



Desenvolvimento de Sistemas

Regras de negócio

Conceito e características

Antes de se falar sobre regras de negócio, é preciso compreender o que é um negócio.

Um negócio pode ser entendido essencialmente como o que uma organização faz.

Um negócio pode ser a comercialização de um produto ou a prestação de um serviço, por exemplo. As partes de um negócio podem ser entendidas como etapas ou processos.

É possível uma empresa mais antiga **viver sem software**, porém, ela não consegue **viver sem regras de negócio**.

A simples comercialização de cachorros-quentes em uma carrocinha pode envolver vários processos. O entendimento dos processos permite que os envolvidos tenham entendimento sobre o negócio, ou seja, entendam como funciona a produção e a venda do cachorro-quente e também faz com que o negócio atinja o seu objetivo, que, na prática, resume-se à venda de cachorros-quentes.

Ainda no exemplo da carrocinha, os processos que consistem no preparo de um cachorro-quente podem ser definidos como uma regra do negócio.





Regras de negócio em TI são processos que determinam o funcionamento da empresa, sendo frequentemente aplicadas no contexto da arquitetura de *softwares*. Para consolidá-las, é preciso criar uma documentação, avaliar o fluxo de trabalho e listar necessidades. Basicamente, são as diretrizes do seu negócio, ou seja, a parte fundamental da sua estratégia.

Regra de negócio é o padrão que condiciona o funcionamento do negócio, que guia o comportamento deste e o define como: **o que, onde, quando, por que e como será feito.**

As regras de negócio definem como será o comportamento da empresa perante as mais variadas situações, que podem influenciar positiva ou negativamente a imagem da empresa, os investimentos, o perfil dos clientes, ou seja, são as regras de negócio bem elaboradas e definidas que fazem uma empresa atuante no mercado.





BPM – Gerenciamento de processos de negócios

Só é possível melhorar aquilo que se pode gerenciar. É nisso que o BPM (Business Process Management ou Gerenciamento de Processos de Negócios) tem o foco.

BPM é uma abordagem de gerenciamento adaptável, com o foco na modelagem dos processos. Dentro do BPM, não existe conclusão, pois as etapas são revisadas e melhoradas constantemente em busca de um alinhamento entre tudo o que está sendo executado e o resultado que se deseja alcançar.

Entender o BPM é fundamental para ajudar no crescimento de uma empresa. É por meio dele que são reconhecidos os processos que estão em execução, sendo também possível medi-los e gerenciá-los, para uma possível realização de melhorias e evolução nos processos.

As vantagens do BPM para uma empresa são:

- ◆ Melhora contínua e transparência dos processos, permitindo que as organizações sejam mais eficientes
- ◆ Aumento de produtividade por trazer benefícios como a eliminação de processos redundantes
- ◆ Redução dos custos
- ◆ Automatização de tarefas
- ◆ Evolução contínua por se tratar de um processo de negócios em constante desenvolvimento

Para saber mais sobre BPM, pesquise sobre “BPM – Regra de negócios”.



Exemplos de regras de negócio

As regras de negócio em um *software* podem ajudar empresas a tomarem decisões de modo ágil e seguro.

Mas muitas pessoas têm dúvidas sobre como isso funciona na prática. Para esclarecer, analise a seguir alguns exemplos de regras de negócio:

Regra de negócio para um *pet shop*

- ◆ Ofertar descontos exclusivos para fidelizar clientes

Caso o cliente traga o seu animalzinho com frequência, ofertar descontos exclusivos em serviços como consulta médica, banho e tosa, entre outros. Os descontos podem ser aplicados também em produtos que o *pet* poderá usar, como, por exemplo, se for cachorro, desconto em ração, biscoitos e coleiras.

Regra de negócio em um *e-commerce*

- ◆ Uma regra simples para detectar futuras fraudes seria:

Se o CPF (Cadastro de Pessoas Físicas) do cliente não tiver restrições e a operadora indicar que o cartão tem limite disponível, liberar a compra.

Regra de negócio de um escritório contábil

- ◆ Definição de critérios para concorrer a promoção interna de gerente:



Para ser admitido na lista de prováveis futuros gerentes da empresa, o candidato deve ter no mínimo cinco anos de casa, falar inglês fluentemente, ter ao menos uma pós-graduação e já ter feito no mínimo três viagens internacionais representando a empresa. Além disso, o candidato deve contar com ótimas avaliações de desempenho.

Pode-se também citar alguns exemplos de regras de negócios em processos comuns dentro de uma empresa, como:

- ◆ **Cálculos de preços:** como é feito esse cálculo e o que é considerado.
- ◆ **Aprovação de orçamentos:** quais são os critérios avaliados.
- ◆ **Liberação de empréstimos:** qual é o perfil da pessoa ou da empresa que deseja contratar o empréstimo.
- ◆ **Benefícios oferecidos:** qual são os benefícios que a empresa oferece e por que são oferecidos.

