

JAN/2018

CURSO DE FÉRIAS

ANÁLISE DE NEGÓCIOS



Sumário

Introdução	3
Técnicas de levantamento de requisitos	8
BPMN	22
Gerenciamento de Projetos	28
Metodologia ágil e SCRUM	40
Analista de Negócios vs. Garçom de Requisitos	49
Modelagem de Dados	51
Introdução à Banco de Dados	66
Prototipação	74
Estórias de Usuário	84
Introdução a GIT	92
Homologação e Testes	103
Referências	108

Sobre a empresa

Olá, seja bem-vindo (a) a SMN! Nós estamos muito felizes por ter você aqui com a gente, e para você não ficar “tããã” perdido, vamos te mostrar um pouquinho do que é a empresa. Se você tiver qualquer dúvida a qualquer momento, pode chamar a pessoa que está te treinando para esclarecer, pode ficar tranquilo, pois ele vai estar enrolando mesmo e você não vai atrapalhar em nada!

História da empresa

Em 2007, aquele gordinho, careca e de cabelo branco, o Ricardo, fundou a SulMinasNet, lá em Carmo do Rio Claro com intuito de ser uma provedora de internet, além de iniciar o atendimento à Momentum Empreendimentos Imobiliários com o desenvolvimento de um ERP.

Já aquele santista, o Mandara, tinha a Engenharia Solutions, que inclusive era de Santos e atendia o Magazine Luiza desde 2008 com o desenvolvimento de soluções.

Em 2012, eles resolveram se juntar através de um “Joint Venture”, unindo a SulMinasNet e a Engenharia Solutions, surgindo assim a SMN Tecnologia da Informação, em Franca-SP!

A evolução

Inicialmente, trabalhávamos na sala 72 do edifício Esmeralda, que fica lá na praça Barão ao lado do Senhor Café, sabe? Onde tínhamos 6 pessoas e o foco era atender Momentum e Magazine Luiza. Trabalhando com ASP clássico, com SQL Server na Momentum e ORACLE no Magazine Luiza.

Com o crescimento dos projetos e da equipe, precisamos nos mudar para a sala 54 do edifício Medical, lá na rua do Comércio. O ano era 2013, já tínhamos muitas pessoas na equipe que inclusive davam seus passos iniciais no ASP .NET, MVC e ASP .NET Web Api. Ano também que começamos a brincar

com o framework SCRUM. Foi aí então que o nosso programa de estágio começou a ganhar força...

E lá vamos nós novamente, crescendo a equipe e a demanda! Uma sala do prédio não era o suficiente para nós, ainda em 2013 também utilizamos a sala 24 do próprio Medical, que posteriormente virou uma sala de Coworking.

Já em julho de 2014 nos mudamos para a famosa “casona”, que fica lá na rua Couto Magalhães, e lá, foi uma mudança de paradigma total...

Você não deve saber, mas na época em que trabalhávamos nos prédios, a gente ia de social, era uma gracinha! Tinha gente que ia até de gravata, acredita?

Mas lá na casona foi diferente, foi aí que começamos a trabalhar como a gente bem entendia! Quer vir de bermuda e chinelo? Então bora!

Lá também tinha uma churrasqueira, piscina, videogame e tudo mais! E foi aqui que começamos a diversificar a nossa gama de produtos, com a chegada do GCPro :)

As festas de confraternização começaram a ficar grandes, o bom e velho JK no restaurante barão não era mais opção.

E como crescemos hein! Já é a nossa 4ª sede desde então, nos mudamos aqui para o barracão em outubro de 2015, onde começamos com 6 pessoas, e agora já estamos com 125. E olha que alegria, vamos aumentar esse número para 125 + 1 com a sua chegada :D

Hoje, já temos diversos produtos e clientes, que vou te mostrar daqui a pouco, mas com a mudança para cá, várias coisas mudaram também.

Antigamente, nós trabalhávamos como PJ e cada um tinha a sua “MEI”, mas agora, todos nós somos registrados pela CLT, mas é só para burocratizar viu? Para nós esse relógio de ponto serve só para isso. Você já deve ter ouvido de alguém que a regra é: “Entregue o projeto no prazo e sem bugs... O que você faz com o seu dia não me importa. ”

E realmente é isso mesmo, se você quer ficar jogando FIFA o dia todo, no pebolim ou no ping-pong, não tem problema! Problema mesmo é deixar o nosso cliente insatisfeito com soluções mal feitas e fora do prazo :(isso é terrível!

Por isso, curta muito tudo isso que está disponível para você, mas lembre-se do seu papel aqui dentro... você é muito importante para desperdiçarmos esse potencial todo!

Se você é estagiário, sua função é estudar para se efetivar o mais rápido possível! Precisamos de você produzindo para empresa crescer, e, por conseguinte, todos crescerem também! :D

A divisão de tarefas

Hoje, nós temos 3 áreas de atuação aqui dentro da empresa...

A SMN, é a empresa que faz a parte de construção de softwares, sistemas, aplicativos e afins.

A EngDigital, é um provedor de internet empresarial, que também está aqui dentro.

E o EdCenter, é um centro de treinamento, que dá vários cursos, que vão desde informática básica até gestão de projetos, programação e outras coisas bem legais!

Nosso time

Hoje a nossa família é composta por aproximadamente 125 pessoas, sendo, 24 analistas de negócio, 40 desenvolvedores back-end, 17 desenvolvedores front-end, 9 analistas de BI, 6 desenvolvedores mobile, 2 analistas de infraestrutura, 3 auxiliares administrativos, 2 auxiliares de limpeza, 3 coordenadores e 2 diretores.

Premiações

E não podemos esquecer também de todas as premiações que nós ganhamos...

Certificação Microsoft Gold Partner em desenvolvimento, premiação da Kriar sobre gestão humanizada e ganhamos o prêmio Great Place to Work, ficando nada mais nada menos entre as 15 melhores empresas para se trabalhar no setor da tecnologia da informação no Brasil, sendo assim um baita lugar a ser trabalhar, não é?

E vamos falar um pouco mais sobre:

Kudo Cards

Kudo Cards são aqueles cartõezinhos que ficam lá na frente perto do relógio de ponto. São cartões de reconhecimento! Se alguém lhe tirou uma dúvida muito foda, ou você chegou num dia triste todo chateado e alguém te contou uma piada e você ficou feliz, escreva um Kudo Card. É um sinal de que o trabalho do seu colega está sendo reconhecido.

Toró de ideias

O “Toró de ideias” é uma maneira de melhorarmos a empresa sugerindo alguma coisa legal que acrescente algo para todos. Você pode sugerir o que quiser, colocar o preço, e no fim de cada mês, um daqueles itens pode ser escolhido para ser comprado.

Grupos de estudo

Nós incentivamos muito o crescimento intelectual do pessoal, por isso, sempre está rolando um grupo de estudo sobre alguma coisa, nos finais de semana ou depois do expediente. Fique atento aos e-mails e aos chats do Skype!

Aulas de inglês

Também temos aulas de inglês aqui na empresa, você pode não manjar nada ou ser o fodão do inglês, com certeza vai ter uma turma de aula ou conversação para você acompanhar. E o preço? É muuito barato! Na verdade, é um valor simbólico, só para você tirar algo do bolso... talvez seja um incentivo a mais para você aprender e não dormir nas aulas rrsrrs

Lojinha

A lojinha é um benefício super legal! Lá tem doces, salgadinhos, balas, chocolates, refrigerantes e tudo mais. Ninguém controla o que você pega lá, só você. Então, sempre que pegar algo, marque em algum lugar para pagar toda segunda-feira ou coloque o dinheiro na caixinha que tem lá.

Então seja bem-vindo-vindo (a) e espero que curta bastante as aulas :D

Técnicas de levantamento de requisito

Os limites do sistema e do contexto

Os requisitos de um sistema a ser desenvolvido não existem simplesmente, eles precisam ser elicitados de alguma fonte. O propósito de saber o limite do sistema e do contexto é identificar a parte do ambiente que influencia ou não os requisitos.

OBS: Fontes de requisitos podem ser: Stakeholders, Documentos, Sistemas legados, etc.



Fonte: Fundamentos da Engenharia de Requisitos, Klaus Pohl - Chris Rupp

Sistema: Onde se localiza o escopo, funcionalidades e componentes do sistema.

Limite do Sistema: Onde separa o que de fato faz parte do sistema e o que faz parte de seu contexto, isto é, separa a parte da realidade que pode ser alterada pelo processo de desenvolvimento daqueles aspectos do contexto.

Contexto do sistema: Onde tudo que precisa ser levado em consideração em relação ao sistema se localiza, aspectos que não podem ser modificados pelo processo de desenvolvimento. Alguns aspectos são: processos de

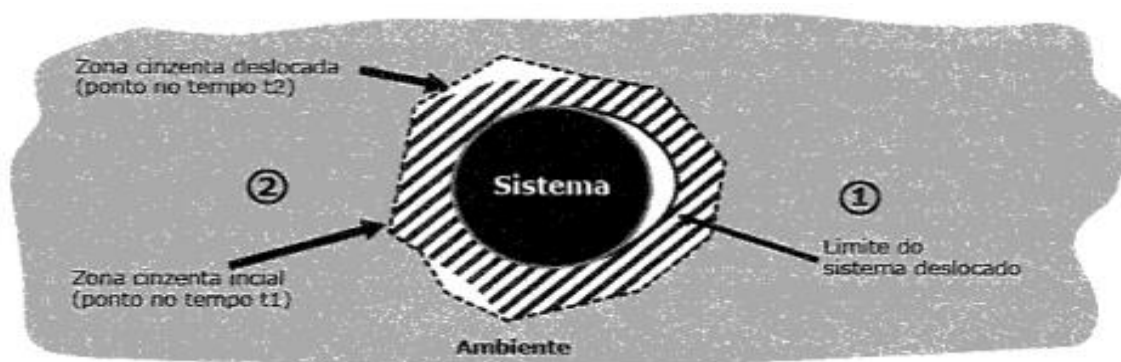
negócios, processos técnicos, normas, leis, pessoas, stakeholders, requisitos, etc. É a parte da realidade que influencia o sistema e seus requisitos.

Limite do Contexto - Onde separa tudo que é relevante para o contexto do sistema do seu ambiente irrelevante.

Ambiente Irrelevante - Onde se localiza tudo que não influencia o sistema. Não precisa ser considerado.

Zona Cinzenta - Onde se localiza as incertezas, faz ou não faz parte do sistema e seu contexto?

OBS: Existem 2 zonas cinzentas, uma entre o sistema e seu contexto e outra entre o contexto e o ambiente irrelevante. Para definir o escopo do sistema e seus limites, é necessário eliminar essa zona cinzenta (incertezas) entre o sistema e seu contexto. Já a zona cinzenta entre o contexto e o ambiente irrelevante, dificilmente é compreendida devido a quantidade de informações que fazem ou não parte do contexto do sistema.



Fonte: Fundamentos da Engenharia de Requisitos, Klaus Pohl - Chris Rupp



Fonte: Fundamentos da Engenharia de Requisitos, Klaus Pohl - Chris Rupp

Interfaces: Fontes e destinos interagem com o sistema por meio de interfaces. Através dessa interface o sistema fornece sua funcionalidade para seu contexto, monitora seu contexto e influencia parâmetros do contexto.

O principal objetivo do levantamento de requisitos é conhecer a real necessidade que o sistema precisa para ser desenvolvido.

O que é requisito?

Condição para alcançar/atingir um determinado objetivo.

Objetivos, propriedades e restrições que o sistema deve possuir para atender de forma correta e cumprir ou satisfazer contratos, padrões e especificações.

Exemplo de requisitos comuns:

- Para ser contratado, deve ter o ensino médio completo.
- Para ter direito a receber esse benefício (PIS), o trabalhador precisa ter cadastro no PIS há pelo menos 5 anos.
- O candidato à obtenção da CNH deve saber ler e escrever.

Na Engenharia de Software, existem três tipos de classificação de requisitos, são eles: Requisitos Funcionais (RF), Requisitos Não-Funcionais (RNF) ou Requisitos de Qualidade e Requisitos de Restrição.

Requisitos Funcionais definem a funcionalidade oferecida pelo sistema. São geralmente subdivididos em requisitos funcionais, requisitos comportamentais e requisitos estruturais. Os requisitos funcionais podem ser por exemplo: cálculos, manipulação de dados, detalhes técnicos.

Exemplo de requisitos funcionais:

- O sistema deverá manter um controle de todos os pacientes que forem atendidos pela agência transfusional, onde será possível inserir, alterar ou editar um paciente.
- O sistema deve gerenciar as requisições que serão geradas.
- O sistema deverá manter um controle de hemocomponentes, onde será possível inserir, alterar ou editar um hemocomponente. Ao final da inserção, o sistema deverá gerar um código de barras.
- O sistema deverá criar requisições, indicando o número do atendimento do paciente, quais hemocomponentes serão usados e qual a quantidade.

Os Requisitos Não-Funcionais ou Requisitos de Qualidade, diz respeito às características e padrões de qualidade que o sistema deve oferecer, como por exemplo, desempenho, usabilidade, portabilidade, robustez, segurança, entre outros. Essas características estão ligadas diretamente às funcionalidades do sistema, definindo sua eficiência. Lembrando que os RNF são características de um software de qualidade, sendo de responsabilidade da empresa de desenvolvimento atendê-los.

Exemplo de requisitos não-funcionais:

- O software deverá ser uma aplicação Web.
- O tempo de resposta das funcionalidades do sistema não poderá ultrapassar 10 segundos.
- A usabilidade do sistema deve ser de fácil aprendizado.
- Somente usuários autorizados deverão ter acesso a essas informações.

Requisitos de Restrição, diz respeito à limitação do espaço da solução, ele não é implementável, limita o sistema e/ou pessoas que estão desenvolvendo.

Exemplo de requisitos de restrição:

- O sistema deverá ser implementado utilizando serviços da WEB.
- O sistema deverá estar disponível no mercado no mais tardar no segundo trimestre de 2017.

Como comunicar esses requisitos?

Os requisitos são comunicados de duas formas, linguagem natural e/ou modelo conceitual.

Ambas têm suas vantagens e desvantagens, por exemplo, a linguagem natural é sem dúvidas a mais utilizadas para a coleta de requisitos, mas por se tratar de uma "proza livre", pode haver ambiguidade na comunicação fazendo com que o requisito tenha dupla compreensão. Mas pelo outro lado é uma forma que não exige treinamento.

Já o modelo conceitual exige um certo conhecimento em modelagem de diagramas, porém sua compreensão e entendimento é mais certo.

Para dar início ao processo de levantamento de requisitos é necessário identificar as partes interessadas e as fontes de requisitos, pois muitas decisões durante sua execução dependem dos chamados stakeholders. O que é e quem são as partes interessadas? Mas o que é stakeholder? Quem são eles? Qual a importância de conhecê-los? Como identificar esses tais stakeholders? E se eu esquecer algum stakeholder?

O que é e quem são as partes interessadas?

As partes interessadas no projeto, que são pessoas e organizações ativamente envolvidas no projeto ou cujos interesses podem ser afetados como resultado da execução ou do término do projeto.

O que é stakeholder?

Stakeholder significa público estratégico e descreve uma pessoa ou grupo que tem interesse em uma empresa, negócio ou indústria.

Quem são eles (stakeholders)?

São todas as pessoas, empresas e instituições que impactam ou são impactadas pelo projeto, de forma direta ou indireta. Na prática cada stakeholder tem seu grau de interação e influência sobre o projeto, sendo necessário classificar essas partes interessadas de modo a dar a devida atenção a cada uma.

Qual a importância de conhecê-los?

Saber quem são os stakeholders do projeto permite que você identifique interesses, alinhe expectativas e descubra o quanto antes a melhor forma de lidar com eles, transformando-os em aliados. Quando você identifica com exatidão quem são as partes interessadas em seu projeto, pode desenvolver uma comunicação clara e objetiva com cada uma delas, estabelecendo assim uma relação de confiança que levará ao **engajamento** desses públicos e à integração de esforços para o sucesso do projeto.

Mas como identificar esses tais stakeholders?

Liste todas as possíveis partes interessadas: A princípio, faça uma sessão de BrainStorming para listar todas as potenciais partes interessadas no projeto. Aqui não importa (ainda) o grau de interesse ou de impacto que cada stakeholder tem, de forma que o que você precisa é ter uma visão de quem são os públicos afetados pela iniciativa.

Além desse BrainStorming inicial, faça pesquisas de mercado, realize um benchmarking com a concorrência, converse com o cliente e envolva o maior número de pessoas possível para compreender a extensão total dos públicos envolvidos.

Entenda os interesses de cada stakeholder: Com a lista de stakeholders formada, procure entender quais são os interesses de cada um a respeito do projeto. Nesse momento é importante pensar tanto positiva quanto negativamente, viu?

Assim, se algum stakeholder tiver interesses que possam prejudicar a iniciativa, relacione! É o caso de um político influente que pode não querer que sua empresa adquira uma área vizinha à sua residência para transformá-la em uma planta fabril, por exemplo, ou de um concorrente que escolhe minar sua iniciativa de expandir os negócios fazendo lobby para que você não adquira um investimento externo. Absolutamente todas as possibilidades devem ser exploradas para que você consiga conhecer profundamente quem são seus stakeholders e como tratá-los, a fim de atingir seus objetivos.

Classifique seus stakeholders por ordem de importância: classifique seus stakeholders em ordem de importância e desenvolva um plano de ação para cada um deles: como você vai conversar com esses stakeholders, como vai obter seu apoio, que tipo de riscos cada um pode trazer ao projeto e como mitigá-los? E se por acaso o inesperado acontecer, como sairá de uma situação de conflito com os stakeholders mais influentes?

Veja que aqui você precisará, mais do que nunca, de duas competências essenciais ao gerente de projetos: comunicação e negociação. O segredo é se manter sempre à frente da sua equipe e evitar embates desnecessários, mantendo um diálogo aberto com todas as partes interessadas.

Identificar os stakeholders depende da complexidade de cada projeto. Assim, quanto mais complexo ele é, mais stakeholders estarão envolvidos e maior deve ser o cuidado ao classificá-los, determinando que tipo de atuação deve ser desenvolvida junto a eles.

E se eu esquecer algum stakeholder?

Bom, não levar em consideração todos os stakeholders necessários acarretará vários problemas futuros, por exemplo, o sistema pode ficar incompleto devido as informações que o stakeholder esquecido tinha para

oferecer, podem haver uma lacuna importante no processo que interfere no resultado final, etc.

Técnicas para levantar requisitos:

Entrevistas: A entrevista é uma das técnicas mais utilizadas e tradicional, produz ótimos resultados na obtenção de dados. O analista faz perguntas sobre o negócio, processos e funcionamento, com objetivo de compreender e capturar requisitos para atender necessidades do cliente.

Vantagens: descobrir informações que o cliente está mais interessado; incluir ou retirar perguntas durante o processo; motivar o entrevistado no decorrer do processo com novas ideias, por exemplo.

Desvantagem: Podem ocorrer desvios de curso no decorrer da entrevista; exige um tempo considerável de atenção do entrevistado; o entrevistado pode não saber se expressar corretamente suas necessidades.

Workshop: O Workshop é realizado em grupo em meio de uma reunião estruturada. Devem compor esta reunião uma equipe de analista e uma seleção de stakeholders que melhor representam a organização e o contexto em que o sistema será utilizado.

Vantagens: Respostas relativamente rápidas; conjunto de requisitos bem definidos; tempo de obtenção de informações é reduzido.

Desvantagens: Nem sempre todos os stakeholders estarão disponíveis; não abre espaço para ideias externas.

BrainStorming: Chuva de ideias, normalmente é utilizado durante um Workshop. É uma atividade desenvolvida para explorar a potencialidade e a criatividade de um indivíduo ou de um grupo.

Vantagens: Várias pessoas pensam melhor do que uma; uma ideia pode ser inspiração para novas ideias; muitas ideias são recolhidas em um curto período.

Desvantagens: Nem sempre todos os stakeholders estarão disponíveis.

Paradoxo de BrainStorming: É o inverso do BrainStorming tradicional. Coleta caminhos que não deve acontecer.

Questionário: Diferente da entrevista, o questionário é interessante quando há um grande número de pessoas para extrair as mesmas informações. É feito um questionário com perguntas pré-definidas pelo analista e entregue aos participantes para que possam responde-las.

