de requisito.

Você viu também quais podem ser os requisitos de um estudo de caso

Análise de Sistemas e Padrões de Projeto

Prof: Wagner

Aula: 02

BeTech. Integrado

Relembrando ...

Um projeto de desenvolvimento de software deve passar por alguns processos até a entrega do produto de software:

- Levantamento de Requisitos
- Análise e Especificação de Requisitos
- Especificação de Sistema
- Implementação / Testes
- Implantação
- Manutenção

Levantamento, Análise e Especificação de Requisitos

O sucesso de um projeto de desenvolvimento de software está fortemente ligado à qualidade tanto do processo de Levantamento de requisitos quanto dos processos de Análise e especificação de requisitos.

Essas etapas desempenham um papel crucial na criação de soluções eficazes que atendam às necessidades dos usuários e das partes interessadas.

Levantamento de Requisitos

É o ponto de partida para compreender as expectativas e requisitos dos stakeholders. É nesta fase que as bases do projeto são estabelecidas, tornando essencial que seja uma abordagem cuidadosa e abrangente.

Técnicas para Levantamento de Requisitos

Entrevistas com os stakeholders para entender suas necessidades e expectativas em relação ao novo sistema.

Workshops colaborativos com membros da equipe de desenvolvimento e stakeholders para discutir e elaborar os requisitos do projeto.

Análise de documentos, como especificações anteriores, relatórios de usuários e manuais de procedimentos, para extrair requisitos relevantes para o novo sistema.

Protótipos interativos do sistema para permitir que os stakeholders visualizem e validem os requisitos antes da implementação completa do software.

Técnicas de modelagem, como diagramas de classes e casos de uso, para representar os requisitos funcionais do sistema.

Levantamento de requisitos - Etapas

Identificação das partes interessadas: Começa-se identificando todas as partes interessadas envolvidas no projeto, como clientes, usuários finais, gerentes, etc pois cada um têm diferente perspectiva e necessidades que devem ser consideradas.

Coleta de informações: Realiza-se entrevistas, questionários, workshops e observações para coletar informações sobre as necessidades, expectativas e restrições do sistema. Deve-se entender o como o sistema deve funcionar.

Análise do ambiente: Compreende-se o contexto ao qual o sistema será utilizado, incluindo aspectos organizacionais, tecnológicos, legais e de mercado. Permite identificar influências externas que possam afetar os requisitos do sistema.

Documentação preliminar: Deve-se documentar as informações coletadas em uma lista de requisitos ou esboço de especificação dos requisitos. Servirá como base para as etapas seguintes de análise e especificação.

Análise de requisitos

A análise de requisitos procura compreender os requisitos identificados, destacando as interações, conflitos e prioridades destes.

Análise de requisitos - Etapas

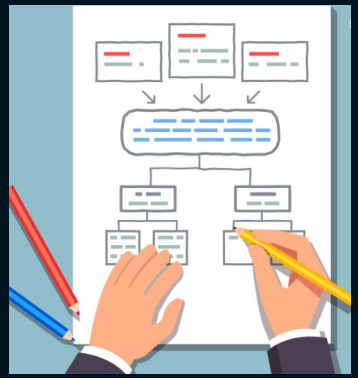
Classificação e Priorização: Os requisitos coletados são classificados e priorizados, com base em sua importância para o sucesso do projeto e sua viabilidade técnica.

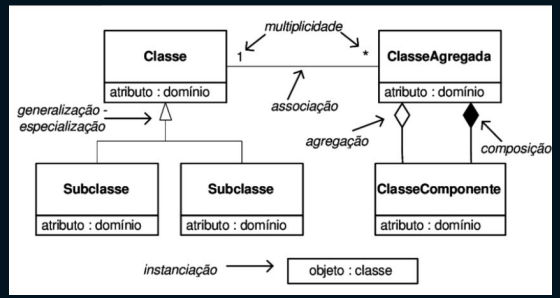
Validação de requisitos: Os requisitos são revisados e validados para garantir que sejam claros, completos, consistentes e viáveis. Qualquer ambiguidade ou inconsistência é identificada e resolvida nessa fase.

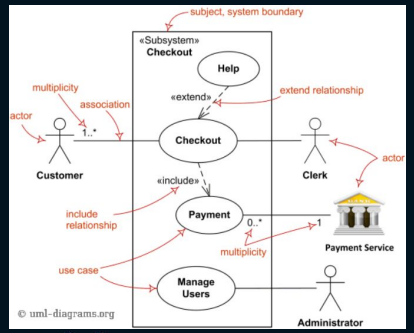
Modelagem de requisitos: Define-se fluxogramas e/ou diagramas UML (Caso de uso e de Classes) para representar os requisitos e os relacionamentos entre eles. Isso ajuda que a equipe de desenvolvimento e as partes interessadas melhorarem a compreensão dos requisitos.

Elicitação adicional: Se necessário, mais sessões de levantamento de requisitos são realizadas para esclarecer dúvidas, resolver conflitos ou capturar novos requisitos (que possam ter sido esquecidos/omitidos no levantamento inicial).

Análise de requisitos - Diagramas





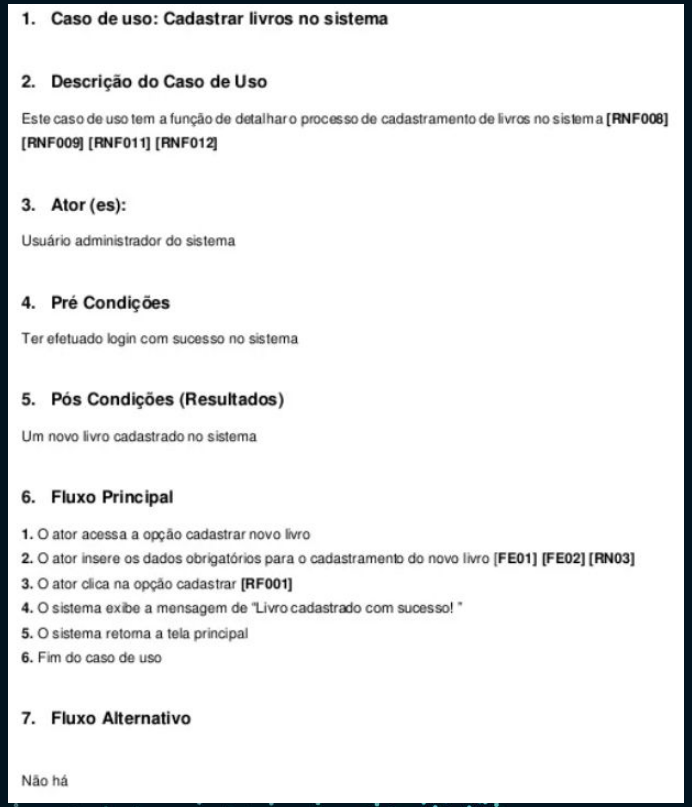


Especificação de requisitos

Visa transformar a informação coletada/analisada de forma que possa ser compreendida pela equipe.