Requisitos funcionais e não funcionais

Quando se fala em regra de negócio, que é a definição da forma de fazer o negócio, é preciso saber “o que fazer”. É nessa parte que entram em ação os requisitos funcionais e não funcionais que auxiliarão na maneira pela qual serão atingidos os objetivos principais e secundários.

De modo geral, requisitos são instruções que definem como atingir o objetivo do negócio. São solicitações, desejos e necessidades. Os requisitos serão as funções de apoio para que os processos da empresa se realizem, sejam eles manuais ou informatizados.

É de extrema importância que os requisitos estejam alinhados com o objetivo do negócio e com o propósito final da empresa em relação ao seu papel. Para exemplificar

melhor, imagine uma empresa de desenvolvimento de *software*, destacando os seguintes objetivos.

Objetivo principal

- ◆ Ser referência no ramo de desenvolvimento de *software*

Objetivos secundários

- ◆ Ter profissionais comprometidos
- ◆ Ter uma sede para a empresa, com espaço amplo e equipamentos de boa qualidade
- ◆ Oferecer sistemas que se adaptam a cada necessidade do cliente

Note, nesse exemplo, como tudo deve estar alinhado: os objetivos secundários são os passos para que a empresa seja referência no seu ramo de atuação. Não basta somente definir a regra do negócio, que, nesse exemplo, pode ser representado pelo objetivo principal; é preciso também saber “como fazer”, que é onde se deve definir os requisitos para o negócio.

Como exemplo de um requisito, pode ser citado o cadastro de clientes e até mesmo as funções mais complexas, como relatórios de gestão.

Quando uma empresa solicita o desenvolvimento de um *software*, os requisitos dele devem representar o objetivo do negócio, logo, devem seguir as regras que regem esse negócio. Caso contrário, o produto final não cumprirá com a missão principal.

Entre as principais funções de um *software* está o processo de automatização e agilidade dos negócios. Isso ocorre com as funcionalidades que realizarão os requisitos funcionais, que definem as necessidades da empresa em termos de funcionalidade, ou seja, a maneira com que os usuários realizam tarefas dentro de uma empresa segue um

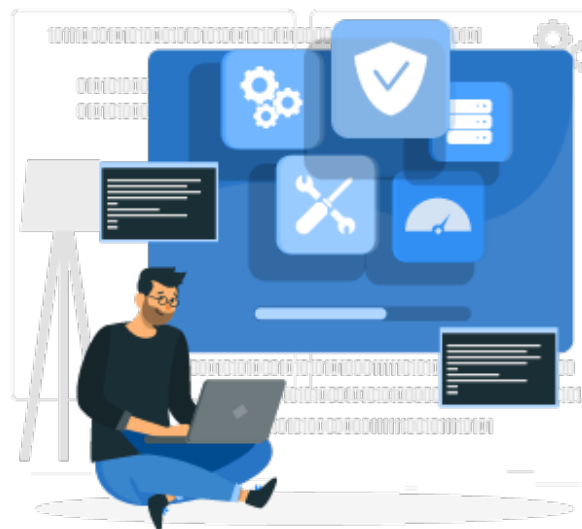
conjunto de regras que o sistema a ser criado deve contemplar.



Requisitos funcionais

Requisito funcional é todo aquele processo que define o funcionamento percebido do sistema pelos usuários. São problemas e necessidades que devem ser atendidos e resolvidos pelo *software* por meio de funções ou serviços. Em resumo, pode-se dizer que é o que realmente o cliente espera que o *software* faça.

Estes são alguns exemplos de requisitos funcionais comuns em diferentes tipos de sistemas:



- ◆ Cadastro de dados dos clientes
- ◆ Realização da baixa de mercadoria em estoque
- ◆ Realização das compras de mercadorias
- ◆ Produção do relatório das mercadorias vendidas

- ◆ Inserção de dados em um formulário
- ◆ Busca de pratos específicos dentro de um cardápio
- ◆ Realização de compras
- ◆ Comunicação com um atendente

Pode-se considerar como função tudo o que for relacionado a ações no sistema. Os requisitos funcionais devem ser descritos objetivamente e não de modo muito complexo para que o objetivo final, que é o desenvolvimento de *software*, seja atendido com qualidade.

Requisitos não funcionais

Já os **requisitos não funcionais** são as qualidades globais do sistema relacionadas como manutenção, usabilidade, desempenho, custos e várias outras. Normalmente, esses requisitos são descritos informalmente, tornando-se difíceis de validar.