Vantagens: Atinge um grande número de pessoas; baixo custo; opinião consolidada.

Desvantagens: Não há garantia de que todos os participantes respondam o questionário; pode haver ambiguidade no entendimento da pergunta para diferentes pessoas.

Observação: A técnica consiste em visitar o local em foco com objetivo de observar como funciona processos, operações e cotidianos das atividades diárias do local.

Vantagens: Capaz de captar o comportamento natural das pessoas; a real necessidade em que sistema irá trabalhar.

Desvantagens: Número restrito de variáveis; requer treinamento especializado; demanda tempo do observador.

Mudança de Perspectiva: Levar em consideração as opiniões e visões externas, pontos de vistas diferentes, conhecimentos diferentes.

Reuso de requisitos: Reutilização de especificações e glossários referente a projetos de sistemas legados já consolidados e que atendem todos os critérios de qualidade.

Vantagens: Economia de tempo e dinheiro; redução de riscos devido o conhecimento prévio pelos stakeholders de determinado requisito.

Estudo de Documentação / Análise de conteúdo: Estudo e reutilização de documentações de diferentes naturezas, para identificação de requisitos a serem implementados no sistema que se está sendo desenvolvido. Uma grande variedade de documentação pode ser analisada incluindo estrutura organizacional da empresa, leis, normas, regras, manuais de usuários, relatórios, resultados, termos de negócios, etc.

Vantagens: A maioria das informações são verdadeiras e já atendem um critério de qualidade.

Desvantagens: Documentos errados podem levar falhas na definição dos requisitos.

Prototipação: Utilizado no estado inicial do projeto, ajuda os stakeholders a desenvolver uma forte noção sobre o funcionamento de uma aplicação na qual ainda não foi implementada. Através de protótipos os stakeholders podem identificar os reais requisitos e fluxos do trabalho do sistema, apoiando também o trabalho de aprovação ou mudanças.

Vantagens: Permite um feedback mais completo do stakeholder; proporciona um alto nível de satisfação do stakeholder devido a sensação de segurança ao ver algo próximo do real.

Desvantagens: Demanda alto custo de investimento; demanda um tempo maior devido à complexidade do sistema e recursos técnicos.

Questionário de ambientes: Permite aos analistas o real entendimento das necessidades dos stakeholders, acompanhando os mesmos em seu ambiente de trabalho para uma coleta de dados detalhada sobre processos.

Vantagens: Permite um levantamento profundo e detalhado sobre os processos e necessidades dos stakeholders; fácil compreensão de processos complexos.

Desvantagens: Dependendo do processo, necessita uma grande quantidade de tempo e pessoas para a realização da tarefa.

Podem ser utilizados também técnicas de apoio para o auxiliar as técnicas de levantamento de requisitos, como por exemplo: Gravações de Áudio e Vídeo.

Conclusão: Todos os métodos de levantamento de requisitos possuem vantagens e desvantagens a serem consideradas e nenhum deles é completo dadas as inúmeras variáveis de complexidade, perfil de stakeholders, comunicação, nível de conhecimento do negócio, nível de qualificação dos profissionais de levantamento de requisitos, situações políticas, nível de comprometimento dos stakeholders, etc. Com isso, a utilização de mais de uma técnica, de forma combinada irá ajudar na complementação de possíveis lacunas de levantamento, além de melhorar a qualidade e completude dos requisitos visto que pode ocorrer o batimento cruzado de requisitos similares.

Categorização do Modelo de Kano

A gestão de qualidade é basicamente a satisfação das partes interessadas. A satisfação das partes interessadas pode ser classificada em 3 categorias segundo Kano:

Fatores Básicos de Satisfação, Fatores Esperados de Satisfação, Fatores Inesperados de Satisfação.

Fatores Básicos de Satisfação: São fatores de extrema importância para o sistema, conhecimento subconsciente, que as partes interessadas classificam como propriedades evidentes e pressupostas.

OBS: Caso não venham a ser implementados, as partes interessadas ficarão insatisfeitas e desapontadas. Mesmo plenamente atendidos os fatores básicos de satisfação não geram uma atitude positiva por parte do stakeholders, servindo apenas para evitar descontentamento. Portanto, técnicas de observação e de estudo de documentação são úteis para a elicitación neste caso.

Fatores Esperados de Satisfação: São fatores conhecidos conscientemente pelas partes interessadas e explicitamente solicitadas por elas. Requisitos e funcionalidades básicas que o sistema necessita para funcionar.

OBS: Caso não venham a ser implementados, as partes interessadas provavelmente não aceitarão o sistema e sua contribuição para o projeto pode ser desmotivada. Requisitos e funcionalidades esperadas podem ser descobertas por técnicas como, pesquisa, entrevista.

Fatores Inesperados de Satisfação: São fatores cujo o valor só é reconhecido quando as partes interessadas experimentam o sistema por si próprio ou quando o analista de negócios os propõe.

OBS: Caso venham a ser implementados, o cliente pode se surpreender positivamente, se animar mais ainda com o projeto e ficar satisfeito com o produto, uma surpresa agradável, útil e motivadora.

Exercícios

Sistema e Contexto

1 - O que faz parte do sistema?

- ☐ Regras de Negócios
- ☐ Stakeholders
- ☐ Componentes
- ☐ Normas e Leis
- ☐ Requisitos

2 - O que faz parte do contexto do sistema?

- ☐ Normas e Leis
- ☐ Requisitos
- ☐ Stakeholders
- ☐ Regras de Negócios
- ☐ Componentes

3 - Que aspectos devem ser considerados para definir do limite do sistema e de seu contexto?

- ☐ O sistema
- ☐ O contexto do sistema
- ☐ O ambiente relevante
- ☐ As interfaces entre o sistema e o contexto do sistema

4 - Constata-se que a norma de regulamentação para proteção e privacidade de dados não precisará ser mais observada, uma vez que os dados processados pelo sistema são mascarados. O que será influenciado por essa constatação?

- ☐ O limite do sistema
- ☐ O limite do contexto
- ☐ As interfaces do sistema
- ☐ A zona cinzenta do limite do sistema

5 - Quais das seguintes afirmações sobre requisitos de qualidade são verdadeiras e quais são falsas? (V - verdadeiro; F - falsa)

- ☐ Os requisitos de qualidade referem-se ao processo de criar um software e não ao produto
- ☐ Os requisitos de qualidade podem complementar requisitos funcionais
- ☐ Os requisitos de qualidade são elicitados após os requisitos funcionais
- ☐ Os requisitos de qualidade podem ser definidos de forma mais concreta através de requisitos funcionais adicionais

6 - Escreva seu entendimento sobre a técnica de brainstorming.

R:

7 - Escreva seu entendimento sobre a técnica de entrevista.

R:

8 - Cite 2 principais vantagens em utilizar questionários para o levantamento de requisitos.

R:

9 - Quais são os 2 aspectos mais relevantes a considerar na escolha das técnicas indicadas para o levantamento de requisitos para um sistema de software de Gerenciamento de Dados de Produtos?

- () A disponibilidade dos stakeholders
- () A idade dos stakeholders
- () Os prazos de entrega do projeto
- () As ferramentas utilizadas
- () O segmento empresarial que o sistema vai ser empregado

Modelo de Kano

10 - Segundo o Modelo de Kano, os fatores básicos de satisfação são difíceis de identificar. Qual a melhor técnica de elicitação para esses fatores?

- () Entrevista
- () Brainstorming
- () Pesquisa e Observação de Campo
- () Prototipagem
- () Workshop

11 - O que são fatores básicos de satisfação?

R:

Autores do capítulo

Ericson Silva

Andrelino Faria

BPMN

BPMN (Business Process Modeling Notation) é uma notação utilizada para representar processos de negócios, utiliza-se diagramas com o intuito de facilitar a visualização e o entendimento das partes.

Para representar a lógica e o fluxo dos projetos passo a passo são utilizados alguns símbolos que, após o seu entendimento, expressam de maneira clara e objetiva o processo de um software. Não sendo utilizado apenas na TI, as notações BPMN tem uma vasta área de aplicações como por exemplo por gestores e administradores, para que possam compreender melhor o funcionamento de seus projetos.

BPMN é uma notação visual, que se utiliza de um quadro (piscinas) para representar todo o processo, algumas divisões (raias e sub processos) que representam os atores e conjuntos de ações para executar o processo principal. Outros elementos são utilizados para representar o início, as ações e o fim do processo.

O software que iremos utilizar será o Bizagi, um software gratuito e que possui um grande número de ferramentas para a construção de um fluxo BPMN, neste módulo veremos algumas funções de seus elementos e como podemos utiliza-las.

Download Bizagi: <https://www.bizagi.com/pt/produtos/bpm-suite/modeler>

Simbologia

Piscina: Chamamos de piscina o conjunto de ações que são responsáveis por todo o processo que o BPMN irá exemplificar. É aconselhável criar um BPMN para cada funcionalidade ou processo.

Exemplo: Cadastrar Cliente.

PISCINA: Cadastra Cliente	
---------------------------	--

Raia: As raias da piscina, normalmente, são utilizadas para separar os responsáveis de cada ação, pode ser um usuário do sistema ou até mesmo o próprio sistema.

Exemplo: Usuário e Sistema

PISCINA: Cadastra Cliente	RAIA 1 - Usuário	
	RAIA 2 - Sistema	

Divisões de Processo: Como o próprio nome já diz, são divisões do processo principal, são as ações necessárias para que seja possível executar o objetivo descrito e facilitam o entendimento do processo em si.

Exemplo: Para cadastra um funcionário é necessário informar o CPF (usuário) e efetuar a validação (sistema).

PISCINA: Cadastra Cliente	RAIA 1 - Usuário	DIVISÕES DE PROCESSO 1 - Preencher as informações	DIVISÕES DE PROCESSO 2 - Validação das informações
	RAIA 2 - Sistema		

Eventos:



Evento Inicial



Evento de Timer

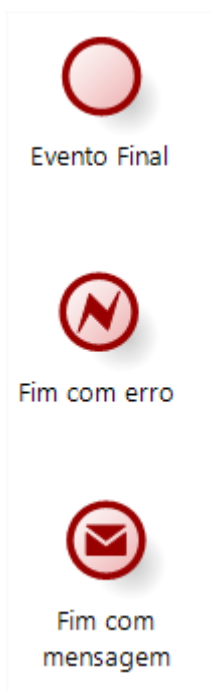


Evento de Mensagem

Eventos de início são identificados por um círculo verde, que podem ser representados de diversas maneiras e indicar como o processo pode ser iniciado. Como uma boa prática é utilizado uma legenda abaixo do símbolo "Início".



Eventos intermediários são círculos amarelos, que representam alguma pausa no processo, aguardando um determinado tempo, alguma mensagem, etc. Não é obrigatório o seu uso e se for utilizado é durante (no meio) o processo.



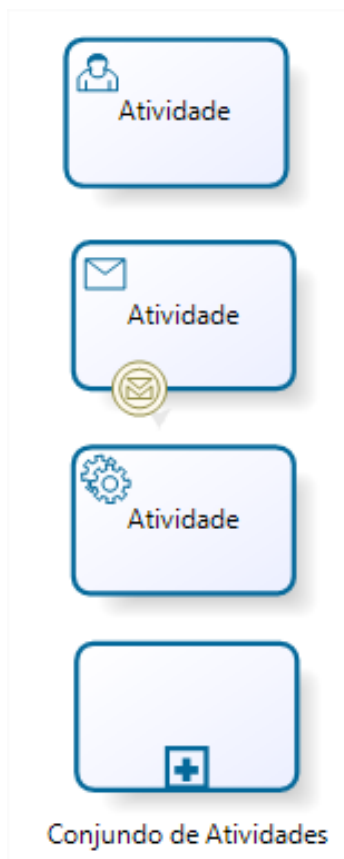
Eventos que representam o final do processo, são identificados como um círculo vermelho e também possuem várias formas de representatividade. Como os eventos intermediários, este tipo de evento pode ser utilizado mais de uma vez durante o processo, mas sempre vão significar o fim de algum caminho percorrido. Também se aconselha o uso de legendas.

Gateways:



Gateways são os elementos que representam as decisões que podem ser tomadas no decorrer do processo, também pode ser representado de diversas maneiras, mas os gateways ou as decisões mais utilizadas são os Gateways Exclusivos (OU), Gateways Paralelos (E) e os Gateways Inclusivos (E/OU).

Atividades:



As atividades são representadas por quadrados com cantos arredondados, representam ações ou os passos que ocorrem dentro do processo. As atividades podem ser simples e representar uma ação do usuário ou do sistema por exemplo, podem ser representadas com um evento intermediário, informando que aquela ação será executada durante um determinado tempo, ou até mesmo ser um conjunto de Atividades, ou seja, dentro de um processo podemos ter vínculos com outros processos, por exemplo dentro do processo "Pedido" é necessário "Cadastrar Cliente".

1 - Você será responsável por desenvolver os processos de uma pizzaria, utilizando a notação BPMN represente os processos abaixo:

- Cadastro de um cliente.
- Cadastro de funcionários.
- Cadastro de produtos.
- Entrega de pedidos.

2 - Será desenvolvido um sistema para uma oficina mecânica, o cliente solicitou os processos detalhados pela notação BPMN. Exemplifique os processos abaixo:

- Cadastrar cliente.
- Cadastrar atendimento.
- Cadastrar venda (exemplificar a forma de pagamento).

3 - Com a notação BPMN, desenvolva o processo de negócio para um sistema de agendamento de férias, onde será necessário ter a aprovação do diretor da empresa e o período está disponível na agenda do RH.

Autores do capítulo

Felipe Araújo

Rafael Borges

Gerenciamento de projetos

Antes de entender o que é o gerenciamento de projetos, é necessário entender alguns conceitos.

O que é:

Projeto - Projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. O projeto é temporário; por ter uma data prevista para iniciar e uma data prevista para terminar.

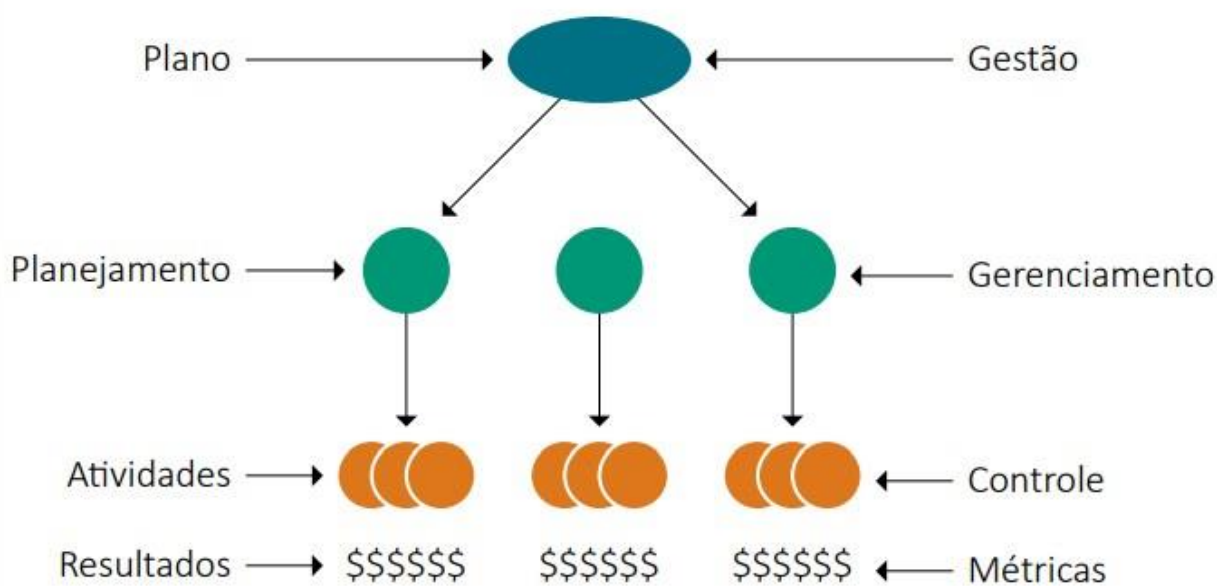
Operações - Combinação de todas as ações técnicas e administrativas destinadas a permitir que um item cumpra uma função requerida, reconhecendo-se a necessidade de adaptação na ocorrência de mudanças nas condições externas.

Plano - É o objetivo de uma empresa em um tempo definido.

Planejamento - São atividades planejadas dentro de um cronograma.

Gestão - Gestão significa Gerenciamento, Administração, onde existe uma instituição, uma empresa, uma entidade social de pessoas, a ser gerenciada ou administrada.

Figura 1 – Visão sobre gestão, plano, gerenciamento e planejamento



Fonte: Torres (2014, p. 49).

A gerência de projetos, pode ser aplicada como disciplina de manter os riscos de fracasso em um nível tão baixo quanto necessário durante o ciclo de vida do projeto, potenciando, ao mesmo tempo, as oportunidades de ocorrência de eventos favoráveis do projeto.(PMBOK 5º edição).

Gerenciamento de projetos é a disciplina de definir e alcançar objetivos ao mesmo tempo que se otimiza o uso de recursos (tempo, dinheiro, pessoas, espaço, etc.).

A importância do Gerenciamento de Projetos

Os gerenciamentos de projetos têm relação direta com a garantia das conquistas de metas das organizações. Tem por objetivo organizar todas as etapas executivas, de modo que o Gerente de Projetos seja capaz de planejar as mesmas, visando analisar a viabilidade do projeto, localizar seus problemas, antecipar imprevistos e definir alocação de recursos financeiros e humanos.

O Gerenciamento de Projetos é também uma poderosa ferramenta de acompanhamento do projeto, pois permite ao Gerente monitorar e controlar o mesmo, revisá-lo e acompanhar o seu desempenho.

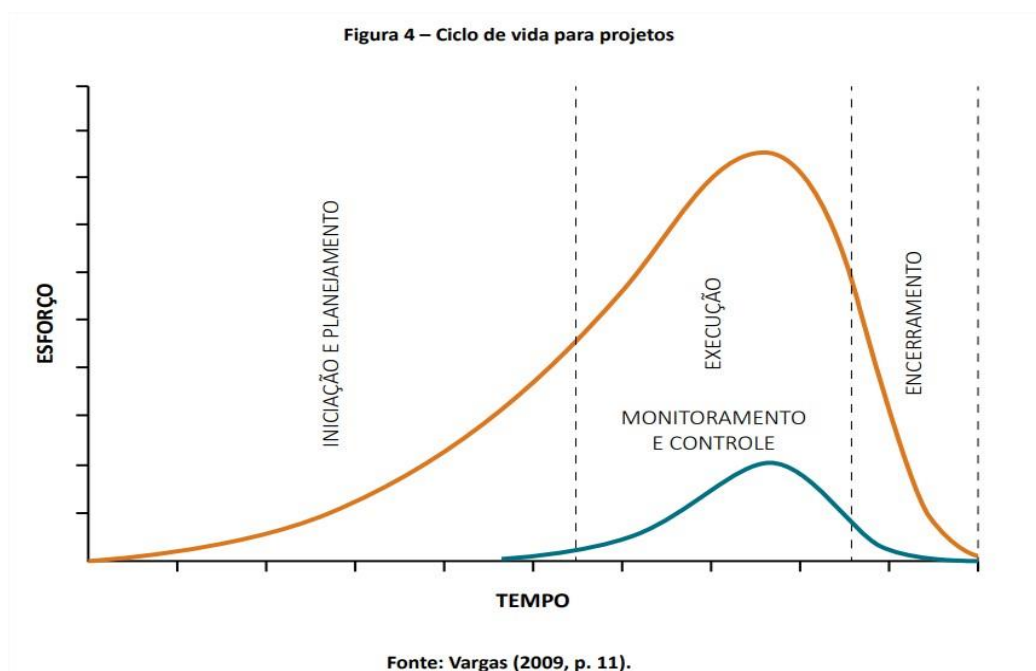
O Gerente de Projetos



Um projeto é desenvolvido pelo profissional chamado de Gerente de Projeto. Sua função é gerenciar o progresso do projeto e através das variáveis (qualidade, custo, prazo e âmbito) verificar seus desvios. Desta forma, seu objetivo geral é cuidar para que as possíveis falhas aos processos sejam minimizadas.

Um gerente de projeto determina e executa as necessidades do cliente. A habilidade de adaptar-se aos diversos procedimentos pode lhe proporcionar um melhor gerenciamento das variáveis e desta forma conquistar uma maior satisfação do cliente.