A precisão e clareza nesta fase garantem que os envolvidos compartilham uma visão comum do sistema a ser desenvolvido.

Evita ambiguidades que podem levar a interpretações equivocadas.



1. Caso de uso: Cadastrar livros no sistema

2. Descrição do Caso de Uso

Este caso de uso tem a função de detalhar o processo de cadastramento de livros no sistema **[RNF008] [RNF009] [RNF011] [RNF012]**

3. Ator (es):

Usuário administrador do sistema

4. Pré Condições

Ter efetuado login com sucesso no sistema

5. Pós Condições (Resultados)

Um novo livro cadastrado no sistema

6. Fluxo Principal

1. O ator acessa a opção cadastrar novo livro
2. O ator insere os dados obrigatórios para o cadastramento do novo livro **[FE01] [FE02] [RN03]**
3. O ator clica na opção cadastrar **[RF001]**
4. O sistema exibe a mensagem de "Livro cadastrado com sucesso! "
5. O sistema retoma a tela principal
6. Fim do caso de uso

7. Fluxo Alternativo

Não há

Levantamento, Análise e Especificação de requisitos

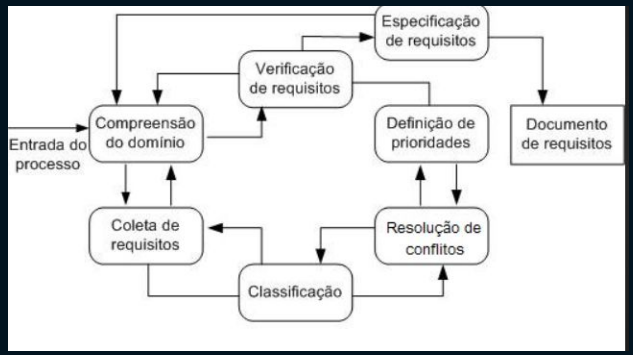
O processo de levantamento, análise e especificação de requisitos tem um impacto significativo nas etapas subsequentes do desenvolvimento de software!

😢Requisitos mal definidos podem resultar em retrabalho, atrasos e custos adicionais.

😇Uma abordagem sólida pode economizar tempo e recursos, aumentando a eficiência do projeto.

Conhecimento adquirido ...

Nesta aula você compreendeu a base dos processos de Levantamento, Análise e Especificação de requisitos.



Logo mais vamos aprofundar esses processos!

Análise de Sistemas e Padrões de Projeto

BeTech. Integrado

Prof: Wagner

Aula: 03

Tamanho, duração e custo de um projeto

O processo de gerenciamento de projetos de software é muito importante, para garantir qualidade na prestação de serviço.

Essas informações são essenciais para:

- planejar adequadamente os recursos necessários;
- definir metas realistas; e
- corresponder às expectativas dos *stakeholders*.

Estimativa de Tamanho

Visa a quantificação do escopo do projeto através de técnicas de estimativa de tamanho:

- Pontos de Função - mede o tamanho funcional de um sistema com base nas funções que oferece aos usuários.
- Casos de Uso - avalia o número e a complexidade dos casos de uso do sistema.
- Story Points - técnica ágil para estimar o tamanho das histórias de usuário.
- Linhas de Código - medida direta do tamanho do código fonte do software.

Estimativa de duração

Envolve a previsão do tempo necessário para concluir as atividades do projeto. Deve considerar complexidade das tarefas, disponibilidade de recursos, dependências entre as atividades e riscos do projeto. Técnicas que apoiam:

- Analogia - usar estimativas de projetos semelhantes e que já ocorreram.
- Decomposição - dividir o projeto em tarefas menores e estimar a duração de cada uma.
- Opinião especializada - especialistas no assunto indicam *insights* sobre a duração das atividades.
- Método PERT (*Program Evaluation and Review Technique*) - abordagem estatística que leva em consideração estimativas pessimistas, otimistas e mais prováveis.

Estimativa de Custo

Envolve a previsão dos recursos financeiros necessários para concluir o projeto. Isso inclui custos de mão de obra, custos de hardware e software, custos indiretos, entre outros. Abordagens que apoiam esta estimativa:

- Estimativa Análoga - usa estimativas de custo em projetos anteriores semelhantes.
- Estimativa Bottom-Up - estimar o custo de cada componente do projeto e somá-los para obter o custo total.
- Estimativa de Três Pontos - leva em consideração estimativas pessimistas, otimistas e mais prováveis para calcular um custo esperado.
- Análise de Custo-Benefício - compara os custos esperados do projeto com os benefícios esperados para determinar se o investimento é justificado.

Cuidados ...

É importante reconhecer que são apenas previsões, e podem variar ao longo do ciclo de vida do projeto.

Deve-se revisar e atualizar regularmente as estimativas na medida que mais informações se tornam disponíveis e/ou circunstâncias do projeto mudam.

Importante envolver todas as partes interessadas relevantes nos processos de estimativa para garantir que as expectativas sejam alinhadas e os compromissos sejam realistas.

Para entregar com qualidade ...

- Definir um bom escopo do projeto.
- Implantar metodologia de gestão de projetos.
- Definir cronograma realista.
- Realizar gestão de custos (dentro do orçamento) e gestão do tempo (garantir cumprimento dos prazos estimados).
- Usar ferramenta de gestão de projetos.