São exemplos de requisitos não funcionais comuns:





- ◆ O tipo do sistema operacional
- ◆ O tipo de banco de dados a ser utilizado pelo sistema
- ◆ Em quais dispositivos o *software* poderá ser usado
- ◆ A linguagem de programação que deverá ser utilizada para o desenvolvimento do *software*
- ◆ A segurança (autenticação por meio de *login*)

É possível também dividir em três categorias principais os requisitos não funcionais. São elas: **requisitos de produto final**, **organizacional** e **externo**.

Requisitos de produto final

São aqueles que estão ligados ao comportamento do *software*, como: tempo de resposta do sistema, velocidade de execução, conexão, portabilidade, consistência, segurança, taxa de erros e confiabilidade.

Requisitos organizacionais

Estão relacionados aos padrões da organização, ou seja, o *software* deve ser desenvolvido de acordo com as políticas e definições da empresa para garantir que o produto final desenvolvido esteja de acordo com as normas empresariais e alinhado com a regra de negócio da empresa. Alguns exemplos de requisitos organizacionais são: infraestrutura, sistema operacional compatível, conexão, criptografia usada pela empresa e linguagem de programação requisitada pela empresa.

Requisitos externos



Estão relacionados a qualquer tipo de agente externo ao *software*, ou seja, qualquer aspecto que não corresponde diretamente ao produto, mas que pode causar impacto no seu funcionamento. Entre os principais, estão: localização geográfica em que o *software* será usado, legislação, sistemas e política de proteção de dados.

Antes de seguir com o conteúdo, descreva (em uma folha de papel ou documento de texto) algumas regras de negócio para uma pequena empresa do ramo de *pet shop*, e também os requisitos funcionais e não funcionais de um *software* que faça o agendamento de consultas, de banho e tosa ou de outro procedimento. Esse *software* também deve emitir um alerta após 15 dias da última vez que o *pet* foi até o *pet shop*, para que o funcionário envie uma mensagem a fim de lembrar o tutor de levar seu *pet* para realizar algum procedimento.

Técnicas para a extração de requisitos





Um ponto muito importante para um bom desenvolvimento de *software* é o levantamento de requisitos, sendo essa uma das atividades mais importantes e que deve ser revista várias vezes dentro do processo de desenvolvimento do *software*.

Sommerville (2003) propõe um processo genérico de levantamento e análise que contém as seguintes atividades:

Compreensão do domínio Os analistas devem desenvolver sua compreensão do domínio da aplicação (pode ser chamada de camada de negócios também e é onde estão localizadas as classes que fazem parte da descrição dos problemas). **Coleta de requisitos** É o processo de interagir com as partes interessadas do sistema, também chamadas de *stakeholders*, para descobrir os requisitos delas. A compreensão do domínio se desenvolve mais durante essa atividade. **Classificação** Essa atividade considera o conjunto não estruturado dos requisitos e os organiza em grupos coerentes. **Resolução de conflitos** Quando as múltiplas partes interessadas estão envolvidas, os requisitos apresentarão conflitos. Essa atividade tem por objetivo solucionar esses conflitos. **Definição das prioridades** Em qualquer conjunto de requisitos, alguns serão mais importantes do que outros. Esse estágio envolve interação com a parte interessada para a definição dos requisitos mais importantes. **Verificação de requisitos** Os requisitos são verificados para descobrir se estão completos e consistentes e se estão em concordância com o que a parte interessada deseja do sistema.

A extração de requisitos é o processo de transformação das ideias que estão na mente do usuário (entrada) em um documento formal (saída).

Durante esse processo, é muito importante que a equipe de desenvolvimento entenda o que o cliente quer, como ele quer, ou seja, qual é o propósito dele para o uso do *software*.

A saída do processo de extração de requisitos é um documento de especificação dos requisitos, que descreve o que o produto a ser desenvolvido deverá fazer, sem, entretanto, descrever como deve ser feito.

O modo como deve ser feito é muito técnico e fica a cargo da equipe de

desenvolvimento. Não há necessidade de envolver as partes interessadas nesse processo.

Processo de extração de requisitos

Durante o processo de extração de requisitos podem ser destacadas quatro etapas, que são: entendimento do domínio, extração e análise de requisitos, especificação e validação.

Observe a seguir cada uma dessas etapas:

Entendimento do domínio

Nessa fase, os desenvolvedores do sistema devem entender o domínio da aplicação como um todo, compreendendo seu foco principal.

Extração e análise de requisitos

Nessa fase acontece a descoberta, a revelação e o entendimento dos requisitos por meio da interação com a parte interessada.

Esse processo é muito importante, pois saber o verdadeiro foco do cliente é ser assertivo em como será o processo de desenvolvimento.