Ciclo de vida de um projeto



Início do projeto

Planejamento e organização

Execução

Encerramento

Termo de Abertura

O Termo de Abertura do Projeto (TAP) ou Project Charter é o documento que autoriza formalmente um projeto, é ele que concede ao gerente de projetos a autoridade para aplicar os recursos organizacionais nas atividades do projeto.

A elaboração do termo de abertura de um projeto liga o projeto ao trabalho em andamento da organização. Em algumas organizações, o termo de abertura e a iniciação do projeto não são formalmente realizados antes do término de uma avaliação de necessidades, um estudo de viabilidade, um plano preliminar ou alguma outra forma equivalente de análise que tenha sido iniciada separadamente.

O desenvolvimento do termo de abertura do projeto trata principalmente da documentação das necessidades de negócios, da justificativa do projeto, do entendimento atual das necessidades do cliente e do novo produto, serviço ou resultado que deve satisfazer esses requisitos.

O termo de abertura do projeto deve abordar, ou referenciar, as seguintes questões:

Requisitos que satisfazem as necessidades, desejos e expectativas do cliente, do patrocinador e de outras partes interessadas.

Necessidades de negócios, descrição de alto nível do projeto ou requisitos do produto para o qual o projeto é realizado.

Objetivo ou justificativa do projeto (que devem ser SMART)

Gerente de projetos designado e nível de autoridade atribuída

Cronograma de marcos sumarizado.

Stakeholders do projeto e os seus papéis e responsabilidades.

Influência das partes interessadas.

Expectativas das partes interessadas.

Organizações funcionais e sua participação

Premissas organizacionais, ambientais e externas (fatores considerados verdadeiros, reais ou certos).

Restrições organizacionais, ambientais e externas (fatores que limitam as opções da equipe).

Caso de negócios justificando o projeto, incluindo o retorno sobre o investimento.

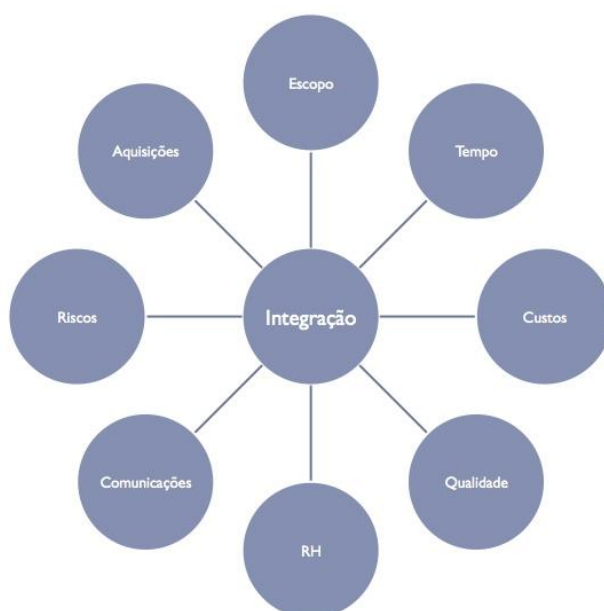
Orçamento sumarizado.

Constrangimentos e riscos.

O que é um Projeto de sucesso?

Um projeto de sucesso é aquele que gerencia as restrições de escopo, prazo e custo, dentro do previsto.

Áreas do Gerenciamento de Projetos



Fonte: Formaggio E.(2014)

A metodologia de gerenciamento de projetos da organização pode melhorar o processo de planejamento do projeto, bem como fornecer algum grau de padronização e consistência. Essa metodologia está estruturada no PMBoK®, no qual são organizadas as áreas de conhecimento em gerenciamento de projetos (Carvalho & Rabechini, 2011), tendo sido agregada em sua última edição, as partes interessadas do projeto (PMI, 2013)

As áreas de conhecimento em gerenciamento de projetos sugerem os processos de gerenciamento de projetos e definem as entradas, as ferramentas e técnicas, bem como as saídas de cada área.

As dez Áreas de Conhecimento em Gerenciamento de Projetos, são (PMI, 2013):

- 1) Gerenciamento da Integração do Projeto
- 2) Gerenciamento do Escopo do Projeto
- 3) Gerenciamento do Tempo do Projeto
- 4) Gerenciamento dos Custos do Projeto
- 5) Gerenciamento da Qualidade do Projeto
- 6) Gerenciamento dos Recursos Humanos do Projeto
- 7) Gerenciamento da Comunicação do Projeto
- 8) Gerenciamento dos Riscos do Projeto
- 9) Gerenciamento das Aquisições do Projeto
- 10) Gerenciamento das Partes Interessadas do Projeto

A restrição do tempo influencia o prazo até o término do projeto. A restrição de custo informa o valor monetário incluído no orçamento disponível para o projeto. Já a restrição do escopo designa o que deve ser feito para produzir o resultado de fim do projeto. Estas três áreas estão frequentemente competindo: o escopo aumentado significa tipicamente o tempo aumentado e o custo aumentado, uma restrição apertada de tempo poderia significar custos aumentados e o escopo reduzido, e um orçamento apertado poderia significar o tempo aumentado e o escopo reduzido.

Descrição e importância das áreas de conhecimento para o gerenciamento de projetos (PMI, 2013).

1) Gerenciamento da Integração do Projeto

O gerenciamento da Integração está ligado diretamente à estrutura do projeto e ao seu desenvolvimento. O seu principal foco está no processo de identificação, suas combinações, suas definições e toda parte da estrutura. Fornece uma visão ampla para o seu desenvolvimento, que envolve:

- Desenvolver o termo de abertura do projeto.

- Desenvolver o plano de gerenciamento do projeto.

- Orientar e gerenciar o trabalho do projeto.

- Monitorar e controlar o trabalho do projeto.

- Realizar o controle integrado de mudança.

- Encerramento do projeto ou fase.

“Os processos de gerenciamento de projetos são geralmente apresentados como distintos e com interfaces definidas, embora, na prática, eles se sobrepõem e interagem de maneira que não podem ser completamente detalhadas no Guia PMBoK® “ (PMI, 2013).

2) Gerenciamento do Escopo do Projeto

Está relacionado ao que falta dentro da estrutura do projeto, o que tem incluso e quais são as suas necessidades. São as exigências especificadas para o resultado fim, ou seja, o que se pretende, e o que não se pretende realizar. A qualidade do produto final pode ser tratada como um componente do escopo. Normalmente a quantidade de tempo empregada em cada tarefa é determinante para a qualidade total do projeto.

Essas variáveis podem ser dadas por clientes externos ou internos. A definição dos valores das variáveis remanescentes fica a cargo do gerente do projeto, idealmente baseada em sólidas técnicas de estimativa. Os resultados finais devem ser acordados em um processo de negociação entre a gerência do projeto e o cliente. Geralmente, os valores em termos de tempo, custo, qualidade e escopo são definidos por contrato.

3) Gerenciamento do Tempo do Projeto

O tempo para concluir o projeto, incluindo todo o processo de planejamento, definição, desenvolvimento e cronograma, é gerenciado por esta área de conhecimento. O tempo requerido para terminar os componentes do projeto é normalmente alterado quando se pretende baixar o tempo para execução de cada tarefa que contribui diretamente à conclusão de cada componente. Ao executar tarefas usando a gerência de projeto, é importante dividir o trabalho em diversas partes menores, de modo que seja fácil a definição das condições de criticidade e de folgas.

4) Gerenciamento dos Custos do Projeto

O gerenciamento dos custos do projeto inclui os processos envolvidos em planejamento, estimativas, orçamentos, financiamentos, gerenciamento e controle dos custos, de modo que o projeto possa ser terminado dentro do orçamento aprovado.

5) Gerenciamento da Qualidade do Projeto

O gerenciamento da qualidade do projeto inclui os processos e as atividades da organização executora que determinam as políticas de qualidade, os objetivos e as responsabilidades, de modo que o projeto satisfaça às necessidades para as quais foi contratado.

6) Gerenciamento dos Recursos Humanos do Projeto

O gerenciamento dos recursos humanos do projeto inclui os processos que organizam, gerenciam e guiam a equipe do projeto. A equipe do projeto consiste das pessoas com papéis e responsabilidades designadas para completar o projeto.

7) Gerenciamento da Comunicação do Projeto

O gerenciamento das comunicações do projeto inclui os processos necessários para assegurar que as informações do projeto sejam planejadas, coletadas, criadas, distribuídas, armazenadas, recuperadas, gerenciadas,

controladas, monitoradas e finalmente dispostas de maneira oportuna e apropriada.

8) Gerenciamento dos Riscos do Projeto

O Gerenciamento dos riscos do projeto inclui os processos de planejamento, identificação, análise, planejamento de respostas e controle de riscos de um projeto.

9) Gerenciamento das Aquisições do Projeto

O gerenciamento das aquisições do projeto inclui os processos necessários para comprar ou adquirir produtos, serviços ou resultados externos à equipe do projeto. A organização pode ser tanto o comprador quanto o vendedor dos produtos, serviços ou resultados de um projeto.

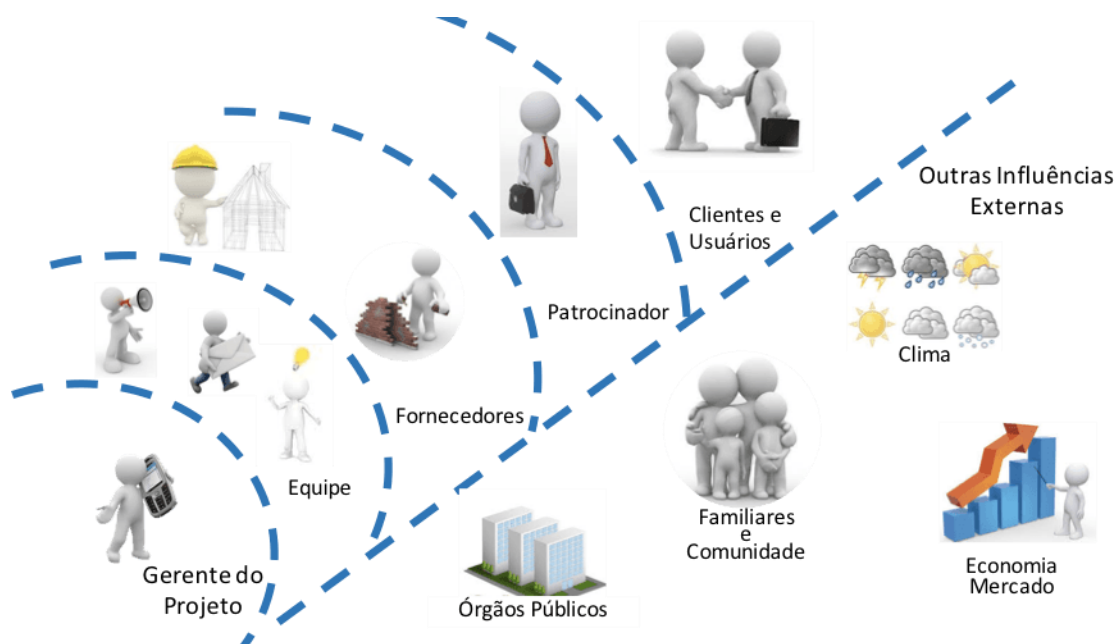
10) Gerenciamento das Partes Interessadas do Projeto

O gerenciamento das partes interessadas também se concentra na comunicação contínua com as partes interessadas para entender suas necessidades e expectativas, abordando as questões conforme elas ocorrem, gerenciando os interesses conflitantes e incentivando o comprometimento das partes interessadas com as decisões e atividades do projeto.

A disciplina de gerenciamento de projetos trata-se de fornecer as ferramentas e as técnicas que permitem à equipe de projeto, e não apenas ao gerente de projeto, organizar seu trabalho para lidar com essas áreas de conhecimento.

Gerenciamento das Comunicações e Partes Interessadas

Stakeholder - PMI (2013, p. 555), “[...] é um indivíduo, grupo ou organização que possa afetar, ser afetado ou sentir-se afetado por uma decisão, atividade ou resultado de um projeto”



Feedback



Na imagem acima, devido a ruídos, o stakeholder tem um entendimento diferente daquilo que foi dito pelo gerente de projetos. O que o gerente de projetos disse na verdade foi: "Olha, Sérgio, NADA bem, pois o recurso NÃO entregou no prazo".

Feedback pode ser compreendido como a forma de reação á alguma coisa, a resposta que se dá a uma questão, uma opinião, que é emitida diante de uma situação ou proposta.

Durante um Feedback, é importante evitar qualquer tipo de ruído para que o receptor entenda de forma clara aquilo que lhe é passado

É sempre importante levar em consideração o feedback recebido, pois através dele, é possível melhorar aspectos que antes não estavam de acordo com o projeto ou postura esperada.

Guias de conhecimento em Gestão de Projetos

Guias de conhecimento (Body of Knowledge -BoKs) em geral são propostas por institutos ou associações de profissionais ligados à área de Gestão de Projetos.

Esses BoKs sintetizam a visão de determinada comunidade sobre as boas práticas de Gestão de Projetos.

O guia mais difundido é o Project Management Body of Knowledge (PM-BoK), proposto pelo Project Management Institute (PMI).

PM-BoK

Está presente em mais de 100 países e é o mais difundido no Brasil.

O Guia PMBOK identifica um subconjunto do conjunto de conhecimentos em gerenciamento de projetos, que é amplamente reconhecido como boa prática, sendo em razão disso, utilizado como base pelo Project Management Institute (PMI).

O Guia PMBOK também fornece e promove um vocabulário comum para se discutir, escrever e aplicar o gerenciamento de projetos possibilitando o intercâmbio eficiente de informações entre os profissionais de gerência de projetos.

O guia é baseado em processos e subprocessos para descrever de forma organizada o trabalho a ser realizado durante o projeto. Essa abordagem se assemelha à empregada por outras normas como a ISO 9000 e o Software Engineering Institute's.

Associado ao guia, existe a certificação profissional de sucesso conhecida como Project Management Professional (PMP).

A certificação PMP atesta sua competência em liderar e dirigir equipes de projetos.

A Certificação Project Management Professional (PMP®) do PMI® é a credencial profissional mais reconhecida e respeitada mundialmente no que tange ao Gerenciamento de Projetos.

No ano de 1999, o PMI® foi pioneiro no reconhecimento de seu programa de certificação pela International Organization for Standardization (ISO) 9001.

Autores do capítulo

Jhonatan Willian Gonçalves

Andreolino Faria

De forma a alcançar os objetivos estratégicos das empresas, é necessário a remoção de restrições em seu caminho. O software é um meio de remover estas restrições auxiliando a empresa a alcançar os seus objetivos estratégicos e chegando ao resultado esperado. Para alcançar tal objetivo, as áreas de desenvolvimento de software utilizam projetos de software e dentro destes projetos estão englobados vários processos de desenvolvimento como a engenharia, documentação e os testes, visando atingir o objetivo final. Mas na prática existem diversos problemas que podem ocorrer como exemplo, o atendimento as expectativas, qualidade, prazo e custo onde nem sempre o que é planejado e combinado é atingido com total sucesso ao final do projeto. Com a utilização da metodologia de desenvolvimento ágil, que será abordada logo em seguida, é possível mudar o este cenário. Nesta metodologia o software é construído de forma incremental e interativa, e para cada parte executa-se um ciclo que informa todas as atividades necessárias para completar as funcionalidades acordadas para este incremento.

Metodologia ágil

Desenvolvimento ágil é o conjunto de métodos e práticas para desenvolver e ajudar outros a desenvolver softwares, sendo utilizado e evoluindo ao longo do tempo desde 1990. Os pilares desta metodologia se baseiam em desenvolvimento iterativo e incremental, equipes multifuncionais, equipes auto organizáveis, maximização do ROI (Retorno de Investimento) e usuários ativamente envolvidos de forma que é direcionado o foco no que as pessoas podem realizar ao invés de focar nos processos e métodos que elas utilizam.

Manifesto ágil

Com o intuito de padronizar a metodologia ágil, em 2001 um grupo de 17 pensadores do desenvolvimento de software se reuniram nas montanhas de Utah nos estados unidos para discutir sobre como aspectos importantes fluíam ao longo do desenvolvimento de um software. Então em forma de documento,

formalizaram os novos processos de software onde foram declarados os valores e os conceitos que os 17 representantes possuíam.

De acordo com o documento de Manifesto Ágil, é valorizado:

"Indivíduos e interação entre eles mais que processos e ferramenta

Software em funcionamento mais que documentação abrangente

Colaboração com o cliente mais que negociação de contratos

Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens a direita, valorizamos mais os itens a esquerda"

Com isso foi criado os 12 princípios ágeis que são:

- 1- Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
- 2- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
- 3- Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
- 4- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
- 5- Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- 6- O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
- 7- Software funcional é a medida primária de progresso.
- 8- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
- 9- Contínua atenção à excelência técnica e bom design, aumenta a agilidade.

- 10-Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- 11-As melhores arquiteturas, requisitos e designs emergem de times auto organizáveis.
- 12-Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo

Frameworks ágeis

- Extreme Programming – XP
- SCRUM

Extreme Programming – XP

É o método ágil usado em projetos constituídos por pequenas ou medias equipes de desenvolvimento de software que trabalham com requisitos indefinidos e em constante mudança e acompanhamento.

Tendo como base a quatro valores, que são:

Comunicação - diariamente face a face

Simplicidade - fazer somente o necessário

Feedback - software entregue o quanto antes e de forma frequente

Coragem - sempre dizer a verdade

Respeito - todos se respeitam

O XP considera o cliente como a fonte mais segura e fundamental para direcionar as informações de que a equipe necessita para a construção do software. Além disso manter o cliente mais próximo é a garantia de que qualquer mudança, duvida, prazo e incertezas, possam ser solucionadas rapidamente pois o mesmo se encontra o tempo todo com a equipe de desenvolvimento.

Scrum

O Scrum é um framework simples para o gerenciar projetos e desenvolvimento de softwares complexos. Com o fornecimento de ferramentas e técnicas de trabalho, o framework adota uma abordagem empírica focando na maximização da habilidade da equipe de resolver os desafios que surgem. Tarefas complexas são divididas em pequenos trabalhos, mantendo a organização e a transparência entre os membros da equipe e as partes interessadas. O Scrum se baseia em 3 pilares fundamentais que são:

Inspeção - atividades processos e práticas devem ser analisados constantemente para que qualquer erro que possa ocorrer seja identificado o mais cedo possível

Adaptação – quando há mudanças significativas ou o resultado esperado não foi satisfatório, então deve se adaptar o que for necessário

Transparência - informações claras e precisas para todos

Papeis do Scrum

Dono do Produto – Product Owner

Ele é o responsável por maximizar o valor do produto e do time de desenvolvimento, determinando o que será incluído no produto, buscando entregar o maior valor para o negócio a fim de garantir um o retorno sobre o investimento (ROI) que está sendo feito. Atuando também como representante do cliente, o dono do produto é o criador da lista dos itens ou requisitos que devem ser desenvolvidos chamado de Backlog do Produto e somente o mesmo pode alterar estas prioridades.

Scrum Master

É o responsável por garantir o uso correto do Scrum agindo como líder auxiliando a equipe na remoção de impedimentos, ajudando o time a definir como o software será desenvolvido, melhorando a qualidade e a produtividade dos processos. Auxilia na remoção dos impedimentos para permitir que o Time de desenvolvimento trabalhe de forma eficaz e eficiente.

Time de Desenvolvimento

É o responsável por transformar os requisitos em um incremento de softwares potencialmente liberável e pronto no final de uma Sprint. Ele é coletivamente responsável pelo sucesso de cada iteração do projeto como um todo demonstrando ser auto organizado e multifuncional. Um time tem normalmente de 3 a 9 participantes com praticamente o mesmo nível hierárquico, ou seja, devem possuir habilidades suficientes para criar, desenvolver, testar o que for realmente necessário para atingir o objetivo do software funcionando.

Eventos do Scrum

De acordo com o Scrum Guide, os eventos foram criados para regularizar e minimizar reuniões sem sentido ou que não foram definidas pelo Scrum. Todos os eventos são definidos por um time-box, ou seja, por um período de tempo fixo, imutável e que possui uma duração máxima. Isto garante que todos os eventos devem possuir um tempo suficiente para ser realizado.