Conhecimento adquirido ...

Espero que você tenha compreendido como estimar tamanho, duração e custo de um projeto.

Sugiro que você continue a aprofundar seu conhecimento e conhecer as técnicas com detalhes!!

Análise de Sistemas e Padrões de Projeto

BeTech. Integrado

Prof: Prof Wagner Monteverde

O que é um Paradigma de Programação?

Paradigma é um modelo ou um padrão a ser seguido.

Um Paradigma de programação representa o tipo de estrutura que vai ser usada para definir como vai ser a construção de um projeto.

E o que significa um Paradigma de Análise e Projeto de Sistemas baseados em UML?!?

Paradigmas de Análise e Projeto de Sistemas baseados em UML

São abordagens ou metodologias que utilizam a UML como principal ferramenta para representar e modelar sistemas de software. Os **paradigmas atuais** são:

- Orientado a Objetos (*Object-Oriented Programming* - OOP)
- Baseado em Componentes
- Desenvolvimento Ágil
- Engenharia de Software dirigido por Modelos (*Model-Driven Engineering* - MDE)

Orientado a Objetos

A UML é utilizada para modelar o sistema em termos de objetos, classes, relacionamentos e seus comportamentos.

Os diagramas são empregados para representar a informação (diagrama de classes) e o comportamento (diagrama de estados e de sequência) dos sistemas orientados a objetos.

Linguagens utilizadas são Java, C++ e C#.

Baseado em Componentes

Pretende-se a modelagem de componentes reutilizáveis e suas interações. Os componentes podem ser bibliotecas de software, serviços web ou módulos de terceiros. Promove a reutilização e a modularidade do software.

A UML pode ser usada para representar componentes, interfaces, dependências e a arquitetura geral do sistema. Diagramas de componentes, diagramas de implantação e diagramas de pacotes são exemplos de diagramas UML.

Linguagens utilizadas são Java (frameworks como Spring), .NET (tecnologias como ASP.NET MVC), Ruby on Rails, Angular (framework JavaScript para interfaces de usuário).

Desenvolvimento Ágil

A UML pode apoiar desenvolvimento mais interativo e colaborativo. Diagramas utilizados são os diagramas de casos de uso, de atividades e de classes. Permitem capturar requisitos, definir funcionalidades e comunicar ideias entre os membros da equipe.

Para o desenvolvimento utiliza-se os frameworks SCRUM e Extreme Programming (xP); as linguagens de programação são Java, Python e JavaScript.

Engenharia de Software dirigido por Modelos

Concentra-se na utilização de modelos abstratos para representar sistemas de software, com a intenção de automatizar o processo de desenvolvimento.

UML é usado com ênfase na criação de modelos precisos e bem definidos que possam ser transformados em código executável.

Diagramas de classes, diagramas de sequência e diagramas de atividades são comumente utilizados na modelagem de sistemas no contexto deste paradigma.

Paradigmas de Análise e Especificação de Sistema baseado em UML

Note que a escolha do paradigma depende das necessidades do projeto, das habilidades da equipe e das preferências individuais.

Vamos focar no **Orientado a Objetos** !

Conhecimento adquirido ...

Você percebeu que existem muitos paradigmas?

Cada um trabalha com frameworks e linguagens distintas, para solucionar problemas diferenciados.

Quer compreender um pouco mais sobre os paradigma de programação? Clique aqui para acessar um Blog.

Análise de Sistemas e Padrões de Projeto

Prof: Prof Wagner Monteverde

Projeto de Sistemas baseados em UML

A Linguagem de Modelagem Unificada (*Unified Modelling Language* - **UML**) é uma linguagem padrão utilizada para modelar sistemas de software.

Fornece uma notação visual e uma metodologia para descrever e projetar sistemas de forma compreensível.

Os diagramas são divididos em:

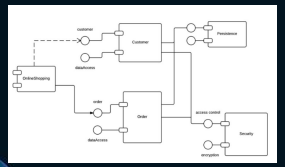
- **estruturais** visam modelar a informação que deve ser armazenada/processada;
- **comportamentais** visam o processamento das entradas/saídas e informações do sistema.

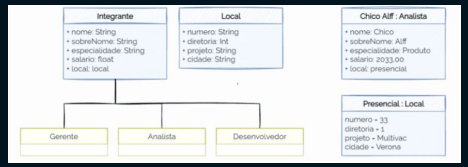
Diagramas UML Estruturais

- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Estrutura composta
- Diagrama de Implementação
- Diagrama de Pacotes

} Sempre ...

} Sistemas com alta complexidade





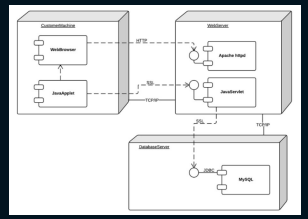
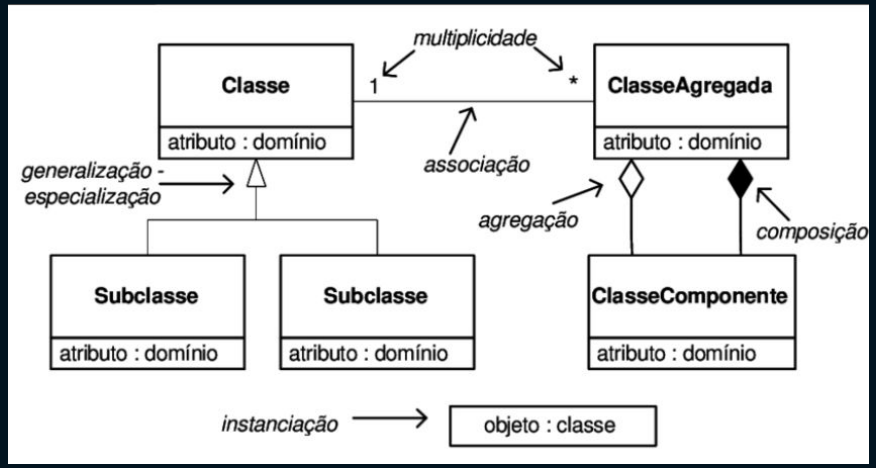