Especificação dos requisitos

Nessa etapa ocorre o armazenamento dos requisitos em uma ou mais formas, incluindo língua natural, linguagem semiformal ou formal, representações simbólicas ou gráficas.

É importante lembrar que esse material será mostrado ao cliente, então deve ser de fácil entendimento. Toda a parte técnica do desenvolvimento do software deve ser conversada entre a equipe de desenvolvimento somente.

Validação dos requisitos

Nessa etapa é feita a verificação dos requisitos, visando a determinar se estão completos e atendendo às necessidades e aos desejos do cliente.

Métodos para a extração e análise de requisitos

Existem métodos e técnicas específicas que auxiliam no processo de extração de requisitos, facilitando o entendimento da equipe de desenvolvimento. Podem ser divididos em técnicas informais e formais.

As técnicas informais são baseadas em comunicação estruturada (mapeamento de interações e respostas possíveis que podem ser utilizadas para comunicações humanas ou de máquinas) e interação com o usuário, questionários etc.

Destacam-se o método JAD (*joint application design*), o *brainstorming*, as entrevistas e a técnica Pieces (desempenho – ou *performance* –, informação e dados, economia, controle, eficiência e serviços).

JAD (*joint application design*)

Também chamado de método de projeto iterativo, substitui as entrevistas individuais por reuniões de grupo, nas quais participam os representantes dos envolvidos no projeto. Essas reuniões são intensas e levam tipicamente de um a três dias.


É um método destinado a acelerar o projeto de sistema. Toda a equipe de desenvolvimento (gerentes, analista e também o usuário), dividida em grupos, faz a projeção do *software*.

Os benefícios do JAD são:

- ◆ Maior produtividade
- ◆ Maior qualidade
- ◆ Trabalho em equipe
- ◆ Custos mais baixos de desenvolvimento e manutenção

A forma mais produtiva de decisão em grupo é aquela obtida por consenso. Deve-se entender consenso não como uma unanimidade de opiniões, mas sim como a concordância entre os participantes de que a solução encontrada é a melhor para o grupo e não fere convicções ou valores essenciais.


Antes de as reuniões iniciarem, devem ser definidos os seguintes papéis, que podem ser cumulativos:

- ◆ Facilitador: coordena a sessão JAD.
-  ◆ Documentador (pode ser um analista ou programador): registra todas as conclusões do grupo.
- ◆ Observador (deverá ser uma pessoa perspicaz).
- ◆ Indicador de assunto: garante que todos os pontos sejam tratados e deve ser um envolvido no projeto, que conheça bem o assunto.
- ◆ Representante dos envolvidos (usuários): representa as áreas envolvidas (devem ser escolhidas as pessoas-chave, independentemente do nível hierárquico).
- ◆ Gerente do projeto: deve acompanhar as sessões, uma vez que o sucesso ou o fracasso do projeto será sua responsabilidade.
- ◆ Especialista: conhece os projetos da organização, que tenham interface com o projeto em estudo.

Brainstorming

Chamado também de “tempestade de ideias”, pode-se definir o *brainstorming* como sendo uma técnica que auxilia a potencializar a criatividade e a encontrar soluções para determinados problemas. Pode ser feita uma reunião, presencial ou *on-line*, na qual cada participante dela pode expor suas ideias e sugestões, com total liberdade, e também debater sobre as ideias e sugestões dos demais participantes.

Um *brainstorming* deve ter como métodos:

- ◆ Divulgação clara dos objetivos
-  ◆ Geração de ideias (entre 20 e 60 minutos)
- ◆ Intervalo para relaxamento
- ◆ Estudo detalhado de cada ideia

Entrevistas

Entrevistar não é somente fazer perguntas, é uma técnica estruturada que pode ser aprendida e na qual os desenvolvedores podem ganhar proficiência com o treino e a prática.

A entrevista consiste em quatro fases: identificação dos candidatos para a entrevista, preparação para uma entrevista, condução da entrevista e finalização da entrevista.

Pieces

Esta técnica é para os desenvolvedores inexperientes, que apresentam dificuldades em como e o que perguntar para extrair os requisitos do cliente.

A técnica Pieces ajuda a resolver esses problemas, pois fornece um conjunto de categorias de problemas que auxiliam o engenheiro de requisitos a estruturar o processo de extração de requisitos.

Em cada categoria existem várias questões que o desenvolvedor deve explorar com o usuário.

As seis categorias são:

- ◆ **Performance** – Reflete o que o usuário espera.
- ◆ **Informação** – Tipo de acesso às informações (relatório, funções *on-line*) que inclui a quantidade de informação oferecida pelo *software*.
- ◆ **Economia** – Custo de usar um produto de *software*, processadores, armazenagem e conexão.
- ◆ **Controle** – Restrições de acesso ao sistema, acesso a algumas informações e habilidade de monitorar o comportamento do sistema.
- ◆ **Eficiência** – Evitar coletar o mesmo dado mais de uma vez e armazená-lo em espaço múltiplo.
- ◆ **Serviços** – Refere-se a que tipo de serviços os usuários necessitam que o *software* realize.