Sprint

É o ciclo de desenvolvimento curto com duração de um mês ou menos onde ocorre todo o trabalho para transformar os itens de requisitos levantados (backlog do produto) em funcionalidades de software. Uma sprint apenas pode

ser cancelada quando o objetivo da mesma não for atingido ou se tornar obsoleto. Isto acontece quando há mudança na tecnologia utilizada ou por condições de mercado. Apenas com a autorização de um dono de produto, uma sprint pode ser cancelada. Uma Sprint é composta por:

- Reunião de planejamento da Sprint

- Reunião diária (Daily)

- Revisão de sprint

- Retrospectiva da sprint

Planejamento da Sprint

É uma reunião onde toda a equipe Scrum participa de maneira colaborativa para definir o que e como será desenvolvido. Ela é uma reunião com time-box de 8 horas de duração para uma sprint de 30 dias. Nesta reunião é dividida em duas partes. Na 1ª parte o time Scrum determina e analisa a meta a ser atingida no sprint baseado no backlog do produto e na 2ª parte é onde o time decide como vai fazer para atingir a meta/objetivo da sprint, criando tarefas a partir dos itens do backlog e fazendo a estimativa para cada tarefa criando assim o backlog da sprint. O backlog da sprint é a soma dos itens serem desenvolvidos nos próximos 30 dias com as suas respectivas tarefas.

Reunião Diária – Daily

É um evento time boxes com duração de 15 minutos e que deve ocorrer todos os dias durante uma sprint, independentemente do comprimento da sprint. Durante a reunião, cada membro do time de desenvolvimento deve responder 3 perguntas básicas:

- O que eu fiz desde a última reunião

- O que vou fazer até a próxima reunião diária

- Há algum impedimento no meu trabalho

Através desta reunião, o time de desenvolvimento avalia o andamento da sprint.

Revisão de Sprint

Ocorre no final de um sprint a fim de inspecionar o incremento produzido. Durante a reunião as partes interessadas (clientes) participam e colaboram sobre o que foi feito durante a sprint e apresentam feedback para novas mudanças a fim de melhorar o resultado das próximas sprints.

Retrospectiva de Sprint

É a reunião de lições aprendidas pelo time Scrum que é realizada após a revisão de sprint onde tem como objetivo responder a 3 perguntas:

O que fizemos de bom e devemos manter?

O que pode ser melhorado?

Como vamos implementar as ações para melhoria?

O objetivo desta reunião é a difusão do conhecimento entre os membros da equipe resultando em membros mais motivados, produtivos e colaborativos.

Artefatos Scrum

Backlog do Produto

Tendo como o dono do produto como responsável pelo seu conteúdo, o backlog do produto é uma lista ordenada de tudo que é necessário no produto. De forma dinâmica, o backlog do produto muda constantemente a fim de identificar o que o produto necessita para ser mais útil e competitivo.

Backlog da Sprint

É a soma dos itens selecionados para a sprint unificado com as tarefas necessárias para desenvolver estes itens. Através deste artefato, é identificado todo o trabalho que o time de desenvolvimento identificou como necessário para atingir a meta da sprint. Somente o time de desenvolvimento pode adicionar ou remover tarefas do backlog da sprint.

Definição de Pronto

É uma lista de atividades que agregam valor ao desenvolvimento do software. Esta definição orienta a equipe Scrum tanto na entrega quanto no planejamento da sprint pois aumenta a transparência no que deve ser produzido juntamente com o aumento da previsibilidade e qualidade. O time de desenvolvimento é o responsável pela definição de pronto.

Exercícios

1 - O que é importante existir antes de um backlog do produto?

- a) Reunião de planejamento da sprint
- b) Backlog da sprint
- c) Definição da visão do produto
- d) Protótipo do produto

2 - O que é Scrum

- a) Scrum é um framework de desenvolvimento de software que só funciona em empresas pequenas
- b) Scrum é uma metodologia de desenvolvimento de softwares, que disponibiliza processos completos e muito precisos para o desenvolvimento de softwares
- c) Scrum é um framework para gerenciamento e desenvolvimento de softwares. Não é um processo detalhado e prescritivo, mas serve como um guia para definir o planejamento, os principais papeis e a forma de trabalho de um time de desenvolvimento de software

d) Scrum é um processo ágil que define apenas práticas de engenharia de software

3 - Os princípios do Scrum são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as atividades estruturais de requisitos, análise, projeto, evolução e entrega. Em cada atividade metodológica, ocorrem tarefas a realizar dentro de um padrão de processo chamado

- a) Process backlog
- b) Scrum Master
- c) Product Owner
- d) Backlog
- e) Sprint

4 - Por que o feedback é uma das coisas mais importantes no desenvolvimento ágil?

5 - Por que a estratégia de desenvolvimento escolhida pelo Product Owner é importante para a equipe e os interessados (futuros usuários, clientes e patrocinadores) no projeto?

Autores do capítulo

Rafael Borges

Felipe Araújo

Analista de negócios x garçom de requisitos

Leva e traz. O que foi solicitado é recebido, na melhor das hipóteses, o que foi pedido. É exatamente essa a função de um garçom, ser o intermédio entre o cliente e a empresa e assegurar que o pedido chegue até seu destino em sua forma integral e exatamente como o cliente pediu. Mas afinal, por que estamos falando da função de um garçom?

Calma, vai fazer sentido. Prometo.

Imagine a seguinte situação, um analista de negócios que trabalha em uma multinacional fica responsável por levantar os requisitos com um cliente já antigo da empresa. Porém, ele tem a adorável fama de ser muito controlador e cabeça dura.

O analista fica sabendo dos tais adjetivos atribuídos a pessoa que ele logo irá se encontrar e a insegurança brota dentro do peito. Mesmo assim, ele pega papel e caneta e segue para reunião.

Chegando lá, a recepção é muito calorosa. Ela é feita por um funcionário que o leva até uma sala de reunião com uma mesa enorme, com várias cadeiras e se senta em uma delas. Não demora muito para o cliente aparecer e a reunião começa.

O levantamento foi um monólogo do cliente e como os requisitos eram para a construção de um site, ele enfatizou que queria no canto da tela, uma animação de uma galinha botando ovos.

Ao longo do desenvolvimento do site, o analista fez o seu papel e no fim, o site de aluguel de carros ficou completamente de acordo com o que o cliente queria, inclusive com a galinha.

Após o site ir ao ar, com a velocidade de propagação de informação da internet, muitas pessoas acharam engraçado e até meio bizarro uma galinha botando ovos dentro de um site na qual o assunto era totalmente voltado para veículos. Por fim, a locadora de veículos perdeu parte de sua credibilidade e o cliente, totalmente insatisfeito, culpou a empresa pelo déficit de aluguéis.

Tudo bem. Garçons e galinhas? Sim, garçons e galinhas.

Percebe que o que esse analista de negócios fez, foi ser apenas um garçom?

Um profissional da área, pela experiência adquirida, costuma saber o que é bom para um sistema e o que não é, e nada mais justo do que externar isso para os clientes. A sinceridade é essencial para um software ser desenvolvido e gerar valor para quem o financiou. Participar, debater e esclarecer ideais faz com o que o relacionamento com o seu cliente fique mais aberto, natural e alinhado.

Um projeto bem-sucedido é um resultado de pessoas que se unem no propósito de fazer dar certo.

Autores do capítulo

Maria Rita Benate

A tecnologia de informação está passando por modificações de forma globalizada, atingindo toda a programação computacional, de dispositivos móveis e outros aparelhos eletrônicos. Em virtude do crescimento dessas informações, os usuários estão necessitando de forma mais intensa, de espaços para armazenar seus dados.

Toda aplicação desenvolvida, torna-se fundamental a presença de um banco de dados (SGBD), do qual tem o objetivo de armazenar os dados feitos via aplicação, possuindo uma interatividade entre base de dados, aplicação e usuário.

A Modelagem de Dados é a criação de um modelo físico que explique a lógica por traz do sistema, com ele você é capaz de explicar as características de funcionamento e comportamento de um software. A modelagem de dados é a base de criação do Banco de dados e parte essencial para a qualidade do sistema.

Modelo

Modelar implica em construir modelos então como fazer isto? Podemos definir as etapas envolvidas na construção de modelos em:

Modelo conceitual - Representa as regras de negócio sem limitações tecnológicas ou de implementação por isto é a etapa mais adequada para o envolvimento do usuário que não precisa ter conhecimentos técnicos. Neste modelo temos:

- Visão Geral do negócio

- Facilitação do entendimento entre usuários e desenvolvedores

- Possui somente as entidades e atributos principais

- Pode conter relacionamentos n para m.

Modelo Lógico - Leva em conta limites impostos por algum tipo de tecnologia de banco de dados. (Banco de dados hierárquico, banco de dados relacional, etc.). Suas características são:

- Deriva do modelo conceitual e via a representação do negócio
- Possui entidades associativas em lugar de relacionamentos n:m
- Define as chaves primárias das entidades
- Normalização até a 3ª forma normal
- Adequação ao padrão de nomenclatura
- Entidades e atributos documentados

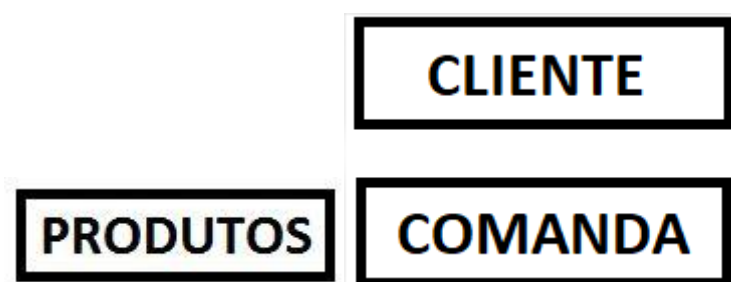
Modelo Físico - Leva em consideração limites impostos pelo SGBD (Sistema Gerenciador de Banco de dados) e pelos requisitos não funcionais dos programas que acessam os dados. Características:

- Elaborado a partir do modelo lógico
- Pode variar segundo o SGBD
- Pode ter tabelas físicas (log, líder, etc.)
- Pode ter colunas físicas (replicação)

Precisamos definir agora entidade, atributo e relacionamento. O que são e o que representam?

Entidade

São representações de algo no mundo físico para um sistema, por exemplo no nosso caso temos as entidades produtos, comanda e cliente. Que são representados pelo símbolo abaixo.



Fonte: William Miranda

Além das **Entidades**, também temos os **relacionamentos** entre as entidades, que nada mais é do que a ligação entre duas entidades, ou algo que faça com que essas entidades tenham algo em comum.

Atributo

Um atributo é tudo o que se pode relacionar como propriedade da entidade. (*Coluna, campo, etc....*). Exemplos de atributos: Código do Produto (Entidade Produto), Nome do Cliente (Entidade Cliente). E temos também algumas definições dos atributos que são:

Atributo obrigatório - é aquele que para uma instância de uma entidade ou relacionamento **deve** possuir um valor. (NOT NULL)

Atributo opcional - É aquele que para uma instância da entidade ou relacionamento **pode** possuir um valor. (NULL)

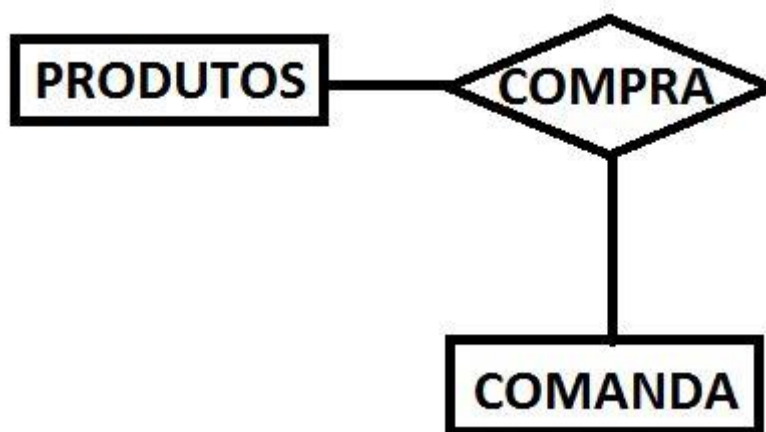
Podemos ainda classificar os atributos como:

Atributo Identificador ou Chave Primária - (#) - Atributo capaz de identificar exclusivamente cada ocorrência de uma entidade. Também conhecido como chave Primária ou Primary Key (PK). Ex: Código do Cliente, Código do Produto, etc. (O símbolo # é usado para representar a chave primária em algumas notações)

Chave Candidata - Atributo ou grupamento de atributos que têm a propriedade de identificar unicamente uma ocorrência da entidade. Pode vir a ser uma chave Primária. *A chave candidata que não é chave primária também se chama chave Alternativa.*

Relacionamento

Uma vez que as entidades são identificadas, deve-se então definir como se dá o relacionamento entre elas. Vou dar um exemplo de relacionamento entre duas entidades.



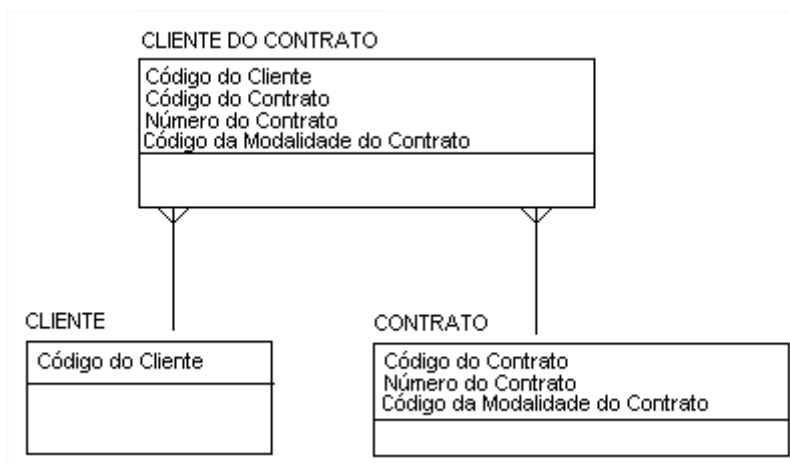
Fonte: William Miranda

De acordo com a quantidade de objetos envolvidos em cada lado do relacionamento, podemos classifica-los de três formas:

Relacionamento muitos-para-muitos (n: n)

Na linguagem mais acadêmica é explicado da seguinte forma: Uma entidade em "A" está associada a qualquer número de entidades em "B" e vice-versa. Alguns autores preferem chamar esta cardinalidade de m: n, por considerar que podem representar valores diferentes.

A cardinalidade **N** para **N** leva para o modelo lógico a necessidade de definição de mais uma entidade. Chamamos isto de **ASSOCIATIVA**. Para o exemplo acima teríamos:



Fonte: José Carlos Macoratti(2013)

A Entidade **CLIENTE DO CONTRATO** é necessária para que possamos identificar o contrato de um determinado cliente. Em toda Cardinalidade **N** para **N** temos a **ASSOCIATIVA**.

Ainda nesta cena temos um outro relacionamento no momento em que o cliente está com uma comanda, está comanda é ativada e até ser feito o pagamento está comanda está atrelada apenas a um cliente e um cliente pode ter apenas uma comanda. Neste momento temos a relação de ...

Relacionamento um-para-um (1:1)

Na linguagem mais acadêmica é explicado da seguinte forma: Uma entidade em "A" está associada com no máximo uma entidade em "B", e uma entidade em "B" está associada com no máximo uma entidade em "A".

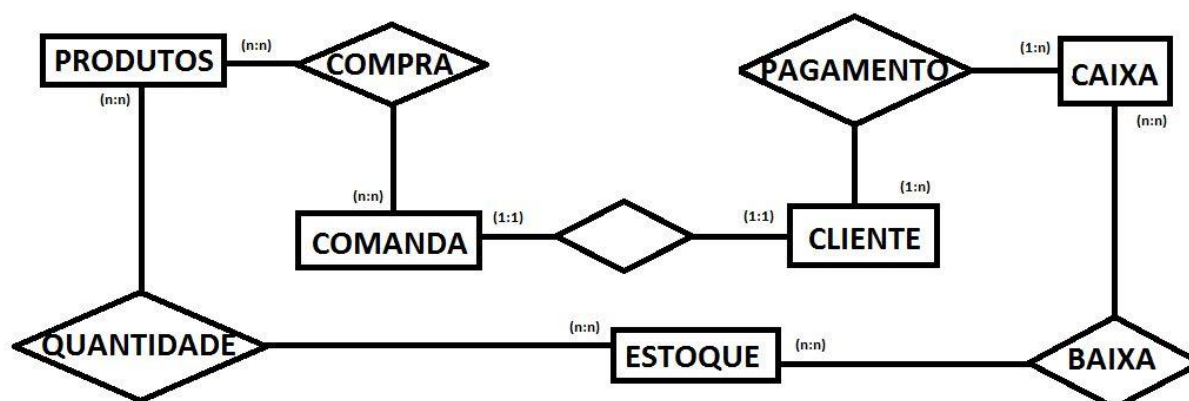
Vamos um pouco mais além do processo, os produtos cadastrados no sistema possuem características diferentes que devem ser levadas em conta, por exemplo os bolos vendidos pelo Sr. João têm validade de 1 dia e precisam ser refrigerados para que a sua consumação possa ser feita sem problemas durante o dia, mas a água vendida também na confeitaria possui uma validade maior e não precisa ser refrigerada para se manter no padrão de consumo. Então vejamos existem tipos diferentes de produtos, mas vejamos este é um relacionamento diferente dos outros, porque um tipo de produto pode estar

atrelado a vários produtos, porém um produto pode ter apenas um tipo. Este relacionamento é o ...

Relacionamento um-para-muitos (1: n)

Na linguagem mais acadêmica é explicado da seguinte forma: Uma entidade em "A" está associada a qualquer número de entidades em "B", e uma entidade em "B", todavia, pode estar associado a no máximo uma entidade em "A".

Após você saber os três tipos de relacionamento, segue o modelo final desenhado.



Fonte: William Miranda

Chave primária

Atributo ou combinação de atributos que possuem a propriedade de identificar de forma única uma linha da tabela. Corresponde a um atributo determinante.

Cada tabela deve incluir um campo ou conjunto de campos que identifique de forma exclusiva, cada registro armazenado na tabela. Essas informações são chamadas de chave primária da tabela.

Desta forma, com a chave primária cria-se uma identificação única, o que dá total segurança para que aplicações possam acessar, alterar e excluir dados sem correr o risco de apagar ou alterar dois campos da tabela ao mesmo tempo.

Chave primária é um importante objeto quando se aplica regras de normalização de dados, muitas das formas normais são baseadas nas relações dos demais atributos com a chave primária da tabela.

Características de uma Chave Primária:

- 1 – Não pode haver duas ocorrências de uma mesma entidade com o mesmo conteúdo na chave primária
- 2 - A chave primária não pode ser composta por atributo opcional, ou seja, atributo que aceite nulo.
- 3 - Os atributos identificadores devem ser o conjunto mínimo que pode identificar cada instância de uma entidade.
- 4 - Não devem ser usadas chaves externas. (Atributos sobre os quais você não tem controle. Ex: CPF)
- 5 - Cada atributo identificador da chave deve possuir um tamanho reduzido
- 6 - Não deve conter informação volátil.

Chave estrangeira

A chave estrangeira ocorre quando um atributo de uma relação for chave primária em outra relação. Em outras palavras sempre que houver o relacionamento 1:N entre duas tabelas, a tabela 1 receberá a chave primária e a tabela N receberá a chave estrangeira.

Vamos ver um exemplo:

Tabela Produtos

- **Codigo_produto**
- Produto
- Categoria
- Preço Data
- Quantidade
- Descricao

Tabela Itens do Pedido

- **Numero_pedido**
- **Codigo_produto**
- Quantidade

Nas tabelas acima temos um caso de **chaves primária e estrangeira**.

Observe que o codigo_produto consta nas duas tabelas. Em Produtos ele é o campo identificador, ou seja, cada produto deverá ser exclusivo, portanto,

uma chave primária. Já em Itens do Pedido o campo `codigo_produto` poderá constar várias vezes e como ele já é chave primária em Produtos, aqui ele será uma chave estrangeira.