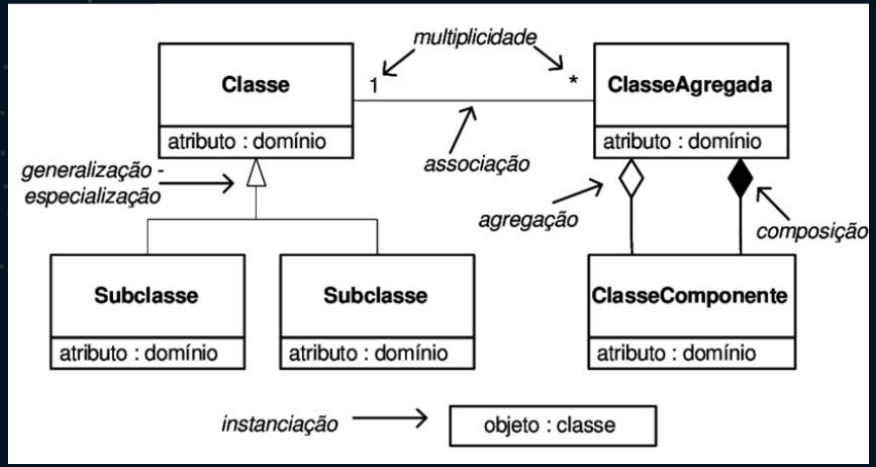
Diagrama de Classes

É o diagrama UML mais usado, a principal base de qualquer solução orientada a objetos. Classes dentro de um sistema, atributos e operações, e a relação entre cada classe. Veja detalhes aqui.



Esquema genérico

Diagrama de Classes



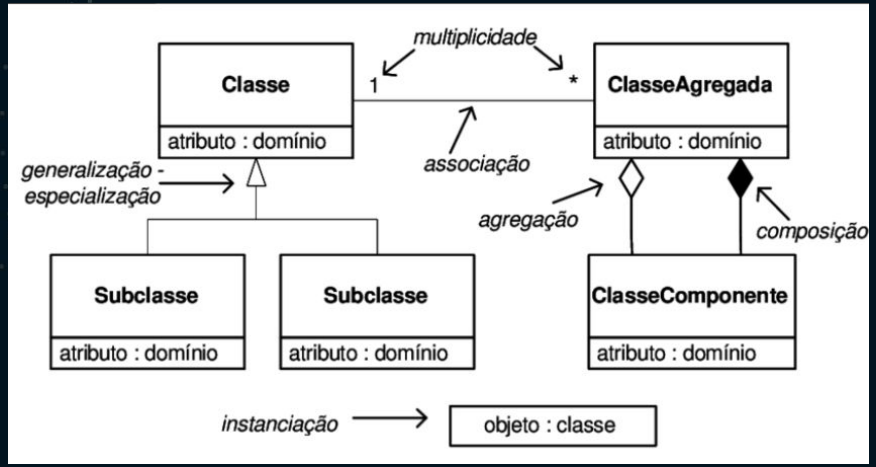
Generalização/Especialização:

A generalização/especialização é um relacionamento hierárquico entre uma classe mais genérica (superclasse) e classes mais específicas (subclasses). As subclasses herdam atributos e métodos da superclasse, podendo adicionar ou sobrescrever comportamentos específicos.

Animal é a superclasse com um atributo name e um método makeSound().

Dog e Cat são subclasses que herdam de Animal. Ambas sobrescrevem o método makeSound() para fornecer sons específicos: "Latir" para cães e "Miar" para gatos.

Diagrama de Classes



Esquema genérico

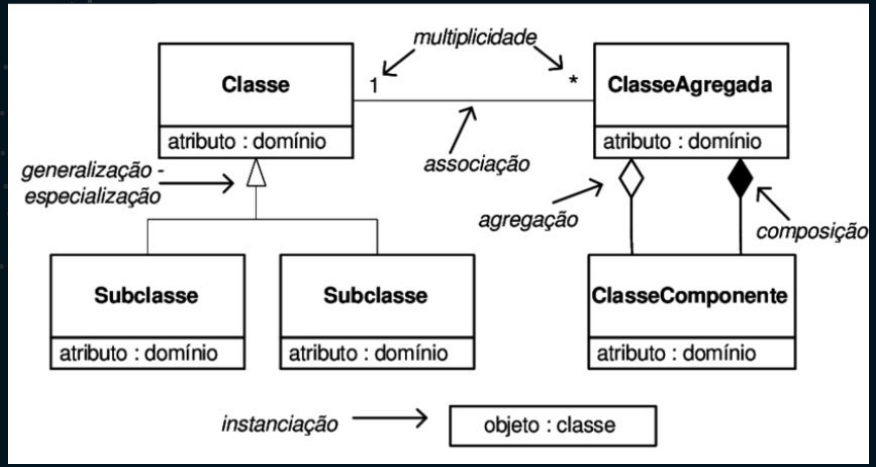
Associação:

A associação representa um relacionamento entre duas classes onde objetos de uma classe estão associados a objetos de outra classe. A multiplicidade define quantas instâncias de uma classe podem estar associadas a instâncias de outra.

Person tem uma associação com Car, indicada pela presença do atributo car em Person.

A multiplicidade aqui é 1:1, significando que uma pessoa pode ter um carro, e cada carro está associado a uma pessoa.