“Olá, meu nome é Manoel e tenho um empreendimento no ramo de pizzarias.

Tenho um ambiente que suporta até 20 mesas, para 4 pessoas cada. Tenho funcionários capacitados, tanto na cozinha quanto no salão. Ainda trabalho com o sistema de ‘anotar’ para os clientes que conheço, mas muitas vezes demoro a receber ou nem recebo o valor que foi gasto por eles. Comecei há pouco tempo a trabalhar com o sistema de cartão de débito e crédito, e desde então as coisas têm melhorado.

A minha dificuldade está na gestão financeira: tudo que gasto e recebo fica somente anotado em um caderno e, muitas vezes, na agitação do dia a dia, acabo não anotando o que recebi de um cliente, paguei para algum fornecedor e gastei com contas mensais. Meu neto comentou que empresas de desenvolvimento de *software* criam soluções para cada tipo de empresa e desenvolvem sua aplicação conforme a necessidade do cliente. Preciso de algo que me ajude a ter o controle da gestão financeira do meu negócio, para que eu sabia os meus ganhos diários e mensais assim como meus gastos nos mesmos períodos. Se for possível, eu também gostaria de cadastrar os meus clientes para informá-los sobre futuras promoções e também ter um plano de fidelidade, para que eu possa ver quanto esse cliente gastou. Assim, eu poderia acompanhar esses valores e,

quando um valor estipulado fosse atingido pelo cliente, este ganharia alguma 'recompensa'. Fico no aguardo do seu contato e aceito sugestões, pois sou um pouco leigo nessa questão de tecnologia”.

Com base nesse texto, encontre os requisitos funcionais e não funcionais para o desenvolvimento do *software* solicitado.

Requisitos funcionais

- ◆ Contabilizar, pelo sistema, os gastos e ganhos, diários e mensais
- ◆ Cadastrar clientes novos e fiéis e também funcionários
- ◆ Adicionar futuras promoções para clientes
- ◆ Ver quanto cada cliente gastou na loja
- ◆ Criar o sistema de recompensa de clientes fiéis

Requisitos não funcionais

- ◆ O sistema só pode ser acessado por usuários ADM
- ◆ É comportado no sistema Windows
- ◆ A paleta de cores do programa é igual à do logo da empresa
- ◆ O sistema é leve, ou seja, é capaz de rodar em qualquer máquina

UML e diagramas de caso de uso





A UML (*unified modeling language* ou linguagem de modelagem) é usada para visualização, especificação, construção, documentação e comunicação.

Pode-se dizer que a UML define os procedimentos que auxiliarão você a modelar e documentar os *softwares* que serão desenvolvidos.

Com toda informação extraída do cliente, por meio de sua demanda, foi possível entender melhor qual é a regra do seu negócio e como ele funciona. Foram identificados os requisitos funcionais e não funcionais para se ter uma visão ampla do sistema. Agora, é o momento de exemplificar isso utilizando alguma forma gráfica, e é aqui que se encaixam os diagramas de caso de uso.

Diagramas de caso de uso





Esse diagrama é de extrema importância, pois documenta o que o sistema faz ou ainda fará do ponto de vista do usuário, descreve **como os usuários interagem com o sistema (as funcionalidades do sistema)**, **facilita a organização dos requisitos desse sistema** e dá a **visão externa** do sistema. O conjunto de casos de uso deve ser capaz de comunicar ao usuário **a funcionalidade e o comportamento** do sistema.


Para criar um diagrama de caso de uso, usa-se um conjunto de símbolos e conectores específicos. Um diagrama bem feito ajuda muito a equipe de desenvolvimento a entender melhor os cenários em que o sistema interage com pessoas ou organizações, identifica as metas gerais do sistema, enfim, monta um escopo do sistema como um todo, para entendimento de todos.

O diagrama de caso de uso é uma ferramenta que auxiliará você a ter um melhor entendimento dos usuários do sistema e suas interações, porém, o diagrama não mostrará a ordem em que os passos devem ser executados. Ele demonstrará uma visão geral do sistema.

Como visto anteriormente, a UML é a linguagem de modelagem, na qual, com o uso de ferramentas, pode-se modelar o diagrama de caso de uso.

O caso de uso é representado por uma forma oval rotulada. Bonecos de palito representam os atores no processo e a participação do ator no sistema é modelada com uma linha entre o ator e o caso de uso. Para representar o limite do sistema, desenhe uma caixa em torno do próprio caso de uso.