Portanto as tabelas ficarão assim:

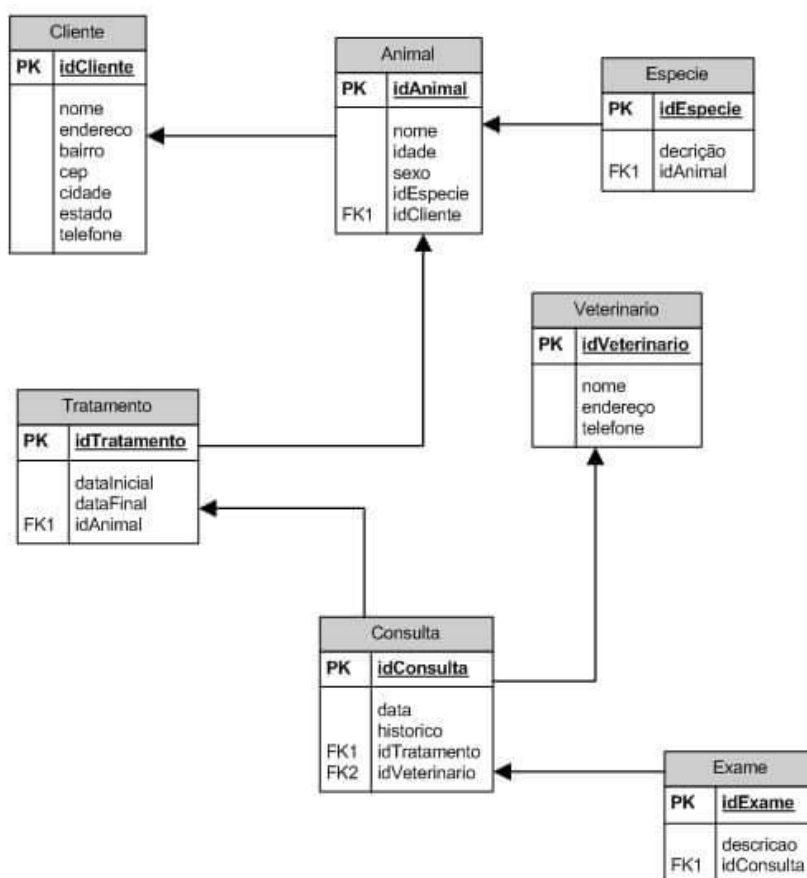
Tabela Produtos

- **Codigo_produto** (chave primária)
- Produto
- Categoria
- Preço Data
- Quantidade
- Descricao

Tabela Itens do Pedido

- **Numero_pedido**
- **Codigo_produto** (chave estrangeira)
- Quantidade

Veja abaixo um diagrama de banco de dados com várias tabelas relacionadas. Perceba a presença de chaves primárias, identificadas por PK (primary key) e as chaves estrangeiras, identificadas por FK (foreign key).



Normalização

Normalização é o conjunto de regras que visa minimizar as anomalias de modificação dos dados e dar maior flexibilidade em sua utilização.

Por que normalizar?

- 1º Minimização de redundâncias e inconsistências;
- 2º Facilidade de manipulações do Banco de Dados;
- 3º Facilidade de manutenção do Sistema de Informações

Para que você compreenda melhor vou dar um exemplo. Vamos supor que você criou uma entidade Funcionários para armazenar as informações dos funcionários de uma empresa e que o resultado físico final seja a tabela mostrada abaixo.

	Codigo	Nome	Cargo	Setor	QuantidadeFuncionarios
	1	Miriam	Gerente	Vendas	23
	2	Jefferson	Programador	Suporte	20
	3	Jessica	Analista	Compras	15
	4	Janice	Programadora	Suporte	16
	5	Mario	Gerente	Design	9

Fonte: José Carlos Macoratti(2013)

Se você olhar bem para a tabela acima vai ter que concordar comigo que ele sofre das seguintes anomalias:

Anomalia de Exclusão - O que acontece se você excluir o funcionário de código igual a 3? O Setor vai ser excluído junto e aí você dançou...

Anomalia de Alteração - O nome do Setor Suporte mudou para Apoio. Você vai ter de alterar o nome em todos os registros da tabela. Dançou novamente...

Anomalia de Inclusão - Foi contratado um novo funcionário para o Setor Suporte. Você vai ter que incluir um funcionário ao campo QuantidadeFuncionarios - em todas as ocorrências com setor de nome SUPORTE. Dançou mais uma vez...

Para poder resolver o dilema acima temos que **NORMALIZAR** a entidade. Para isto aplicamos as **formas normais**:

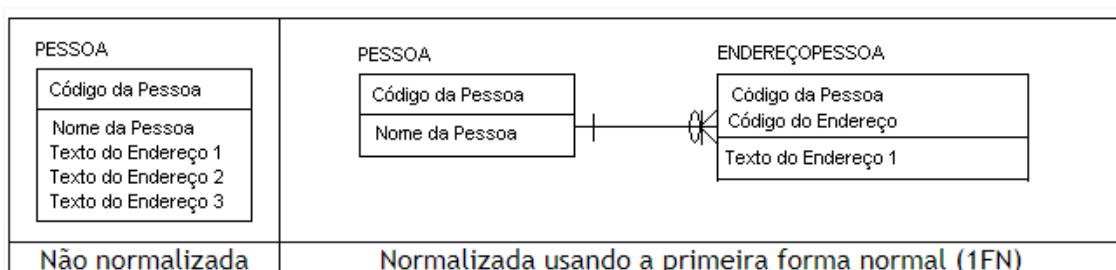
Primeira Forma Normal (1FN) - Uma relação está na 1FN se somente todos os domínios básicos contiverem somente valores atômicos (não contiver grupos repetitivos). Para atingir esta forma normal devemos eliminar os grupos de repetição. Como fazer isso?

Procedimentos:

- identificar a chave primária da entidade;
- identificar o grupo repetitivo e excluí-lo da entidade;
- criar uma nova entidade com a chave primária da entidade anterior e o grupo repetitivo.

A chave primária da nova entidade será obtida pela concatenação da chave primária da entidade inicial e a do grupo repetitivo.

Abaixo temos um exemplo de como efetuar a normalização para a primeira forma normal:



Fonte: José Carlos Macoratti(2013)

Segunda Forma Normal (2FN) - Uma relação R está na 2FN se é somente se ela estiver na primeira e todos os atributos não chave forem totalmente dependentes da chave primária (dependente de toda a chave e não apenas de parte dela).

Procedimentos:

- identificar os atributos que não são funcionalmente dependentes de toda a chave primária.
- Remover da entidade todos esses atributos identificados e criar uma nova entidade com eles.

A chave primária da nova entidade será o atributo do qual os atributos do qual os atributos removidos são funcionalmente dependentes.

Exemplo:

Sejam as entidades:

Arquivo de Notas Fiscais (Num. NF, Série, Código do Cliente, Nome do cliente, Endereço do cliente, Total Geral da Nota)

Arquivo de Vendas (Num. NF, Código da Mercadoria, Descrição da Mercadoria, Quantidade vendida, Preço de venda e Total da venda)

NotaFiscal : Tabela		
	Nome do campo	Tipo de dados
?	NumeroNotaFiscal	Número
	Serie	Texto
	CodigoCliente	Número
	NomeCliente	Texto
	EnderecoCliente	Texto
	Total	Texto

Vendas : Tabela		
	Nome do campo	Tipo de dados
?	NumeroNotaFiscal	Número
?	CodigoMercadoria	Número
	Descricao	Texto
	QuantidadeVendida	Número
	PrecoVenda	Moeda
	Total	Moeda

Fonte: José Carlos Macoratti(2013)

Normalizando para segunda forma normal (2FN):

Arquivo de Notas Fiscais (Num. NF, Série, Código do Cliente, Nome do cliente, Endereço do cliente, Total Geral da Nota)

Arquivo de Vendas (Num. NF, Código da Mercadoria, Quantidade vendida e Total da Venda)

Arquivo de Mercadorias (Código da Mercadoria, Descrição da Mercadoria, Preço de venda)

NotaFiscal : Tabela		
	Nome do campo	Tipo de dados
?	NumeroNotaFiscal	Número
	Serie	Texto
	CodigoCliente	Número
	NomeCliente	Texto
	EnderecoCliente	Texto
	Total	Texto

Vendas : Tabela		
	Nome do campo	Tipo de dados
?	NumeroNotaFiscal	Número
?	CodigoMercadoria	Número
	QuantidadeVendida	Número
	Total	Moeda

Mercadorias : Tabela		
	Nome do campo	Tipo de dados
?	CodigoMercadoria	Número
	Descricao	Texto
	PrecoVenda	Moeda

Fonte: José Carlos Macoratti(2013)

Como resultado desta etapa, houve um desdobramento do arquivo de **Vendas** (o arquivo de Notas Fiscais, não foi alterado, por não possuir chave composta) em duas estruturas a saber:

Primeira estrutura (**Arquivo de Vendas**): Contém os elementos originais, sendo excluídos os dados que são dependentes apenas do campo Código da Mercadoria.

Segundo estrutura (**Arquivo de Mercadorias**): Contém os elementos que são identificados apenas pelo Código da Mercadoria, ou seja, independentemente da Nota Fiscal, a descrição e o preço de venda serão constantes.

Terceira Forma Normal (2FN) - Uma relação R está na 3FN se somente estiver na 2FN e todos os atributos não chave forem dependentes não transitivos da chave primária (cada atributo for funcionalmente dependente apenas dos atributos componentes da chave primária ou se todos os seus atributos não chave forem independentes entre si).

Procedimentos:

- a) identificar todos os atributos que são funcionalmente dependentes de outros atributos não chave;
- b) removê-los e criar uma nova entidade com os mesmos.

A chave primária da nova entidade será o atributo do qual os atributos removidos são funcionalmente dependentes.

Estrutura na segunda forma normal (2FN):

Arquivo de Notas Fiscais (Num. NF, Série, Data emissão, Código do Cliente, Nome do cliente, Endereço do cliente, Total Geral da Nota)

Arquivo de Vendas (Num. NF, Código da Mercadoria, Quantidade vendida e Total da venda desta mercadoria)

Arquivo de Mercadorias (Código da Mercadoria, Descrição da Mercadoria, Preço de venda)

Estrutura na terceira forma normal (3FN):

Arquivo de Notas Fiscais (Num. NF, Série, Data emissão, Código do Cliente e Total Geral da Nota)

Arquivo de Vendas (Num. NF, Código da Mercadoria, Quantidade vendida e Total da venda desta mercadoria)

Arquivo de Mercadorias (Código da Mercadoria, Descrição da Mercadoria, Preço de venda)

Arquivo de Clientes (Código do Cliente, Nome do cliente, Endereço do cliente)

Como resultado desta etapa, houve um desdobramento do arquivo de Notas Fiscais, por ser o único que possuía campos que não eram dependentes da chave principal (Num. NF), uma vez que independente da Nota Fiscal, o Nome, Endereço são inalterados. Este procedimento permite evitar inconsistência nos dados dos arquivos e economizar espaço por eliminar o armazenamento frequente e repetidas vezes destes dados. A cada nota fiscal comprada pelo cliente, haverá o armazenamento destes dados e poderá ocorrer divergência entre eles.

As estruturas alteradas e o motivo das alterações:

Primeira estrutura (**Arquivo de Notas Fiscais**): Contém os elementos originais, sendo excluído os dados que são dependentes apenas do campo Código do Cliente (informações referentes ao cliente).

Segundo estrutura (**Arquivo de Clientes**): Contém os elementos que são identificados apenas pelo Código do Cliente, ou seja, independente da Nota Fiscal, o Nome, Endereço serão constantes.

Após a normalização, as estruturas dos dados estão projetadas para eliminar as inconsistências e redundâncias dos dados, eliminando desta forma qualquer problema de atualização e operacionalização do sistema. A versão final dos dados poderá sofrer alguma alteração, para atender as necessidades específicas do sistema, a critério do analista de desenvolvimento durante o projeto físico do sistema

Exercícios

1 - Considere as seguintes relações para um banco de dados que registra a matrícula do aluno nas disciplinas e os livros adotados para cada disciplina:

ALUNO(Cpf, Nome, Curso, Datanasc)

DISCIPLINA(Num_Disciplina, Dnome, Dept)

MATRICULA(Cpf, Num_Disciplina, Semestre, Nota)

LIVRO_ADOTADO(Num_Disciplina, Semestre, ISBN_Livro)

LIVRO(ISBN_Livro, Titulo_Livro, Editora, Autor)

Especifique as chaves estrangeiras para este esquema, explicando a razão de suas escolhas.

2 - Explique o que é uma superchave(Chave primária) . Apresente um esquema de relação como exemplo. A partir desse exemplo, mostre algumas possíveis superchaves para esse esquema, justificando.

3 - Dada a relação a seguir:

Compra(id_comprador, endereço, id_produto, descrição, valor, data_hora_compra, quantidade, temporada, tipo)

Considere as seguintes dependências funcionais:

DF1: id_comprador, id_produto, data_hora_compra \twoheadrightarrow endereço, descrição, valor, quantidade, temporada, tipo

DF2: id_comprador \twoheadrightarrow endereço

DF3: id_produto \twoheadrightarrow valor, temporada, tipo, descrição

DF4: tipo \twoheadrightarrow temporada

Normalize a relação **Compra** passo a passo, criando um esquema lógico que atenda a 1FN, 2FN e 3FN.

Autores do capítulo

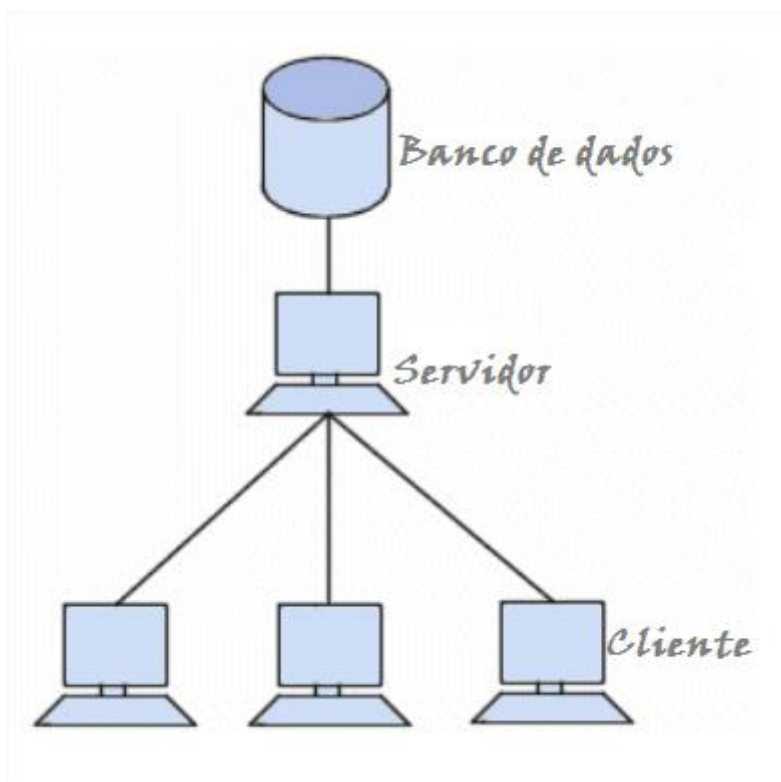
André Luiz Assunção

Ericson Silva

Banco de Dados

O que é um banco de dados?

Um banco de dados (em inglês, database) é um local onde é possível armazenar dados de maneira estruturada e com a menor redundância possível. Estes dados devem poder ser utilizados por programas e usuários diferentes. Assim, a noção básica de dados é acoplada a uma rede, a fim de poder reunir estas informações, daí o nome banco. Geralmente, fala-se de um sistema de informação para designar qualquer estrutura que reúne os meios organizados para poder compartilhar dados:



Abaixo estão apresentadas algumas definições de um banco/base de dados:

Coleção de dados inter-relacionados representando informações sobre um domínio específico;

Coleção de dados integrados que tem por objetivo atender as necessidades dos usuários;

Conjunto de dados persistentes e manipuláveis que obedecem a um padrão de armazenamento;

Conjunto de dados com uma estrutura regular que organizam uma informação;

Exemplos: dicionário, lista telefônica, controle do acervo de uma biblioteca, sistema de controle dos recursos humanos de uma empresa, dados pessoais de uma pessoa.

Dados

Os dados referem-se a uma recolha de informações organizadas, eventos, atividades e transações que são gravados, classificados e armazenados dentro de um sistema de computador.

Exemplo: texto, fotos, figuras, sons gravados, animação, numéricos, alfanuméricos entre outros.

Informação

A informação é o dado organizado, sendo uma abstração informal (não pode ser gerada através de uma teoria lógica ou matemática), que está na mente da pessoa que está inserindo tal informação, possuindo algum significado.

Exemplo: um texto pode ser uma informação uma fonte de muitas informações ou um conjunto de informação, pois se os dados agrupados gerarem sentido para quem o lê e ficando claro ou não a que se refere, o dado passa a ser o valor de um determinado item, evento do que se refere. Outros exemplos de informação são relatórios, boletim escolar, folha de pagamento.

Conhecimento

O conhecimento é resultado de várias informações organizadas de forma lógica e suficiente para criar um evento. Pode ser caracterizado também como uma abstração interior, algo que foi experimentado, vivenciado por alguém.

Exemplo entre dado e informação

Abaixo é exibido uma tabela em que o dado é o nome, endereço e telefone de cada pessoa. Mostrando de forma individual, podendo ser gerado uma informação, quando estão aglomerados e cadastrados em uma lista telefônica.

Tabela 1 Exemplo de dados e informação da tabela lista telefônica que possui alocados dentro dela

Nome	Endereço	Telefone
Maria Joquina	Av. Flores dos Reis, 332	5568-4445
João Aparecido	Rua Jardim Vieira, 122	5211-8854

Fonte: TIAGO VINÍCIUS(2016)

Sistema de Gerenciamento de Banco de Dados (SGBD)

Conhecidos como **SGBD**, em inglês Data Base Management System - DBMS - são um conjunto de programas que permite aos usuários criar e manter um banco de dados, do qual sua principal meta é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que os clientes possam consultar, alterar/atualizar, incluir ou deletar os dados armazenados em um banco de dados.

Tem como característica guardar grandes informações de dados em massa, estruturando em registros e tabelas com funções para acesso e processamento das informações. Abaixo encontram-se alguns exemplos:

PostgreSQL ❤

Oracle;

MySQL;

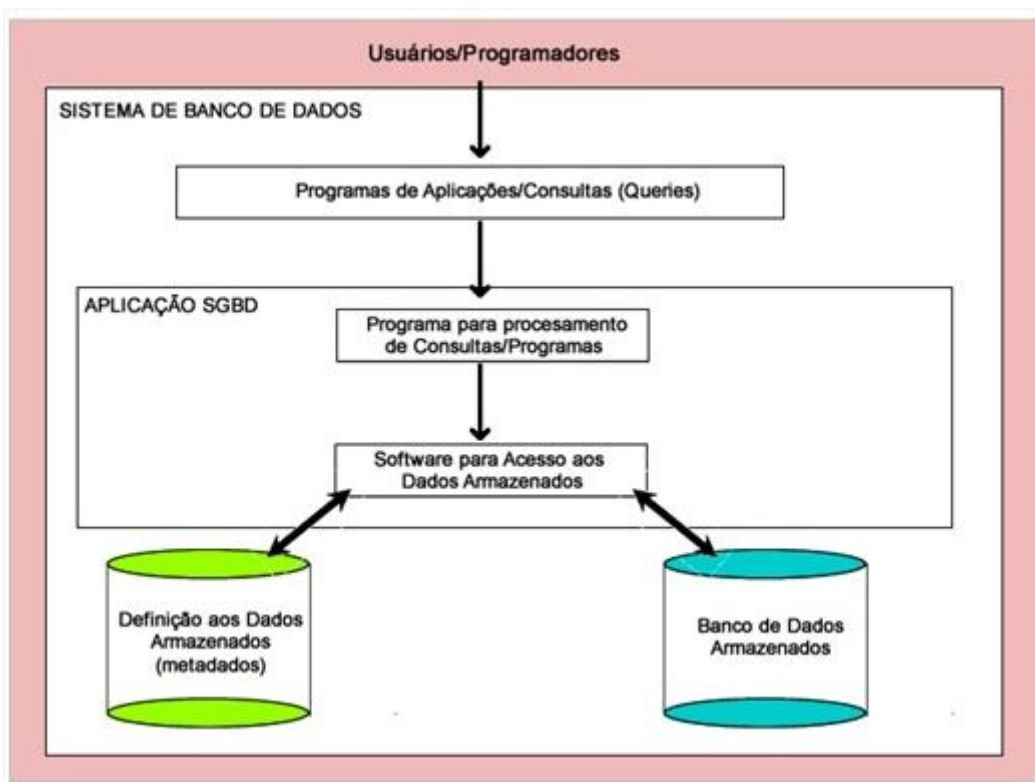
SQL Server;

FireBird;

Access;

DBase;

Outros;



Fonte: TIAGO VINÍCIUS(2016)

Figura 1: Configuração de um sistema de banco de dados

As vantagens de um SGBD

Maior disponibilidade: Uma das principais vantagens de um SGBD é que a mesma informação pode ser disponibilizada a utilizadores diferentes, ou seja, compartilhamento de dados.