Diagrama de Classes



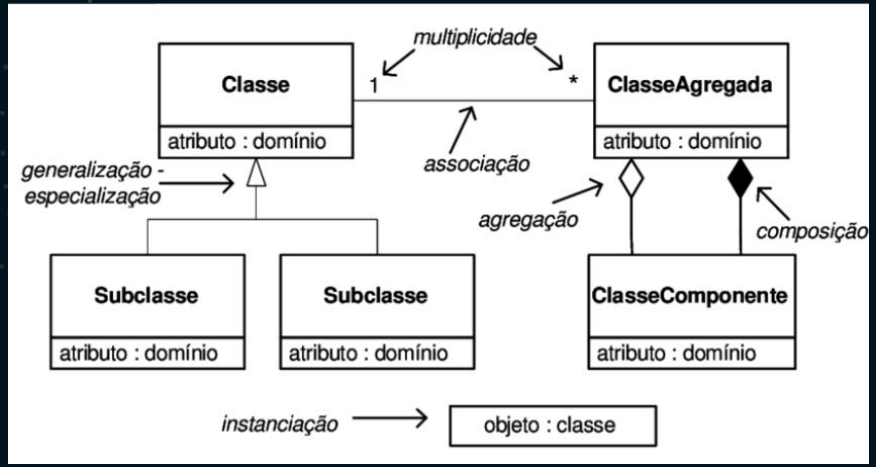
Agregação:

A agregação é um tipo especial de associação que representa uma relação todo/parte. O todo (classe agregadora) contém partes (classe agregada), mas as partes podem existir independentemente do todo.

Team agrega Player, indicando que um time é composto por vários jogadores.

Player pode existir independentemente de Team, ou seja, jogadores podem existir fora do contexto de um time específico.

Diagrama de Classes



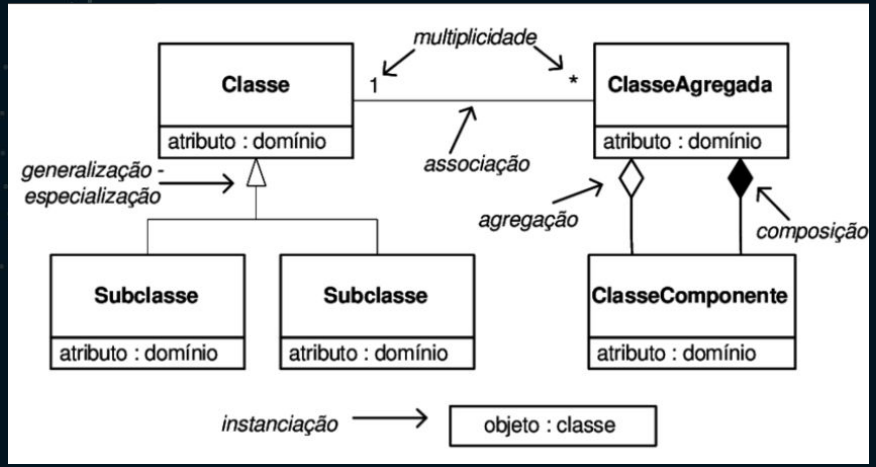
Composição

A composição é uma forma mais forte de agregação. Indica uma relação de contenção onde a parte (classe componente) não pode existir sem o todo (classe composta). Se o todo é destruído, as partes também são.

House é composta por Room. Os quartos são criados e adicionados à casa.

Se a casa for destruída, os quartos também deixam de existir, indicando a forte relação de dependência.

Diagrama de Classes



Instanciação

A instanciação é o processo de criação de objetos a partir de classes. Representa a criação de uma instância de uma classe no tempo de execução.

Criamos instâncias das classes Dog e Cat, e chamamos seus métodos específicos.

Associamos um Car a um Person e acessamos seus atributos.

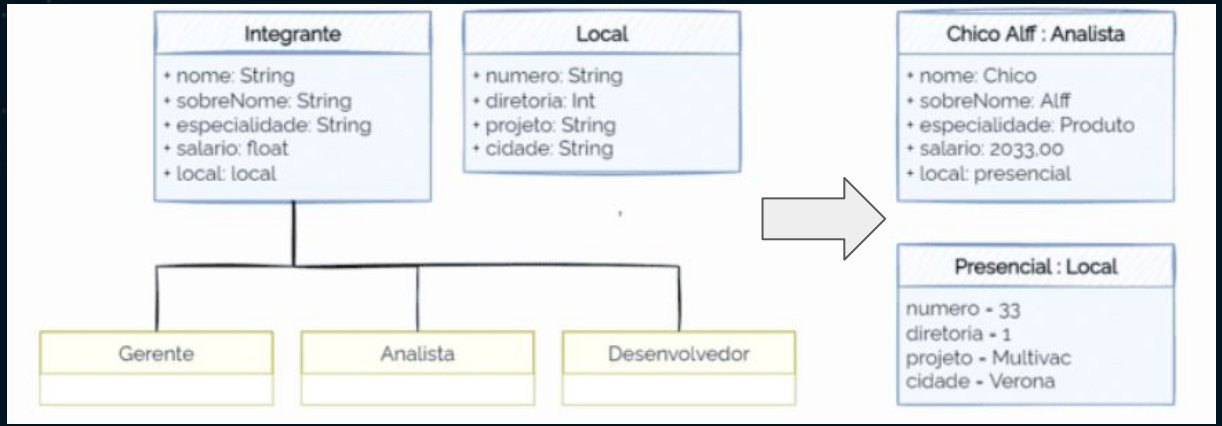
Criamos uma Team com uma lista de Player demonstrando agregação.

Criamos uma House e adicionamos Room demonstrando composição.

Instanciamos um Animal genérico para mostrar a criação de objetos a partir de uma classe.

Diagramas de Classes + Objetos

Mostra a relação entre objetos usando exemplos do mundo real e retrata um sistema em um determinado momento. Veja detalhes aqui.