O diagrama de caso de uso UML é ideal para:

- ◆ Representar as metas de interações entre sistemas e usuários
-  ◆ Definir e organizar requisitos funcionais no sistema
- ◆ Especificar o contexto e os requisitos do sistema
- ◆ Modelar o fluxo básico de eventos no caso de uso

Para fazer a modelagem UML, você pode usar ferramentas *on-line*, como o *site* Lucidchart. Pesquise por “Lucidchart diagrama de caso de uso” e faça o cadastro gratuito. Além do diagrama de caso de uso, é possível criar fluxogramas pela ferramenta também.

O diagrama de caso de uso é composto de quatro partes, basicamente:

Cenário

Sequência de eventos que acontecem quando um usuário interage com o sistema.

Ator

Usuário do sistema, ou melhor, um tipo de usuário.

Use case

É uma tarefa ou uma funcionalidade realizada pelo ator (usuário).

Comunicação

É o que liga um ator com um caso de uso.



Exemplos de diagramas de caso de uso

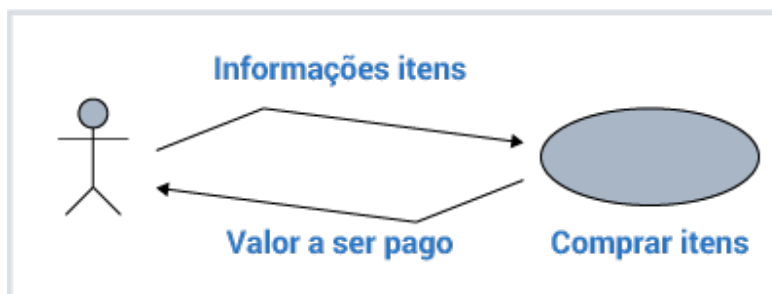
Confira agora alguns exemplos de diagramas de caso de uso.

Exemplo 1: Especificação do caso de uso “validar usuário”



O caso de uso inicia-se quando o sistema apresenta uma tela que pede ao cliente o seu cartão eletrônico. O cliente introduz o seu cartão e, com um pequeno teclado, digita sua senha. O cliente pode limpar a introdução da sua senha inúmeras vezes e digitar um novo número antes de pressionar o botão **Entrar**, que servirá para confirmar. O sistema lê a senha e a respectiva identificação do cartão e verifica se é válida. Se a senha for válida, o sistema aceita a entrada e o caso de uso termina.

Exemplo 2: Especificação do caso de uso “comprar itens”





1. Cliente chega a um operador de caixa com vários itens que deseja comprar.
2. O operador começa a nova venda.
3. O operador registra o identificador de cada item.
4. Sistema registra o item vendido
 - Preço do item e sua descrição são exibidos
 - Os passos 3 e 4 são repetidos, até que o operador indique o seu fim.
5. O sistema apresenta o total da venda.
6. O operador informa o cliente do total e solicita pagamento.
7. O cliente realiza o pagamento.
8. O operador registra o valor recebido no caixa.
9. Um recibo é gerado.
10. O operador entrega o troco ao cliente.
11. Cliente sai com os itens comprados e o recibo.