Redundância minimizada: Os dados de um SGBD são mais concisos, porque, como regra geral, a informação nela aparece apenas uma vez. Isto reduz a redundância de dados, ou em outras palavras, a necessidade de repetir os mesmos dados uma e outra vez. Minimizando a redundância pode, portanto, reduzir significativamente o custo de armazenamento de informações em discos rígidos e outros dispositivos de armazenamento.

Precisão: dados precisos, consistentes são um sinal de integridade dos dados. SGBDs fomentam a integridade dos dados, porque as atualizações e alterações dos dados só tem que ser feitas em um só lugar. As chances de se cometer um erro são maiores se você é obrigado a alterar os mesmos dados em

vários lugares diferentes do que se você só tem que fazer a mudança em um só lugar.

Programa e arquivo de consistência: Usando um sistema de gerenciamento de banco de dados, formatos de tabelas e programas do sistema são padronizados. Isso faz com que os tabelas de dados sejam mais fáceis de manter, porque as mesmas regras e diretrizes se aplicam a todos os tipos de dados. O nível de consistência entre os tabelas e programas também torna mais fácil de gerenciar dados quando vários programadores estão envolvidos.

User-friendly: Os dados **são** mais fáceis de acessar e manipular com um SGBD do que sem ele. Na maioria dos casos, SGBDs também reduzem a dependência de usuários individuais à especialistas em computação para atender às necessidades de seus dados.

Maior segurança: Como afirmado anteriormente, SGBDs permitem que múltiplos usuários acessem os recursos dos mesmos dados. Esta capacidade é geralmente vista como um benefício, mas há riscos potenciais para a organização. Algumas fontes de informação devem ser protegidas ou garantida e vista apenas por indivíduos selecionados. Através do uso de senhas, sistemas de gerenciamento de banco de dados podem ser usado para restringir o acesso aos dados a apenas aqueles que devem vê-lo.

Outros: Tempo de desenvolvimento de aplicações é reduzido, Maior flexibilidade para realizar alterações (independência de dados) e Maior economia, informações atualizadas, menor volume de papel.

As desvantagens de um SGBD

Existem basicamente duas desvantagens principais em SGBDs. Um deles é o custo, e a outra o perigo para a segurança dos dados.

Custo: A Implementação de um sistema de SGBD pode ser cara e demorada, especialmente em grandes organizações. Requisitos de formação pode ser bastante oneroso.

Segurança: Mesmo com salvaguardas no lugar, pode ser possível para alguns usuários não autorizados acessar o banco de dados. Em geral, o acesso

de banco de dados é uma proposição de tudo ou nada. Uma vez que um usuário não autorizado fica no banco de dados, eles têm acesso a as tabelas, e não apenas algumas. Dependendo da natureza dos dados envolvidos, essas quebras na segurança também podem representar uma ameaça à privacidade individual. Cuidados também devem ser tomados regularmente para fazer cópias de backup das tabelas e armazená-las por causa da possibilidade de incêndios e terremotos que poderiam destruir o sistema.

Qual é a utilidade de um banco de dados

Um banco de dados permite colocar dados à disposição de usuários para uma consulta, uma introdução ou uma atualização, assegurando-se dos direitos atribuídos aos mesmos. Isso é ainda mais útil quando os dados informáticos são numerosos. Um banco de dados pode ser local, ou seja, utilizável em um dispositivo por um usuário, ou repartido, isto é, quando as informações são armazenadas em dispositivos remotos e acessíveis pela rede. A grande vantagem do uso dos bancos de dados é a possibilidade de poderem ser acessados por vários usuários, simultaneamente.

Exercícios

1 - Construir um modelo de entidades e relacionamentos (MER) para o banco de dados de uma clínica abaixo:

- Cada médico que trabalha na clínica é identificado pelo seu CRM, um nome, uma data de admissão e um salário.
- Para todo paciente internado na clínica são cadastrados alguns dados pessoais: código, nome, RG, CPF, endereço e telefone para contato.
- Um paciente tem sempre um médico como responsável, com um horário de visita diário predeterminado.

- Pacientes estão sempre internados em quartos individuais que são identificados por um número e está em um andar da clínica.

2 - Um médico passa várias receitas. Cada receita é passada apenas por um médico. Uma receita pode ter vários medicamentos discriminados, mas, cada receita só pertence a um doente. Cada doente pode ter várias receitas.

3 - Construir um modelo de entidades e relacionamentos (MER) para um banco de dados de uma empresa contendo (tipo de produto, fornecedor, cliente, venda e loja).

Suponha que:

- Um tipo de produto pode ser fornecido por vários fornecedores e um fornecedor pode fornecer vários tipos de produto.
- A venda a um cliente pode conter vários tipos de produto e um tipo de produto pode fazer parte de várias vendas.
- A venda a um cliente é realizada em uma das lojas da empresa.
- Cada produto numa venda tem preço e quantidade.
- Cada venda tem número da nota fiscal, data e valor total.
- Cada tipo de produto tem código, nome e descrição.
- Cada cliente tem código, nome e endereço.
- Cada fornecedor tem código, nome, CNPJ e endereço.
- Cada loja tem código, nome e CNPJ.

4 - Construir um modelo de entidades e relacionamentos (MER) para uma companhia de seguros de automóveis com um conjunto de clientes, onde cada um possui um certo número de automóveis. Os dados do cliente são código, nome, RG, CPF, endereço e telefone. Do carro deve-se armazenar a placa, código RENAVAN, fabricante, modelo e ano. Associado a cada automóvel há um histórico de ocorrências. Cada ocorrência deve ter um número (único), data, local e descrição.

Autores do capítulo

André Luiz Assunção

Ericson Silva

Prototipação

Alinhar os requisitos com o cliente muitas vezes não é uma tarefa fácil. A comunicação é complexa e depende de vários fatores para que ela seja bem-sucedida e ambos os falem a mesma língua. Sendo assim, uma das ferramentas mais eficazes na consolidação da elicitação é a prototipação. Mas afinal, o que é prototipação? Prototipação e os protótipos são ferramentas de ouro para a validação de uma ideia. Os protótipos podem vir em forma de

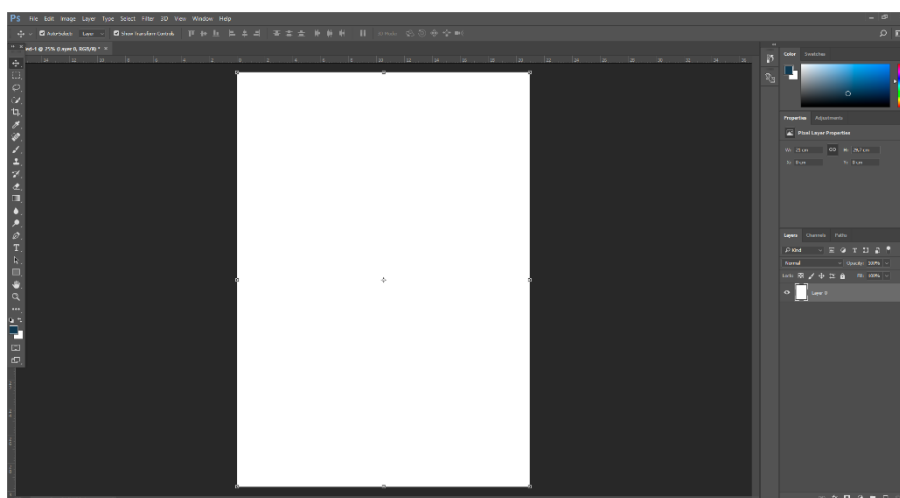
Wireframes e rascunhos



São protótipos de baixa fidelidade, rápidos para se desenvolver e modificar.

Ferramenta indicada: Papel e Caneta

Protótipos visuais

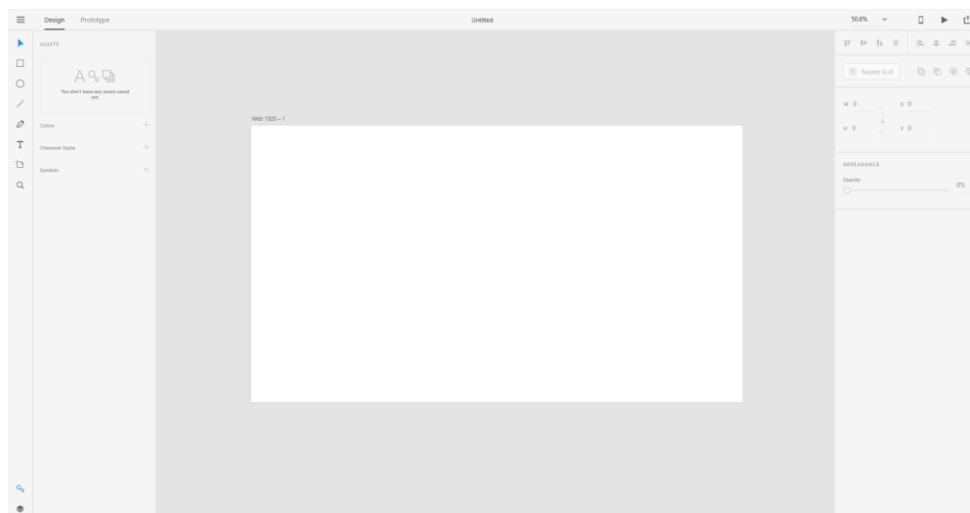


Photoshop / SMN Tecnologia da Informação 2018

Criados com programas de edição gráfica.

Ferramenta indicada: Adobe Photoshop

Protótipos interativos



Adobe XD / SMN Tecnologia da Informação 2018

São protótipos completos e representativos.

Ferramenta indicada: Adobe XD

Usabilidade

Uma simples explicação para o termo usabilidade pode ser definida pela facilidade com que os usuários utilizam uma interface. Derivada da IHC (Interação Humano Computador) em que as questões de design da interface e funcionalidade coincidem para tornar a experiência com o sistema agradável para com o usuário, o termo em questão trata de certificar que a forma como a interface é apresentada seja usual e esteja de acordo com ISO 9241, que constata que um sistema, para ter usabilidade deve operar com Eficácia, Eficiência e Satisfação.

Explicando os termos, se o sistema funciona com eficácia significa que ele executa suas funcionalidades de forma correta e completa, ser eficiente é o quanto ele gasta de recursos para se chegar a eficácia, recursos estes que

podem ser dinheiro, produtividade ou memória, e a satisfação de um sistema é referente ao nível de conforto que o usuário tem e sente ao utilizar o sistema.

Iremos apresentar algumas boas práticas que se tornam essenciais e ajudam na produção de um software com usabilidade:

1- Diálogos Simples e Naturais

As interfaces de usuários devem ser o mais simples possível. Deve-se apresentar exatamente a informação que o usuário precisa – nem mais nem menos – na hora e lugar exatos onde é necessária.

2- Falar a Linguagem do Usuário

A terminologia da interface deve ser baseada na linguagem do usuário. Deve ser expressado com palavras, frases e conceitos familiares ao usuário ao invés dos termos originados do sistema.

3- Minimizar a Sobrecarga de Memória do Usuário

O sistema deve exibir elementos de diálogo para o usuário e permitir que o mesmo faça suas escolhas, sem a necessidade de lembrar de um comando específico.

4- Consistência

Um mesmo comando ou uma mesma ação terá sempre o mesmo efeito. A mesma operação deverá ser apresentada na mesma localização em todas as telas e deverá ser formatada da mesma maneira para facilitar o reconhecimento.

5- Feedback

O sistema deverá informar continuamente ao usuário sobre o que ele está fazendo. O tempo de resposta influi no tipo de feedback que deve ser dado ao usuário. Dez segundos (10s) é o limite para manter a atenção do usuário focada no diálogo.

6- Saídas Claramente Marcadas:

De modo a fazer com que o usuário sinta que pode controlar o sistema, deverá ser fácil abortar uma tarefa ou desfazer uma ação.

7- Atalhos

Os sistemas devem conter atalhos para usuário experiente executar mais rapidamente operações frequentemente utilizadas.

8- Boas mensagens de erro

As mensagens de erro devem ter linguagem clara e sem código, devem ser precisas e ajudar o usuário a resolver o problema. Não devem intimidar ou culpar o usuário.

9- Prevenir Erros

Melhor do que possuir boas mensagens, é evitar situações de erro. Conhecer as situações que mais provocam erro e modificar a interface para que estes erros não ocorram.

10- Ajuda e Documentação

O melhor é que um sistema que seja tão fácil de usar que não necessite de ajuda ou documentação. No entanto, se preciso, esta ajuda deve estar facilmente acessível on-line.


User experience (UX) ou Experiência do usuário


O termo surgiu na década de 1990 e foi idealizado por Donald Norman, com intuito de cobrir todos aspectos da experiência de uma pessoa com o sistema, incluindo industrial, gráficos e interface, a interação física e a manual.

O termo caracteriza como um usuário se sente ao usar, no nosso caso, um software, mas não significa que o termo se restringe apenas ao meio de tecnologia da informação. O princípio de se trabalhar com a experiência do usuário é ser empático ao usuário, pensar como ele e se colocar atrás da tela do computador como aquele usuário ingênuo e/ou leigo que não tem muita

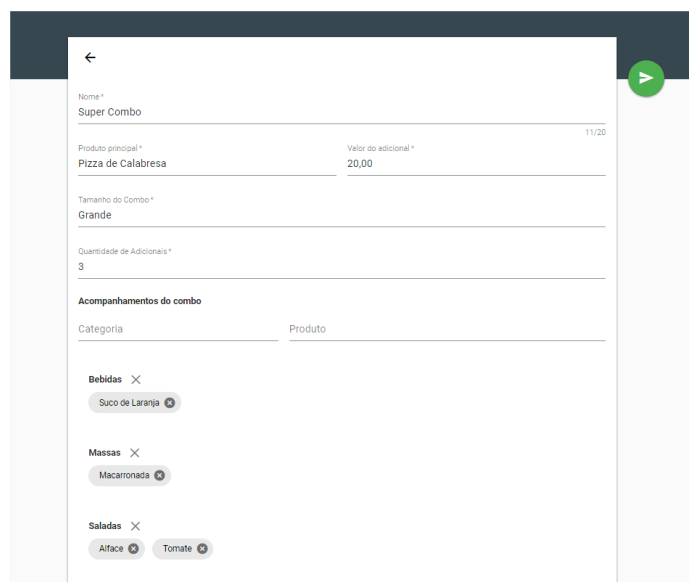
informação de como fazer suas tarefas, a empatia serve para pensar, durante a construção do sistema, como um usuário reagiria em certos casos, pensar em como ajuda-lo da melhor forma possível para que ele consiga realizar todas suas tarefas com eficácia.

Um dos fatores que podem auxiliar o usuário é fazer com que a interface do sistema, seja intuitiva, ou seja, usar elementos, sejam elas palavras, figuras, ícones etc., capazes de passar ao usuário um significado coerente a aquilo que ele faz, por exemplo:

No caso abaixo, o ícon.  representa voltar, até mesmo para um usuário sem muita experiência uma flecha representa voltar, seja por experiências no cotidiano ou etc.

O ícone  representa exclusão, uma vez que a cultura em que vivemos nos leva a pensar

que o X representa fechamento, tal como usado no Windows o que nos fez acostumar com esse padrão.



SMNDelivery – em desenvolvimento 12/2017

Para pensar em como produzir um produto capaz de proporcionar ao usuário uma experiência de usuário positiva, podemos nos perguntar “Por que?”, por que um usuário compraria esse produto, por qual motivo? “O que? ”, “O que

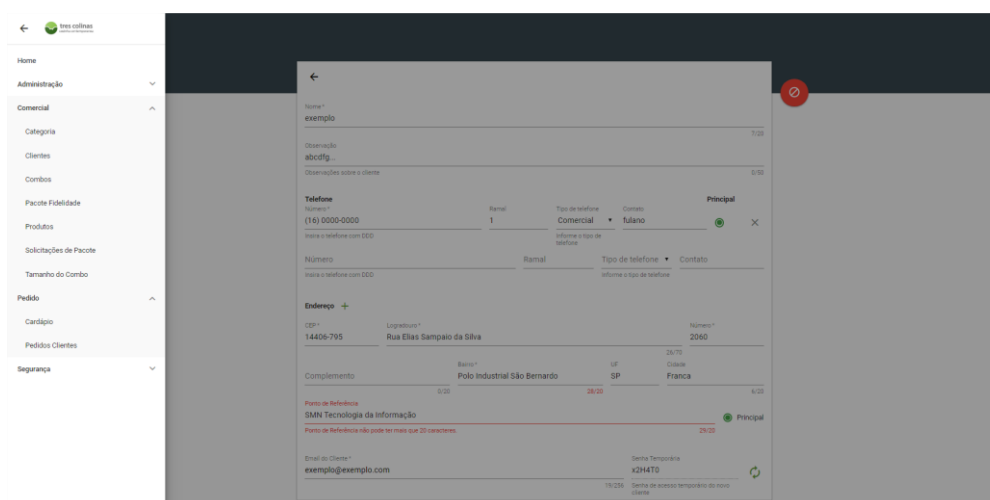
o usuário poderá fazer com esse produto? ” e por fim “Como?”, como faremos o produto com um design esteticamente agradável?

Material design

O material design foi anunciado em 2014 pela empresa que o idealizou e o criou, a Google. Trata-se de uma linguagem de design visual, e como dito pelo própria Google, o Material é uma metáfora, inspirado na realidade, em papel e tinta, usando de flexibilidade, sombras e movimentos. Elementos como tipografia, imagens, grids demonstram mais do que apenas design, eles criam uma hierarquia e uma organização visual e os movimentos são capazes de entreter o usuário de uma forma rápida e limpa.

Para o tema de prototipagem, o material design é essencial porque como dito, trata-se de uma forma visual agradável aos olhos e diversos dos produtos e serviços oferecidos pela Google (Gmail, YouTube, Google Drive, Google Documentos, Planilhas e Apresentações, Google Maps) já adotaram essa linguagem, o próprio Google disponibiliza as regras de como se utilizar a estrutura de forma eficaz, disponibilizando formas e especificações de como se criar elementos e seus padrões.

Exemplo de tela de sistema seguindo material design:



SMNDelivery – em desenvolvimento 12/2017

Elementos presentes na imagem:

Floating action button: Um botão de ação flutuante representa a ação principal em um aplicativo, na imagem trata-se do botão vermelho no canto superior direito, é o botão de salvar cliente, porém só será permitido salvar quando todos os campos obrigatórios estiverem preenchidos corretamente.

Tipografia: nota-se que para títulos as cores são diferentes, isso exemplifica a tipografia hierárquica em que se dá maior relevância à aquilo que deve ser lido primeiro

Validação de campo: ajudando o usuário a entender como deve ser preenchido o campo, na imagem trata-se das linhas vermelhas no formulário em que o usuário preencheu de forma errada.

Linha de entrada

A linha de entrada indica onde inserir texto, exibido abaixo do rótulo.

Quando um campo está ativo ou contém um erro, a cor e a espessura da linha variam.

Para ter acesso a todo o material do Material design acesse: <https://material.io/>

Cores

Olhe ao seu redor. O que você vê?

Provavelmente várias cores.

As cores estão presentes em nossas vidas e muitas vezes, são responsáveis pelo o que sentimos e pensamos. Elas podem ser nossas aliadas no desenvolvimento de um software, mas para isso, precisamos conhecê-las.

Amarelo

Remete a clareza, felicidade, otimismo.

Vermelho

Remete a coragem, juventude e excitação.

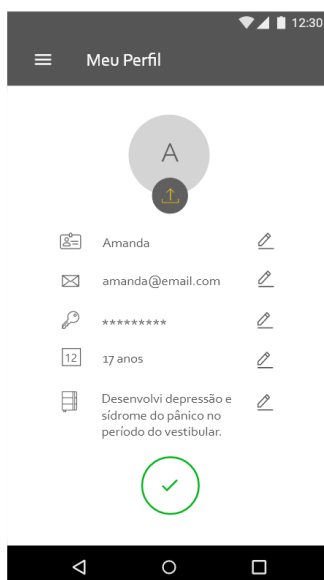
Azul

Remete a segurança, força e confiança.