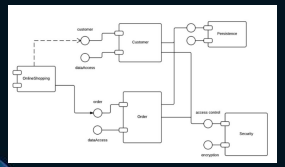


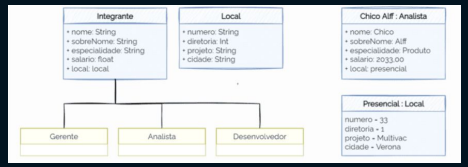
Exemplo

Diagramas UML Comportamentais

- Diagrama de Caso de Uso } Sempre …
- Diagrama de Atividade
- Diagrama de interação
  - Diagrama de comunicação / sequência
  - Diagrama de máquina de estados
  - Diagrama de tempo
  - Diagrama de Implementação
  - Diagrama de Pacotes

} Sistemas com grande complexidade





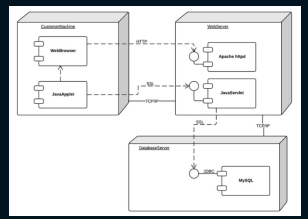
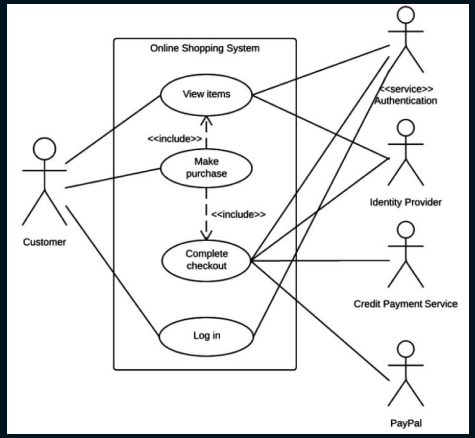


Diagrama de Casos de Uso

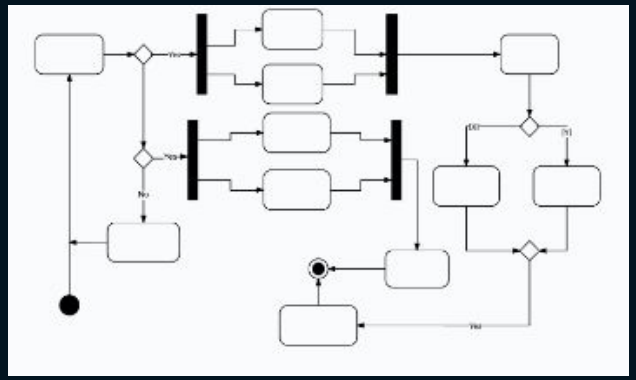
Descreve as interações entre o sistema e seus atores externos. Utilizado para identificar os requisitos funcionais do sistema e as funcionalidades que ele deve oferecer aos usuários. Veja detalhes aqui.



Exemplo

Diagrama de Atividades

Descreve o fluxo de controle entre diferentes atividades do sistema. Utilizado para modelar processos de negócio, fluxos de trabalho e algoritmos. Veja detalhes aqui.



Esquema genérico

Conhecimento adquirido ...

Até agora você compreendeu que existem diferentes diagramas UML para representar um Sistema.

Sugiro que você procure aprofundar seu conhecimento em UML, existem muitos detalhes!

Spring Boot

Prof: Prof Dra Iara Carnevale de Almeida

Aula: 06

Ferramenta Lucidchart

Vamos conhecer uma das ferramentas que permite a criação de Diagramas UML?

Chama-se 

Para você criar uma conta gratuita, clique aqui.

Spring Boot

BeTech.
Integrado

Prof: Prof Dra Iara Carnevale de Almeida

Aula: 07

Estudo de Caso

PROBLEMATIZAÇÃO - Uma escola está enfrentando desafios na gestão eficiente da frequência e das notas dos alunos. O processo atual é baseado em registros manuais e planilhas. Devido a isto, ocorrem problemas referente ao volume de informações e acompanhamento dos estudantes. Portanto, há inconsistências nos registros, dificuldades na análise de dados e insatisfação por parte dos pais e responsáveis.

Projeto de sistemas baseados em UML

PROBLEMAS

Registros propensos a erros - o processo destes registro de frequência e lançamento de notas é manual; aumentando o risco para erros e imprecisões nos registros.

Dificuldade na análise e interpretação dos dados - não existem ferramentas adequadas para grande volume de dados.

Comunicação com os pais/responsáveis dos alunos é limitada - dificuldade de acompanhar o progresso escolar.

Falta de integração de dados - informações de frequência e notas dos alunos estão dispersas em diferentes sistemas e planilhas, tornando difícil a integração e consulta centralizada.

Projeto de sistemas baseados em UML

SOLUÇÕES

Registro de Frequência e Notas que permita o registro da frequência dos alunos e lançamento de notas pelos professores.

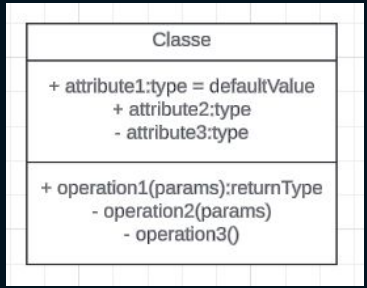
Ferramentas para análise de dados para visualizar e interpretar facilmente as informações de frequência e notas dos alunos.

Estabelecer um canal de comunicação com os pais/responsáveis fornecendo informações de frequência e notas dos alunos.

Centralizar e Integrar todas as informações dos alunos em um único sistema integrado, facilitando a consulta e atualização.

Diagrama de Classes

CLASSE - atributos e suas relações



Aluno - nome, matrícula, data de nascimento, endereço

Professor - nome, disciplina

Disciplina - nome, código, carga horária

Turma - ano, período, disciplina, professor

Registro Frequência - aluno, turma, data, presença (booleano)

Avaliação - tipo{bimestral, substituição, recuperação}, aluno, disciplina, valor

Responsável Aluno - nome, telefone, email, aluno

* operações ***get*** e ***set*** para todos os atributos de todas as classes.

Diagrama de Classes

MULTIPLICIDADE e RELAÇÃO entre CLASSES

um ou mais Alunos está em uma Turma

um ou mais Alunos tem um Responsável Aluno

um Professor ministra uma ou mais Disciplinas

uma Turma possui uma ou mais Disciplinas

uma Turma tem um ou mais Professores

um Aluno tem um ou mais Registro Frequência

um Aluno tem uma ou mais Nota

Conhecimento adquirido ...

Espero que você tenha compreendido melhor o diagrama de classes UML.

Sugiro que você continue a se aprofundar seu conhecimento em UML, muitos são os detalhes!

Procure construir o diagrama de classes no Lucidchart

Análise de Sistemas e Padrões de Projeto

BeTech. Integrado

Prof: Prof Dra Iara Carnevale de Almeida

Aula: 08

Estudo de Caso (cont)

Relembrando ...