Exemplo 3: Cenário simples com definição de atores e ações de cada usuário

Atores: paciente, secretária e médico

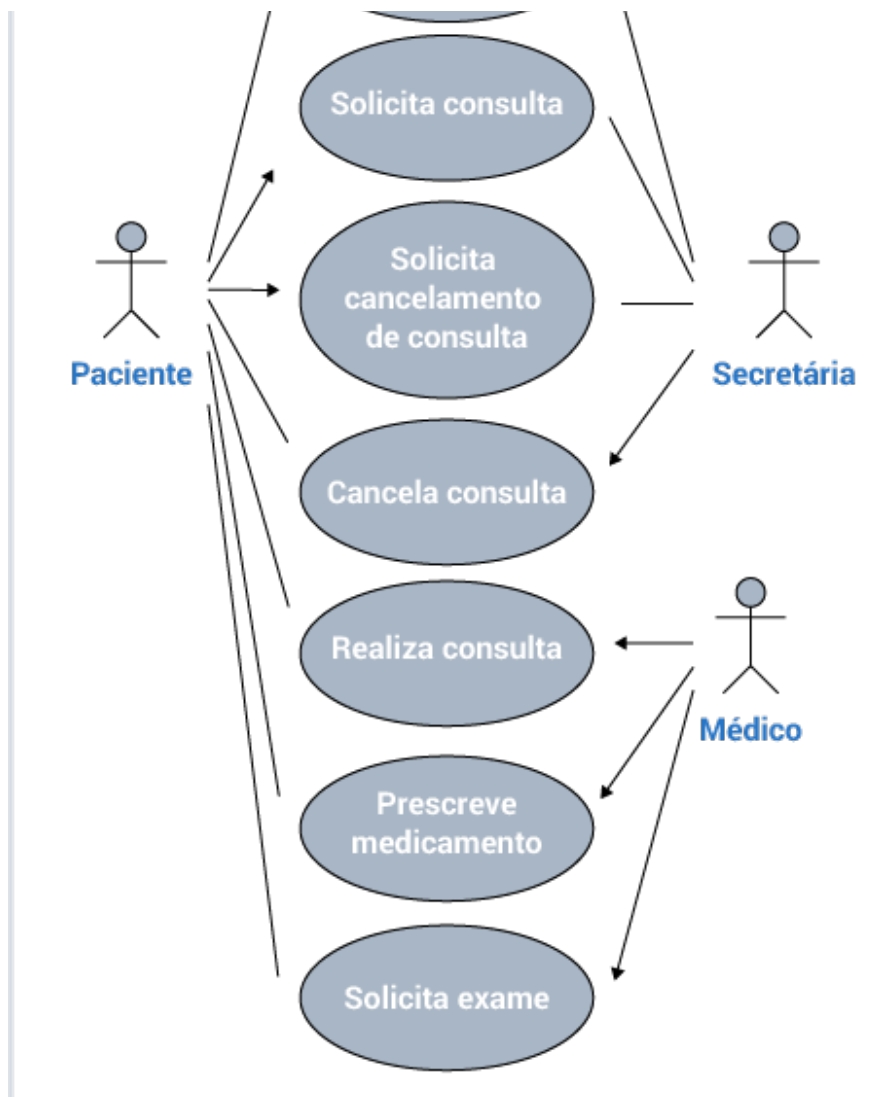
Ações de cada usuário:

Paciente: solicita consulta e solicita cancelamento de consulta

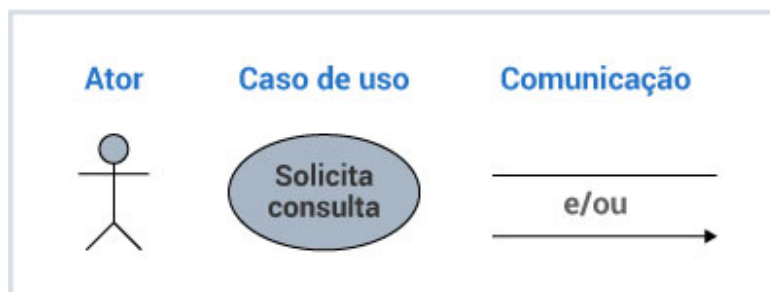
Secretária: consulta agenda, marca consulta e cancela consulta

Médico: realiza consulta, prescreve medicamento e solicita realização de exames





A seguir, veja a definição de algumas figuras utilizadas no diagrama:



Métodos para identificar um caso de uso

- ◆ Identificar os atores relacionados a um sistema ou organização
- ◆ Para cada ator, identificar os processos que eles iniciam ou dos quais pertencem/participam
- ◆ Identificar os eventos externos aos quais um sistema deve responder
- ◆ Relacionar os eventos a atores e a casos de uso

Encerramento

Com uma regra de negócio bem definida e alinhada com os princípios da empresa, certamente o negócio terá sucesso.

Todo o conteúdo é focado em ações que ajudem o programador ou a equipe de desenvolvimento a terem documentação e um norte. É importante deixar tudo claro (visualmente, por exemplo) para que todos saibam o que fazer, como fazer, quem deve fazer e quando deve ser feito.

A documentação de um projeto de *software* acaba sendo tão importante quanto a programação em si.

Requisitos de produto final

São aqueles que estão ligados ao comportamento do *software*, como: tempo de resposta do sistema, velocidade de execução, conexão, portabilidade, consistência, segurança, taxa de erros e confiabilidade.

Requisitos organizacionais

Estão relacionados aos padrões da organização, ou seja, o *software* deve ser desenvolvido de acordo com as políticas e definições da empresa para garantir que o produto final desenvolvido esteja de acordo com as normas empresariais e alinhado com a regra de negócio da empresa. Alguns exemplos de requisitos organizacionais são: infraestrutura, sistema operacional compatível, conexão, criptografia usada pela empresa e linguagem de programação requisitada pela empresa.

Requisitos externos

Estão relacionados a qualquer tipo de agente externo ao *software*, ou seja, qualquer aspecto que não corresponde diretamente ao produto, mas que pode causar impacto no seu funcionamento. Entre os principais, estão: localização geográfica em que o *software* será usado, legislação, sistemas e política de proteção de dados.

Entendimento do domínio

Nessa fase, os desenvolvedores do sistema devem entender o domínio da aplicação como um todo, compreendendo seu foco principal.