Verde

Remete a crescimento, saúde e pacificidade.

Essas cores podem ser utilizadas para tornar o sistema ou aplicativo intuitivo. Por exemplo, em botões de adição, você pode utilizar a cor verde que indiretamente indicará uma ação positiva e sem muitos riscos.



Sim à Vida / Aplicativo mobile SMN Tecnologia da Informação – Em desenvolvimento 11/2017

Trazer cores para o cotidiano da prototipação é se preocupar com a melhor experiência que o usuário pode ter. É fazer com que ele se sinta familiarizado e confortável ao navegar por aí.

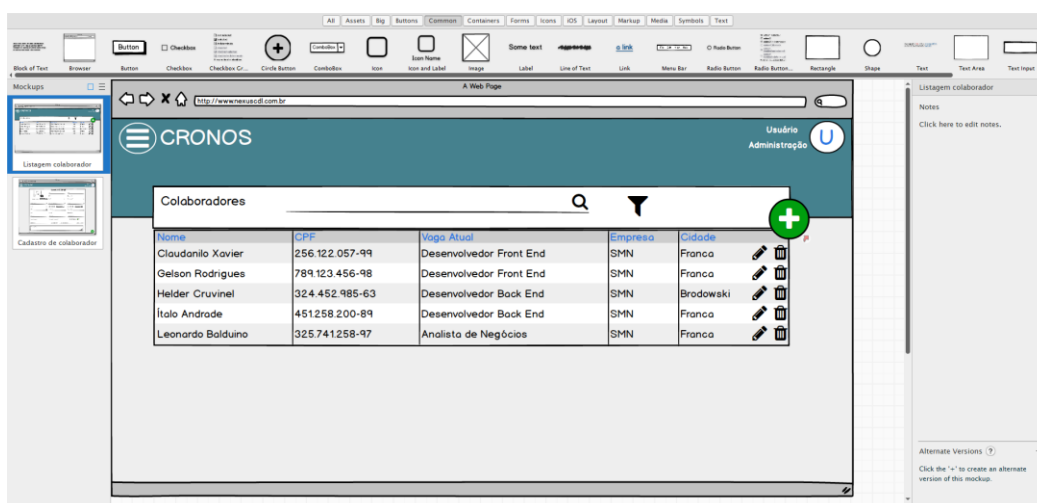
Ferramentas

Atualmente, é comum a prática da utilização de protótipos interativos para a realização de validações com os clientes.

Com o mercado cada vez mais acirrado, ganhar a confiança das partes interessadas é essencial e pensando nisso, ao longo do tempo, ferramentas foram sendo desenvolvidas para que cada vez mais os protótipos se tornem a representação perfeita de um software ou aplicativo.

Porém, antes da utilização de protótipos interativos, era bastante comum a prototipação ser feita através de wireframes ou rascunhos de papel e caneta. Mas como isso estava se tornando um processo trabalhoso e demorado, foi-se desenvolvido o Balsamiq.

A função do Balsamiq era e ainda é simular telas que seriam desenhadas a mão, se aproximando da realidade papel e caneta.



CRONOS / SMN Tecnologia da Informação – Em desenvolvimento 05/2016

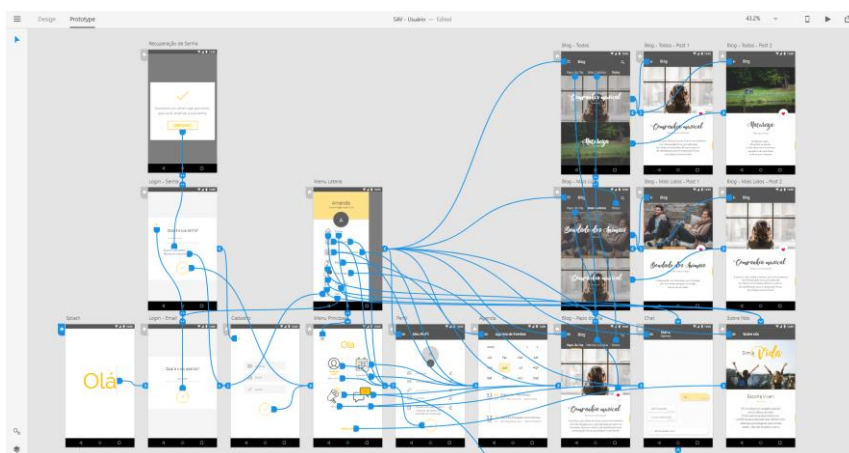
Algumas empresas ainda utilizam wireframes, mas com a ascensão de ferramentas que praticamente simula a realidade, o Balsamiq está se tornando obsoleto.

Ferramentas de prototipação interativa

Existem inúmeras opções para se criar um protótipo interativo. Aqui estão três opções para que seu cliente fique completamente confiante e satisfeito:

Adobe XD

Ele trabalha com *artboards* que podem se ligar entre si, dando uma visão geral de navegação.



Sim à Vida / Aplicativo mobile SMN Tecnologia da Informação – Em desenvolvimento 11/2017

Exercícios

Para que a prototipação de torne algo natural e mais fácil é necessário que exercitemos a criatividade.

Então... vamos começar?

Inicie um parágrafo com “Eu me lembro ...” e conte uma memória.

Por fim, desenhe algo que resuma sua memória.

Exercitar a criatividade é mais fácil do que imaginamos, você pode aprender um novo instrumento, ler um livro, tirar um tempo apenas para pensar. Pequenos exercícios fazem com que nosso cérebro trabalhe e isso nos faz enxergar o que ninguém está enxergando.

Autores do capítulo

Wilson Guiraldelli

Maria Rita Benate

Documentação

De extrema importância para um processo de desenvolvimento de um software, a documentação de requisitos é responsável por trazer aos envolvidos no projeto de forma escrita, o que o sistema deve prover de funcionalidades. Sendo acessível a todos os envolvidos no processo, tal como os stakeholders, os programadores, o analista de negócios e testadores, a especificação dos requisitos, outro nome dado a documentação, deve seguir critérios de qualidades para que ela passe aos membros do projeto, aquilo que o cliente realmente deseja. Neste capítulo da apostila, iremos tratar a respeito da documentação/especificação de requisitos, sua importância, seus critérios de qualidade, seus usos e o modelo User Stories utilizado em metodologias ágeis.

Por que é importante documentar os requisitos?

Requisitos formam a base para o desenvolvimento do sistema!

Isso quer dizer que o documento de requisitos traz quais são as funcionalidades do sistema, especificando de forma clara como dada funcionalidade deve ser exercida, ou seja, a partir dela que um programador vai começar a desenvolver o projeto.

Requisitos tem relevância legal

Em alguns casos, a documentação pode ser o contrato entre a empresa que fabrica o software e o cliente que a contrata, por exemplo serviços prestados à órgãos públicos. Nestes casos, se a documentação não for seguida, processos jurídicos podem ser gerados.

Documentos de requisitos são complexos

A complexidade de uma especificação não se dá pelo seu tamanho, mas sim pela relação entre os requisitos dentro dela, uma documentação pode conter poucos requisitos, mas diversas relações entre eles tornando-a mais complexa, então, para uma documentação complexa se tornar mais simples, é necessário especificar as relações entre os requisitos.

Requisitos devem ser acessíveis por todas partes envolvidas no projeto

Todos aqueles envolvidos no processo de produção devem ter acesso ao documento porque é nele que estão contidas as descrições das funcionalidades que devem ser produzidas.

Compartilhar uma visão única dos requisitos

Dentre as maiores importâncias de uma especificação, compartilhar a mesma visão dos requisitos é a que se deve ter a maior atenção, porque é papel do analista de negócios ou engenheiro de requisitos, entender como o stakeholder deseja aquela funcionalidade, ou seja, ele deve entender e ter a mesma visão do cliente para que ele passe para os demais membros do time, programadores, testadores etc., esta visão e a coloque de forma clara na documentação para que todos tenham a mesma ideia a respeito do sistema a ser desenvolvido, desta forma, satisfazendo o cliente.

O que uma documentação deve conter?

A documentação nada mais é que a especificações dos requisitos/funcionalidades que o software deve conter, portanto são os requisitos que precisam estar presentes na documentação, dentre as classificações que requisitos recebem, veremos as principais:

Requisitos funcionais: são aqueles requisitos que representar e descrevem funcionalidades que o sistema deve realizar, descrevem as ações e como devem ocorrer.

Exemplo: Cada pedido deve ser associado a um identificador único (PID), o qual o usuário pode copiar para a área de armazenamento permanente da conta

Requisitos não-funcionais ou requisitos de qualidade: representam as qualidades ou restrições que o sistema deve conter. Em alguns casos, requisitos não funcionais podem ser mais críticos caso não atendidos.

Exemplo: o sistema deve operar com 10 acessos simultâneos

Critérios de qualidade para uma documentação

Uma boa documentação deve seguir critérios de qualidade, dentre esses critérios estão:

Não ambiguidade e consistência:

Um requisito não pode ser interpretado de formas diferentes, é necessário que ele seja consistente e permita apenas uma interpretação e um requisito individual não pode contradizer outros requisitos.

Estrutura clara:

O documento deve ser claro e bem estruturado, permitindo uma leitura seletiva.

Modificável e extensível:

A estrutura deve permitir mudanças de forma fácil e entendível, o documento deve ser extensível para maior facilidade de gerenciamento de versionamento do projeto.

Completeness:

Deve conter todos requisitos importantes para o sistema, tanto os funcionais quanto os não-funcionais; identificação dos dados de entrada e fatores que influenciam, além das reações exigidas do sistema.

Rastreabilidade:

O documento deve conter os relacionamentos com documentos externos que influenciam no sistema, tais como documentação de sistemas legado, leis, fórmulas matemáticas etc.

Para que a documentação é usada?

- Planejamento
- Projeto arquitetônico
- Implementação
- Teste
- Gerenciamento de mudanças
- Uso e manutenção do sistema
- Gerenciamento de contrato

User Stories

As user Stories ou histórias de usuário, é um modelo/estrutura de documentação que se adequa aos chamados processos de desenvolvimento ágil de software, trazendo uma nova maneira de escrever os requisitos de um sistema, mais efetiva e até mesmo mais divertida.

Tido como um processo repetitivo, semelhante a uma linha de produção de uma fábrica, a produção de um software na verdade é um processo complexo, e os desenvolvedores não aplicam um processo repetitivo forma, para X funcionalidade um programador pode pensar diversas maneiras de se realizar a tarefa. Por causa disto, é necessário que passemos a ele (s) não apenas o que fazer, mas também para quem e por que, assim ele sabe o que fazer e a melhor maneira de fazer para atender ao usuário. Para escrever a documentação usando estes tópicos, usamos:

Sendo (Quem?): para quem está sendo desenvolvida a funcionalidade

Posso (o que?): descrevendo a funcionalidade em si

Para que (por que?): descrevendo o motivo daquela funcionalidade e se possível o ganho de negócio por conta dela

Abaixo, um exemplo de como e por que usar a User Stories:

SENDO um vendedor que realiza 50 visitas por dia
POSSO consultar as últimas compras de cada cliente
PARA QUE ao chegar no cliente eu possa consultar qual foi sua última compra, e assim conseguir negociar com ele estando melhor informado.

Com estas informações, o desenvolvedor vai conseguir trabalhar “mais armado”, e provavelmente vai criar uma funcionalidade mais bem elaborada do que se recebesse apenas a necessidade do cliente sem o detalhamento de quem vai usar e por que vai usar

Para uma User story ficar completa, usamos os Cenários e a técnica BDD (Behavior Driven Development) de Dan North, onde as palavras **DADO QUE**, **QUANDO** e **ENTÃO** ajudam a descrever detalhadamente como uma parte o sistema deve reagir a uma ação, desta forma também influenciando na forma de testar.

Abaixo um exemplo de como o BDD trabalha na user story:

Cenário 1: Estoque disponível

Dado que o estoque da Coca-Cola é de 50 unidades

Quando informo uma venda de 40 unidades

Então a venda é registrada

Também podemos usar os termos “E” e “OU” para tornar a documentação mais abrangente e completa, veja abaixo o exemplo já mostrado acrescido desses termos:

Cenário 1: Estoque disponível

Dado que o estoque da Coca-Cola é de 50 unidades

Quando informo uma venda de 40 unidades

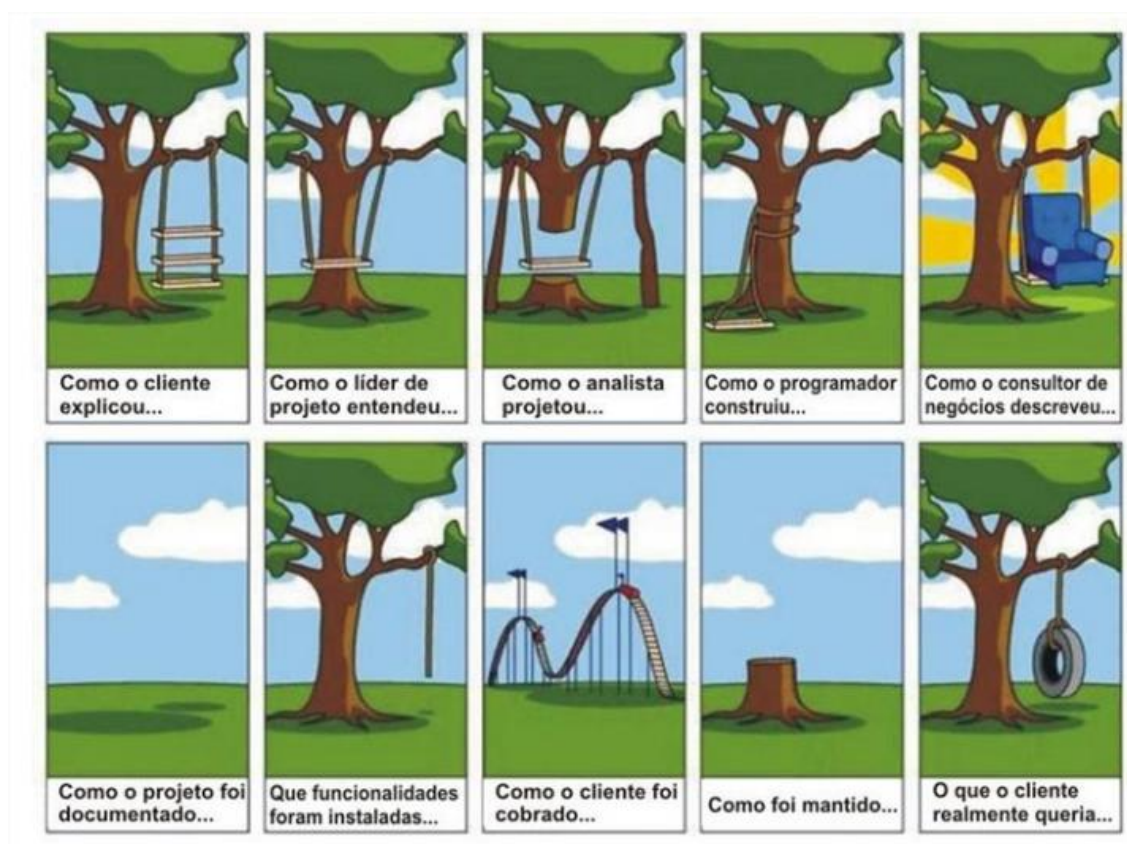
Então a venda é registrada

E o estoque passa a ser de 10 unidades

Textual

Outra forma de especificar os requisitos é utilizando a forma textual, aparentemente é um pouco mais fácil que as User Stories por se tratar de uma descrição textual simples, porém é necessário tomar muito cuidado com a ambiguidade e sempre ter certeza da consistência de seu texto.

Escrever algumas linhas detalhando as funcionalidades do sistema acaba sendo uma forma mais amigável para o cliente/usuário entender o que irá ser desenvolvido no sistema, mas corremos um grande risco de existir mais de um entendimento entre o cliente, analista e o desenvolvedor.



Fonte: Luís Gonçalves(2009)

Uma das formas de se evitar a ambiguidade em seus textos é uso de protótipos logo após a descrição e detalhar com a maior clareza possível as funcionalidades do sistema, utilizando-se ou não de tópicos. Os protótipos transmitem ao leitor (cliente, analista ou desenvolvedor) uma visão mais real do objetivo do software, assim conseguimos deixar mais claro como e oque desenvolver.

Exercícios

1 - Vamos imaginar que você trabalha em um sistema bancário de autoatendimento (caixa eletrônico). Seu cliente envia para você um e-mail solicitando e explicando como funciona o saque do banco:

Olá! Precisamos disponibilizar a operação de saque no caixa eletrônico. Segue as regras do banco para saques em caixas eletrônicos: -

- Por questões de segurança o valor máximo de cada saque é de 800,00;
- Os saques só estão liberados entre 6h00min e 22h59, em qualquer dia, útil ou não;
- O saldo do cliente não pode ficar negativo, exceto se ele possuir limite de cheque especial; - O cliente jamais poderá ultrapassar seu limite de cheque especial;
- Pode ser impresso um comprovante de saque ao final da operação (se o cliente assim desejar).

Como você transformaria este e-mail do cliente em uma história de usuário?

2 – Agora, você trabalha conosco da SMN Informática e é um dos analistas de negócios da nossa equipe. Você está encarregado de fazer a documentação da página de pedidos do SMNDelivery, um sistema para pedidos de um restaurante, o cliente exigiu que tenha na tela de pedidos:

- Um campo para busca de cliente, caso o cliente já tenha cadastro os campos de telefone e endereço serão preenchidos com os dados

cadastrados, se não, deve haver a opção de cadastro e a opção de fazer pedido sem cadastro

- Um campo para busca dos pratos oferecidos pelo restaurante
- Campo de telefone
- Campo de endereço, mas para cada pedido a opção de o usuário retirar o pedido na loja
- Forma de pagamento
- Deve ser atualizado o valor do pedido a cada item adicionado
- Opção concluir pedido (trazendo dados do pedido e preço final)
- Você tem a tarefa de documentar essas funcionalidades de maneira detalhada para passar ao cliente.

Autores do capítulo

Felipe Araújo

Wilson Guiraldelli

GIT

Um projeto de desenvolvimento de software as vezes necessita de tempo e de pessoas trabalhando, em alguns casos as pessoas estão fisicamente longe uma das outras, cada colaborador faz uma parte separada do projeto para depois unir todas as partes, como isso muitas alterações e novas partes desses projetos são feitas com frequência, o que pode complicar muito a organização dos arquivos manipulados.

Sistema de Controle de Versão

Para organizar esses arquivos pessoas e empresas utilizam um Sistema de Controle de Versão que armazenas os arquivos e os controla por colaborador, partes alteradas e versões diferentes, sendo possível a qualquer momento enviar, baixar, alterar ou excluir arquivos. Esses arquivos podem ser basicamente de qualquer tipo, como documentos, imagens e código fonte de algum programa.

Tipos de sistemas

Atualmente existem diversos tipos de versionadores no mercado, cada um tem suas qualidades e facilidades de uso, os sistemas mais comuns são: Mercurial, Subversion, Git, etc. iremos falar apenas do Git por ser um sistema de controle de versão distribuído.

Git

É um Sistema de controle de versão distribuído, foi desenvolvido por Linus Torvalds em 2005, a grande maioria dos projetos opens source são controlados pelo Git e ficam armazenas em um repositório de arquivo na web através do serviço GitHub, o Git é fácil de aprender, utilizar e trabalha com versionamento descentralizado.

Repositórios digitais

São coleções de informação digital, que podem ser construídas de diferentes formas e com diferentes propósitos. Podem ser colaborativos e com

um controle suave dos conteúdos e da autoridade dos documentos, tal como as dirigidas para o público em geral (a Wikipedia é um exemplo). Mas podem, também, ter um alto nível de controle e ser concebidos para públicos específicos. A criação destes repositórios obriga a um enorme trabalho de colaboração entre professores bibliotecários, professores, alunos e outros agentes sociais.

Vantagens de utilizar um repositório de arquivos (GitHub)

Segurança: Cada software de controle de versão de mecanismos para evitar possíveis corrupções em arquivos. Além disso, apenas pessoas autorizadas e identificadas podem mexer no código fonte controlado.