PROBLEMATIZAÇÃO - Uma escola está enfrentando desafios na gestão eficiente da frequência e das notas dos alunos. O processo atual é baseado em registros manuais e planilhas. Devido a isto, ocorrem problemas referente ao volume de informações e acompanhamento dos estudantes. Portanto, há inconsistências nos registros, dificuldades na análise de dados e insatisfação por parte dos pais e responsáveis.

Projeto de sistemas baseados em UML

SOLUÇÕES

Registro de Frequência e Notas que permita o registro da frequência dos alunos e lançamento de notas pelos professores.

Ferramentas para análise de dados para visualizar e interpretar facilmente as informações de frequência e notas dos alunos.

Estabelecer um canal de comunicação com os pais/responsáveis fornecendo informações de frequência e notas dos alunos.

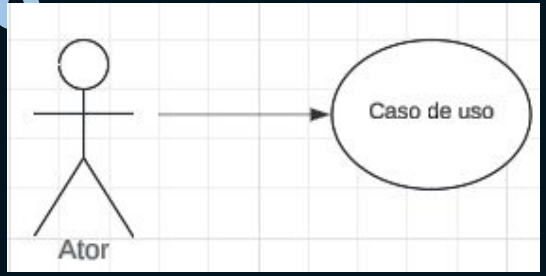
Centralizar e Integrar todas as informações dos alunos em um único sistema integrado, facilitando a consulta e atualização.

Diagrama de Caso de Uso

ATORES

- Administrativo: fazer a gestão de todas as informações; são o coordenador pedagógico, diretor e secretária;
- Professor: fazer a gestão das frequências e notas de seus alunos;
- Aluno e Pais/Responsáveis: consultar informações do aluno.

Diagrama de Caso de Uso



Casos de Uso e Ator

Gerenciar Alunos
Gerenciar Professores
Gerenciar Disciplinas
Gerenciar Turmas
} Administrativo

Gerenciar Frequência
Gerenciar Notas
} Professor

Consultar Frequência e Notas
} Aluno; Pai/Responsável

Conhecimento adquirido ...

Espero que você tenha compreendido melhor o diagrama de caso de uso UML.

Sugiro que você continue a se aprofundar seu conhecimento em UML!

Procure construir o diagrama de caso de uso no Lucidchart

Spring Boot

BeTech. Integrado

Prof: Prof Dra Iara Carnevale de Almeida

Aula: 9

Especificação do Sistema

Os requisitos (funcionais e não funcionais) são traduzidos para uma especificação mais técnica e detalhada para posterior construção do projeto e implementação do sistema.

- identificação das principais funcionalidades do sistema,
- definição das interfaces do usuário e do sistema
- modelagem do comportamento do sistema (requisitos funcionais) e especificação de desempenho, segurança e qualidade (requisitos não funcionais).

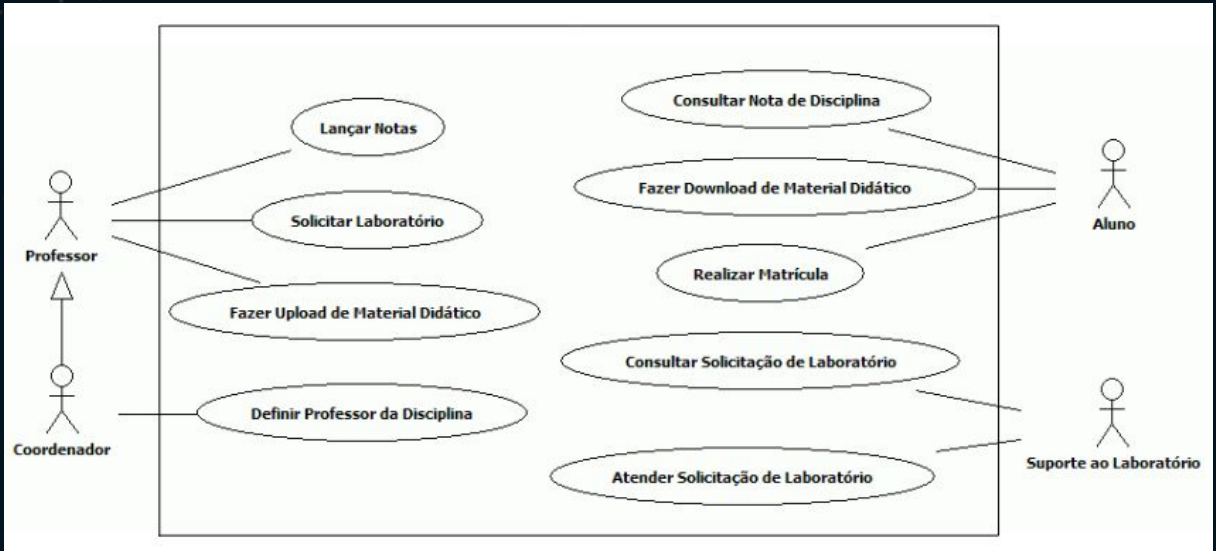
Especificação do Sistema

As técnicas utilizadas incluem a criação de diagramas de casos de uso, diagramas de sequência, diagramas de atividades, especificações de interface e a definição de modelos de dados.

Serve como base para o desenvolvimento e teste do software.

Uma especificação clara e abrangente do sistema garante que os *stakeholders* tenham uma boa compreensão sobre os requisitos e expectativas do sistema.

Detalhando Casos de Uso



Conhecimento adquirido ...

Você compreendeu um pouco mais sobre Análise e Especificação de Sistemas?

Sugiro que você continue a aprofundar seu conhecimento e procure conhecer melhor as técnicas apresentadas!!

Design Patterns

BeTech.
Integrado

Prof: Wagner Monteverde

Padrões de Design de Software

O que são Padrões de Design?

- **Definição breve:** Soluções típicas para problemas comuns em design de software.
- **Propósito:** Facilitar a comunicação entre desenvolvedores e refinar o processo de software.

Padrões de Design de Software

Por que aprender Padrões de Design?

- **Melhoria da Comunicação:** Termos universais que todos os desenvolvedores podem entender.
- **Código Reutilizável:** Promove a reutilização de software, evitando soluções repetitivas e redundantes.
- **Soluções Provadas:** Abordagens testadas e aprovadas para problemas frequentemente encontrados.