Extração e análise de requisitos

Nessa fase acontece a descoberta, a revelação e o entendimento dos requisitos por meio da interação com a parte interessada.

Esse processo é muito importante, pois saber o verdadeiro foco do cliente é ser assertivo em como será o processo de desenvolvimento.

Especificação dos requisitos

Nessa etapa ocorre o armazenamento dos requisitos em uma ou mais formas, incluindo língua natural, linguagem semiformal ou formal, representações simbólicas ou gráficas.

É importante lembrar que esse material será mostrado ao cliente, então deve ser de fácil entendimento. Toda a parte técnica do desenvolvimento do *software* deve ser conversada entre a equipe de desenvolvimento somente.



Validação dos requisitos

Nessa etapa é feita a verificação dos requisitos, visando a determinar se estão completos e atendendo às necessidades e aos desejos do cliente.



JAD (*joint application design*)

Também chamado de método de projeto interativo, substitui as entrevistas individuais por reuniões de grupo, nas quais participam os representantes dos envolvidos no projeto. Essas reuniões são intensas e levam tipicamente de um a três dias.

É um método destinado a acelerar o projeto de sistema. Toda a equipe de desenvolvimento (gerentes, analista e também o usuário), dividida em grupos, faz a projeção do *software*.

Os benefícios do JAD são:

- ◆ Maior produtividade
- ◆ Maior qualidade
- ◆ Trabalho em equipe
- ◆ Custos mais baixos de desenvolvimento e manutenção

A forma mais produtiva de decisão em grupo é aquela obtida por consenso. Deve-se entender consenso não como uma unanimidade de opiniões, mas sim como a concordância entre os participantes de que a solução encontrada é a melhor para o grupo e não fere convicções ou valores essenciais.

Antes de as reuniões iniciarem, devem ser definidos os seguintes papéis, que podem ser cumulativos:



- ◆ Facilitador: coordena a sessão JAD.
- ◆ Documentador (pode ser um analista ou programador): registra todas as conclusões do grupo.
- ◆ Observador (deverá ser uma pessoa perspicaz).
- ◆ Indicador de assunto: garante que todos os pontos sejam tratados e deve ser um envolvido no projeto, que conheça bem o assunto.
- ◆ Representante dos envolvidos (usuários): representa as áreas envolvidas (devem ser escolhidas as pessoas-chave, independentemente do nível hierárquico).
- ◆ Gerente do projeto: deve acompanhar as sessões, uma vez que o sucesso ou o fracasso do projeto será sua responsabilidade.
- ◆ Especialista: conhece os projetos da organização, que tenham interface com o projeto em estudo.



Brainstorming

Chamado também de “tempestade de ideias”, pode-se definir o *brainstorming* como sendo uma técnica que auxilia a potencializar a criatividade e a encontrar soluções para determinados problemas. Pode ser feita uma reunião, presencial ou *on-line*, na qual cada participante dela pode expor suas ideias e sugestões, com total liberdade, e também debater sobre as ideias e sugestões dos demais participantes.

Um *brainstorming* deve ter como métodos:

- ◆ Divulgação clara dos objetivos
- ◆ Geração de ideias (entre 20 e 60 minutos)
- ◆ Intervalo para relaxamento
- ◆ Estudo detalhado de cada ideia



Entrevistas

Entrevistar não é somente fazer perguntas, é uma técnica estruturada que pode ser aprendida e na qual os desenvolvedores podem ganhar proficiência com o treino e a prática.

A entrevista consiste em quatro fases: identificação dos candidatos para a entrevista, preparação para uma entrevista, condução da entrevista e finalização da entrevista.



Pieces

Esta técnica é para os desenvolvedores inexperientes, que apresentam dificuldades em como e o que perguntar para extrair os requisitos do cliente.

A técnica Pieces ajuda a resolver esses problemas, pois fornece um conjunto de categorias de problemas que auxiliam o engenheiro de requisitos a estruturar o processo de extração de requisitos.

Em cada categoria existem várias questões que o desenvolvedor deve explorar com o usuário.

As seis categorias são:

- ◆ **Performance** – Reflete o que o usuário espera.
- ◆ **Informação** – Tipo de acesso às informações (relatório, funções *on-line*) que inclui a quantidade de informação oferecida pelo *software*.
- ◆ **Economia** – Custo de usar um produto de *software*, processadores, armazenagem e conexão.
- ◆ **Controle** – Restrições de acesso ao sistema, acesso a algumas informações e habilidade de monitorar o comportamento do sistema.
- ◆ **Eficiência** – Evitar coletar o mesmo dado mais de uma vez e armazená-lo em espaço múltiplo.
- ◆ **Serviços** – Refere-se a que tipo de serviços os usuários necessitam que o *software* realize.