Padrões de Design de Software

Tipos de Padrões?

- **Padrões de Criação:**

  Padrões de criação são ferramentas de design que abstraem o processo de instanciação, tornando um sistema independente de como seus objetos são criados, compostos e representados.

Padrões de Design de Software

Padrão de Criação: Singleton

Definição: O Singleton é um padrão de design que restringe a instanciação de uma classe a um único objeto. Este padrão é usado quando um único objeto é necessário para coordenar ações em todo o sistema.

Padrões de Design de Software

Aplicações Comuns:

- Gerenciamento de conexões de banco de dados.
- Configurações globais de aplicativos.

Problemas Resolvidos:

- Garante que uma classe tenha apenas uma instância, e proporciona um ponto de acesso global a essa instância.
- Economiza recursos, como conexões de banco de dados.

Padrões de Design de Software

Padrão de Criação: Factory Method

Definição: O Factory Method é um padrão que define uma interface para criar um objeto, mas permite que as classes que implementam a interface decidam qual classe instanciar.

Padrões de Design de Software

Aplicações Comuns:

- Sistemas de gestão que precisam de flexibilidade na criação de diferentes tipos de objetos.
- Quando a implementação de uma interface ou uma classe é esperada para mudar frequentemente.

Problemas Resolvidos:

- Encapsula a criação de objetos e separa a implementação da interface do uso da classe.

Padrões de Design de Software

Padrão de Criação: Builder

- **Definição:** O Builder é um padrão de construção que permite a criação de diferentes representações de um objeto complexo, construindo-o passo a passo.

Padrões de Design de Software

Aplicações Comuns:

- Criação de objetos complexos com múltiplas configurações.
- Construção de objetos que devem ser imutáveis após a criação.

Problemas Resolvidos:

- Separa a construção de um objeto complexo de sua representação, permitindo que o mesmo processo de construção crie diferentes representações.

Padrões de Design de Software

Tipos de Padrões

- **Padrões Estruturais:** Os padrões estruturais são padrões de design que simplificam o design ao estabelecer maneiras de organizar entidades de modo que estas possam formar estruturas maiores, mantendo a flexibilidade e a eficiência da arquitetura do software.

Padrões de Design de Software

Padrão Estrutural: Adapter

- **Definição:** Permite que interfaces incompatíveis trabalhem em conjunto. O Adapter converte a interface de uma classe em outra interface que o cliente espera encontrar.

Padrões de Design de Software

Aplicações Comuns:

- Integração de sistemas que utilizam interfaces distintas.
- Compatibilidade entre dados ou tipos de arquivos não suportados originalmente por uma aplicação.
-

Exemplo Prático de Código:

Suponha que temos um sistema de reprodução de mídia que só reproduz MP4, mas precisamos suportar também arquivos VLC.

Padrões de Design de Software

Padrão Estrutural: Facade

- **Definição:** Fornece uma interface simplificada para um conjunto complexo de classes, uma biblioteca ou um framework.

Aplicações Comuns:

- Simplificação de interfaces complexas de sistemas.
- Redução de dependências entre sistemas e clientes.

Padrões de Design de Software

Padrão Estrutural: Proxy

- **Definição:** Fornece um substituto ou representante de outro objeto para controlar o acesso a este.

Aplicações Comuns:

- Controle de acesso a recursos que são custosos em termos de tempo ou memória.
- Proteção de objetos subjacentes de acessos diretos.

Padrões de Design de Software

Exemplo Prático de Código:

- Suponha que temos um sistema onde documentos confidenciais são acessados por diferentes tipos de usuários (administradores e usuários comuns). Queremos garantir que somente administradores possam acessar certos documentos sensíveis.

Padrões de Design de Software

Tipos de Padrões?

- **Padrões Comportamentais:** Os padrões comportamentais são padrões de design que gerenciam as interações e responsabilidades entre objetos, facilitando a comunicação e a distribuição de comportamentos em um sistema.

Padrões de Design de Software

Padrão Comportamental: Observer

- **Definição:** Permite que um objeto (o "subject") notifique uma lista de objetos observadores sobre mudanças em seu estado.

Aplicações Comuns:

- Sistemas de notificação onde mudanças no estado de um objeto precisam ser comunicadas a outros objetos.

Padrões de Design de Software

Exemplo Prático de Código:

Imagine um sistema de segurança residencial onde diferentes serviços, como o departamento de polícia e os bombeiros, precisam ser notificados imediatamente quando um alarme é disparado (por exemplo, em caso de invasão ou incêndio).

Padrões de Design de Software

Padrão Comportamental: Strategy

- **Definição:** O padrão Strategy define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis. O Strategy permite que o algoritmo varie independentemente dos clientes que o utilizam.

Aplicações Comuns:

- Sistemas que requerem flexibilidade nos métodos de execução de tarefas específicas.
- Aplicações que necessitam adaptar seu comportamento sem alterar seus objetos.

Padrões de Design de Software

Exemplo Prático de Código:

Suponha que temos um aplicativo de pagamento que precisa suportar diferentes métodos de pagamento (cartão de crédito, PayPal, etc.).

Padrões de Design de Software

Padrão Comportamental: Command

- **Definição:** O padrão Command transforma um pedido em um objeto independente que contém toda a informação necessária para executar a ação.

Aplicações Comuns:

- Operações Parametrizadas com diferentes conjuntos de parâmetros, objetos de comando podem ser configurados com esses parâmetros e usados para executar a ação.
- O padrão Command é ideal para implementar funcionalidades de desfazer e refazer em aplicativos, como editores de texto ou de imagem

Padrões de Design de Software

Exemplo Prático de Código:

Em um sistema de automação residencial, os usuários podem querer automatizar tarefas como acender as luzes, ajustar o termostato ou reproduzir música. O padrão Command é perfeito para esse cenário porque permite encapsular cada ação como um objeto de comando, facilitando a programação de tarefas e a interação com diferentes dispositivos de forma modular e expansível.

Conhecimento adquirido ...

Padrões de Design de Software