Versionamento: Caso se deseje voltar a versão de um determinado arquivo por algum erro cometido ou simplesmente mudança de escopo, é possível fazê-lo de forma simples e estruturada, minimizando eventuais erros e efeitos colaterais.

Rastreabilidade: Quando se trata de algo importante, é sempre interessante saber “Quem”, “Quando”, “Como”, “Por que” e “Onde”. Todos esses metadados estão disponíveis nas ferramentas mais populares de controle de versão.

Organização: Os sistemas que possuem interface visual disponibilizam uma visualização completa do ciclo de vida de cada arquivo controlado, desde sua criação até o momento atual.

Colaboração: O trabalho em equipe, principalmente as distribuídas, é muito facilitado. Pessoas que talvez nem se conhecem pode colaborar num determinado projeto cujo repositório central é disponibilizado a todos os envolvidos.

Confiança: O uso de repositórios remotos ajuda muito na recuperação de eventos imponderáveis. Situações do tipo “Perdemos o projeto inteiro que estava na máquina de fulano” são minimizadas. Além disso, é possível testar novas ideias sem danificar a linha base do desenvolvimento.

Sistema de Controle de Versão Distribuído

Sistemas de Controle de Versão Distribuídos (Distributed Version Control System ou DVCS). Os clientes não apenas fazem cópias das últimas versões dos arquivos: eles são cópias completas do repositório. Assim, se um servidor falhar, qualquer um dos repositórios dos clientes pode ser copiado de volta para o servidor para restaurá-lo.

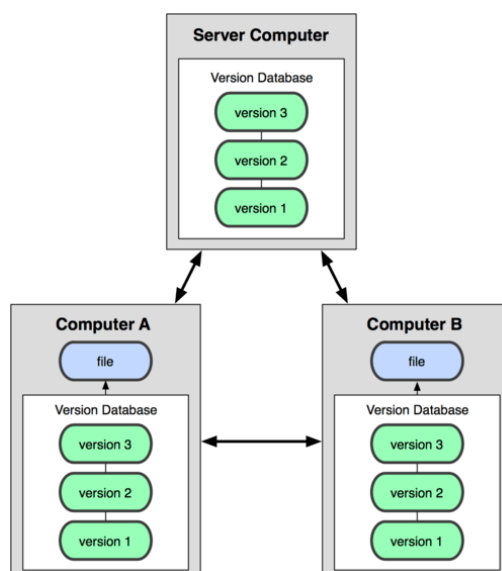


Figura 1-1. Diagrama de Controle de Versão Distribuído.

Fonte: Scott Chacon and Ben Straub (2014, p. 7).

Além disso, muitos desses sistemas lidam muito bem com o aspecto de ter vários repositórios remotos com os quais eles podem colaborar, permitindo que você trabalhe em conjunto com diferentes grupos de pessoas, de diversas maneiras, simultaneamente no mesmo projeto. Isso permite que você estabeleça diferentes tipos de workflow (fluxo de trabalho) que não são possíveis em sistemas centralizados, como por exemplo o uso de modelos hierárquicos

Como o Git Trabalha:

Você modifica arquivos no seu diretório de trabalho.

Você seleciona os arquivos, adicionando eles para sua área de preparação.

Você faz um commit, que leva os arquivos como eles estão na sua área de preparação e os armazena permanentemente no seu diretório Git.

Se uma versão particular de um arquivo está no diretório Git, é considerada consolidada. Caso seja modificado, mas foi adicionada à área de preparação, está preparada. E se foi alterada desde que foi obtido, mas não foi preparado, está modificada.

Locais dos Arquivos

Git faz com que seus arquivos sempre estejam em um dos três estados fundamentais: consolidado (committed), modificado (modified) e preparado (staged). Dados são ditos consolidados quando estão seguramente armazenados em sua base de dados local. Modificado trata de um arquivo que sofreu mudanças, mas que ainda não foi consolidado na base de dados. Um arquivo é tido como preparado quando você marca um arquivo modificado em sua versão corrente para que ele faça parte do próximo commit (consolidação).

Isso nos traz para as três seções principais de um projeto do Git: o diretório do Git (git directory, repository), o diretório de trabalho (working directory), e a área de preparação (staging area).

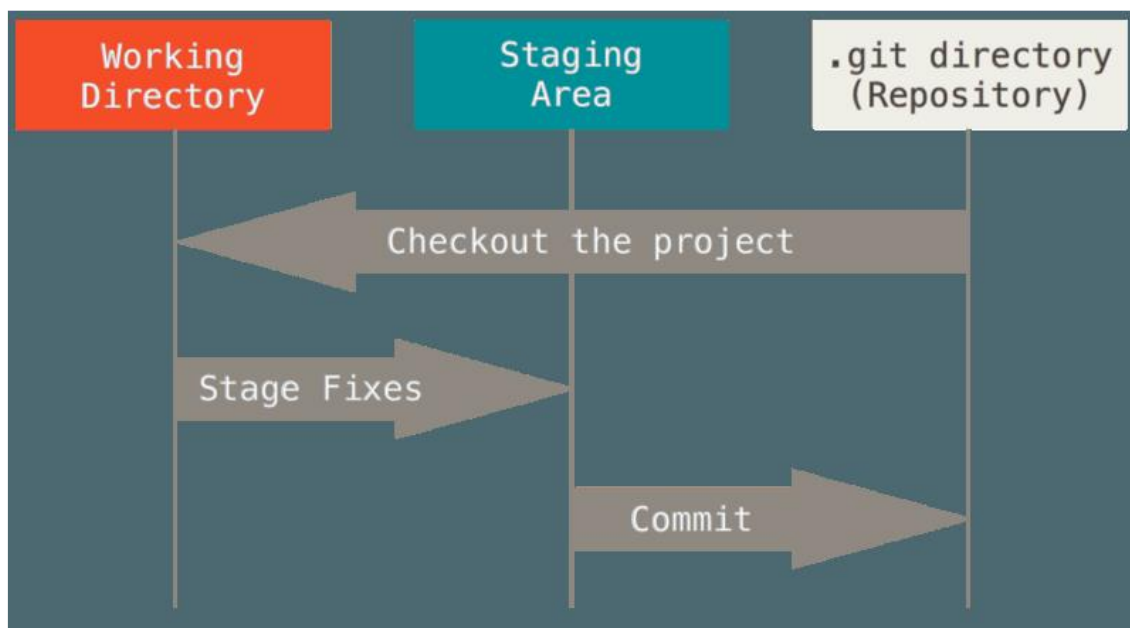


Figura 1-2. Diretório de trabalho, área de preparação, e o diretório do Git.

Fonte: Scott Chacon and Ben Straub (2014, p. 16).

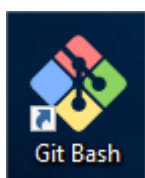
Instalando o Git

O Git pode ser instalado em diferentes sistemas operacionais, basta entrar na url abaixo e escolher sua versão e seguir o passo a passo da sua versão.

<https://git-scm.com/downloads>

Utilizando o Git

Em seu sistema operacional abra o programa Git-Bash



Será aberto uma tela para contendo o nome de usuário que você entrou no seu sistema e nome do computador que você está utilizando e na linha de baixo o símbolo \$ onde iremos digitar os comandos do git.

Comandos do Git

Abaixo serão apresentados alguns comandos do git, nessa apostila será mostrado apenas a função principal que cada comando executa, lembrando que a maioria deles devem ser combinados com tags ou complementos, os detalhes de suas funcionalidades serão apresentas no curso básico presencial oferecido pela empresa SMN.

Definindo um usuário

A primeira coisa que você deve fazer quando instalar o Git é definir o seu nome de usuário e endereço de e-mail. Isso é importante porque todos os commits no Git utilizam essas informações, e está imutavelmente anexado nos commits que você realiza:

\$ git config --global user.name "Davi Lucas"


```
$ git config --global user.email davilucas@example.com
```

Inicializando um Repositório em um Diretório Existente

Caso você esteja iniciando o monitoramento de um projeto existente com Git, você precisa ir para o diretório do projeto e digitar o comando:

```
$ git init
```

Verificando o Status de Seus Arquivos

A principal ferramenta utilizada para determinar quais arquivos estão em quais estados é o comando git status.

```
$ git status
```

Pedindo ajuda

Outro comando muito utilizado pelos iniciantes é o comando help, que você digitar para verificar os comandos existentes.

```
$ git help
```

Monitorando novos arquivos

Para passar a monitorar um novo arquivo que você criou ou alterou, use o comando git add e em seguida o comando commit.

```
$ git add "NomeDoArquivo.txt" ou
```

```
$ git add . (Adiciona todos os arquivos.)
```

Commit

A melhor tradução de commit, pensando no seu uso em Git, é fechar sob segurança. Quando um commit é feito, cria-se um instante na linha do tempo que salva o estado do projeto.

\$ git add

\$ git commit -m 'Projeto com a versão Inicial'

Clonando um Repositório Existente

Caso você queira copiar um repositório Git já existente, por exemplo, um projeto que você queira contribuir o comando necessário é git clone. Você clona um repositório com git clone [url]. Por exemplo, caso você queria clonar a biblioteca Git do Ruby chamada Grit, você pode fazê-lo da seguinte forma:

\$ git clone git://github.com/schacon/grit.git

Branch

São ramificações (“Área de trabalho”) que podemos criar em paralelo do que já está em produção ou na Master. Usamos branches para colocar arquivos, codificar alguma funcionalidade específica ou então para corrigir bugs, por exemplo.

\$ git branch Teste (Cria uma “Área de Trabalho” Teste)

Merge

Utilizado para mesclar arquivos entre diferentes branches

\$ git merge Teste (Mescla minhas alterações com a base do cliente)

Pull

Atualiza seu repositório local com a versão mais nova que está no repositório remoto

\$ git pull < origin > <master>

Fetch

Atualiza seu repositório local com a versão mais nova que está no repositório remoto porem não mescla os arquivos que estão diferentes

\$ git pull < origin > <master>

Push

Envia seus arquivos para o repositório remoto

\$ git push < origin > <master>

Log

Mostra o historio de alterações de commit

\$ git log

Show

Mostra vários tipos de objetos

\$ git show

Diff

Verifica as modificações realizadas entre verões de um arquivo

\$ git diff

Exercícios

1 - Instalando GIT

a) baixar e instalar o git de acordo com seu sistema operacional, no site <https://git-scm.com>

b) abrir o git bash, criar um usuário e e-mail

\$ git config --global user.name "Ezequiel Elias Alves"

\$ git config --global user.email "ezequiel@smn.com.br"

\$ exit (Para sair do bash)

2 – Preparando o GIT

a) Criar as seguintes pastas C:\git\AreaLocal

b) Entrar na pasta C:\git\AreaLocal clicar com o botão direito do mouse escolher a opção:

Git Bash here

c) Verifique o status **\$ git status**

d) Dentro do git bash digitar o seguinte comando: **\$ git init**

e) Verifique o status

3 – Adicionando arquivos

a) Crie os arquivos chamados arquivo01.txt , arquivo02.txt dentro da pasta

C:\git\AreaLocal

b) Digite o comando: **\$ git status** , para verificar os arquivos.

c) Digite: **\$ git add .** , para adicionar todos os arquivos para a área de monitoramento :

d) Digite: **\$ git status**

4 - Commit

Digite: **\$ git commit -m "arquivos sem alteração"**

\$ git status (para verificar)

\$ git log (para verificar os commit)

5 - .gitignore (ocultar arquivos)

a) Abrir o bloco de notas e criar um arquivo chamado .gitignore (do tipo Todos os arquivos, não vai ser um txt) dentro da pasta C:\git\AreaLocal

b) Abrir o arquivo .gitignore com o bloco de notas e escrever arquivo02.txt, fechar e salvar o bloco de notas

c) Dentro do git bash verifique os arquivos com o comando: **\$ git status**

6 – Alterando arquivo e fazendo commit

Abrir o arquivo01.txt e acrescentar os seguintes dados: Nome:" Nome completo", Idade:.

Dentro do bash verificar o status

Faça os comandos: **\$ git add .** (enter)

\$ git commit -m "Arquivo alterado 01" (enter)

\$ git log (para verificar o commit)

7 - Comparando commit com git diff

Dentro do git bash verifique os commits com o **git log**

Compare os arquivos com o comando **\$ git diff "chave do commit 1" .. "chave do commit 2)**

8 - Criando uma nova branch

Abra o git bash use o comando **\$ git branch "AreaTeste"**

\$ git checkout AreaTeste (Para entrar na nova branch)

\$ git checkout (Para verificar as branch)

9 - Alterando o arquivo01.txt

Abra o arquivo arquivo01.txt e acrescente seu e-mail e telefone, salve e feche o arquivo

Dentro git bash use os seguintes comandos: **\$ git add .**

\$ git commit -m "fone e e-mail"

\$ git log para verificar

10 - Verificando o arquivo alterando dentro da branch master

Dentro do git bash entre no branch master com o comando: **\$ git checkout master**

Abra o arquivo01.txt e verifique suas alterações

11 - Unindo os arquivos com merge

Abra o git bash e digit o seguinte comando: **\$ git merge AreaTeste**

Abra o arquivo arquivo01.txt e verifique as alterações

Autores do capítulo

Ezequiel Elias

Andreino Faria

Teste de Software

"Conjunto de atividades planejadas com antecedência e executadas sistematicamente (PRESSMAN, 2016)".

Para que servem os testes de software?

Para trazer à tona erros cometidos inadvertidamente quando o software foi projetado e construído.

Para analisar o controle de qualidade, assegurar que todas suas funcionalidades estão funcionando corretamente.

Quem faz os testes de software?

Os responsáveis pelo planejamento da estratégia de testes de software são os especialistas em testes (testadores), além dos engenheiros de software e o próprio gerente de projeto.

Como funciona?

Inicialmente são realizados testes em um único ou em pequeno grupo de componentes, com o objetivo de descobrir erros nos dados e na lógica do processamento. Após ser realizados testes nesses componentes eles são integrados até que o sistema esteja terminado.

Após essa integração de componentes formando um sistema completo, outros tipos de testes são feitos, para verificar se os requisitos e as funcionalidades estão sendo atendidos.

O que é gerado a partir dos testes de softwares?

É gerado um documento de especificações do teste, que documenta o plano que descreve as estratégias de testes. O processo de se diagnosticar e corrigir os erros descobertos nos testes de sistema se chama Depuração.

Teste e Depuração são atividades diferentes. Porém, a depuração deve estar associada a uma estratégia de software.

Curiosidade: Testes são bases para estimar a qualidade do software, é importante lembrar que a qualidade não é implementável.

Para garantir que o software está em conformidade com as ideias e necessidades do cliente, é necessário a realização de testes. Sabe aquele software que fica travando ou faz o "pc" ficar lento? Na maioria das vezes é porque deve ter passado pelo processo de testes indevidamente. Existem algumas ferramentas que apoiam a etapa de testes. Os tipos de ferramentas de teste de software são tão amplos quantos os tipos de testes que podem ser realizados. Ou seja, para cada teste a ser aplicado é possível encontrar uma aplicação/ferramenta que o auxilie. Por isso, destacamos apenas os principais, que você confere a seguir:

Ferramentas de apoio para testes de software

Teste de funcionalidade: O objetivo é avaliar a interface do programa, os links, campos de preenchimento, botões etc. Uma ferramenta que pode ser útil nesse caso é o Push Test Maker. Para aplicações web, temos o Apodora, o Sikuli e o Watir.

Teste de desempenho: Visa avaliar a performance de um programa, se ele atende aos requisitos mínimos preestabelecidos, como o tempo que demora para dar respostas a ações realizadas e se ele rende como o esperado. Um software que pode ajudar é o Apache Jmeter, que pode ser usado em aplicações Web.

Teste unitário: Visa avaliar pequenas unidades que compõem um software, responsáveis por funções diferentes dentro dele. Podem ser avaliados códigos, sub-rotinas entre outros. O foco aqui é descobrir se todas essas partes estão funcionando adequadamente. As ferramentas que podem ajudar aqui são: NUnit e JUnit (para aplicações Java).

Gestão dos testes – Existem ferramentas que auxiliam no gerenciamento dos testes a serem realizados, permitindo acompanhamento e controle dos casos e atividades de testes. Entre elas temos o TestMaster e o Testlink.

Bugs e defeitos – Ferramentas que são importantes para se encontrar bugs e erros durante o desenvolvimento dos programas. Podemos destacar o Redmine, Mantisbt e o Bugzilla.

A importância dessas ferramentas é que ao automatizarem tarefas repetitivas, podem não só liberar o profissional encarregado para atividades mais importantes, como garantem maior confiabilidade e qualidade ao teste realizado.

Tipos de teste

Teste de Unidade: Teste em um nível de componente ou classe. É o teste cujo objetivo é um “pedaço do código”.

Teste de Integração: Garante que um ou mais componentes combinados (ou unidades) funcionam. Podemos dizer que um teste de integração é composto por diversos testes de unidade.

Teste Operacional: Garante que a aplicação pode rodar muito tempo sem falhar.

Teste Positivo-negativo: Garante que a aplicação vai funcionar no “caminho feliz” de sua execução e vai funcionar no seu fluxo de exceção.

Teste de regressão: Toda vez que algo for mudado, deve ser testada toda a aplicação novamente.

Teste de caixa-preta: Testar todas as entradas e saídas desejadas. Não se está preocupado com o código, cada saída indesejada é vista como um erro.

Teste caixa-branca: O objetivo é testar o código. Às vezes, existem partes do código que nunca foram testadas.

Teste Funcional: Testar as funcionalidades, requerimentos, regras de negócio presentes na documentação. Validar as funcionalidades descritas na documentação (pode acontecer de a documentação estar inválida).

Teste de Interface: Verifica se a navegabilidade e os objetivos da tela funcionam como especificados e se atendem da melhor forma ao usuário.

Teste de Performance: Verifica se o tempo de resposta é o desejado para o momento de utilização da aplicação.

Teste de carga: Verifica o funcionamento da aplicação com a utilização de uma quantidade grande de usuários simultâneos.

Teste de aceitação do usuário: Testa se a solução será bem vista pelo usuário. Exemplo: caso exista um botão pequeno demais para executar uma função, isso deve ser criticado em fase de testes.

Teste de Volume: Testar a quantidade de dados envolvidos (pode ser pouca, normal, grande, ou além de grande).

Testes de stress: Testar a aplicação sem situações inesperadas. Testar caminhos, às vezes, antes não previstos no desenvolvimento/documentação.

Testes de Configuração: Testar se a aplicação funciona corretamente em diferentes ambientes de hardware ou de software.

Testes de Instalação: Testar se a instalação da aplicação foi OK.

Testes de Segurança: Testar a segurança da aplicação das mais diversas formas. Utilizar os diversos papéis, perfis, permissões, para navegar no sistema.

Teste de Fumaça: Esse tipo de teste é bastante usado quando se trata de projetos com prazos crítico, pois permite que a equipe de software avalie o projeto frequentemente.

Teste de Fronteira: Testar o valor limite, devem ser elaborados casos de testes que ultrapassem os limites do software. (Extrapolar os valores máximos e mínimos suportados pelo sistema).

Exemplo: Vamos supor que uma empresa tem as seguintes regras de negócio no que se refere à contratação de funcionários:

De 0 a 16 anos: não contrata

De 16 a 18 anos: contrata em período parcial

De 18 a 55 anos: contrata em período integral

De 55 anos a 99 anos: não contrata

Portanto: Se (idade ≥ 0 && idade ≤ 15) situação = "NC";

Se (idade ≥ 16 && idade ≤ 17) situação = "Parcial";

Se (idade ≥ 18 && idade ≤ 54) situação = "Integral";

Se (idade ≥ 55 && idade ≤ 99) situação = "NC";

Conjuntos de valores a serem testados são:

{-1,0,1};

{15,16,17};

{17,18,19};

{54,55,56};

{98,99,100}.

Lembrando que para a realização dos testes é preciso entender os requisitos funcionais do negócio, para definir exatamente o nível de testes que pretende estabelecer e avaliar a resposta que o software trará.

Autores do capítulo

Referências

CARVALHO, C. E. M. **O que é BPMN?**. <https://pt.linkedin.com/pulse/o-que-%C3%A9-bpmn-carlos-eduardo-msc-mba-pmp-itol-cobit-cisa-edge>, 2017.

CHACON S. e STRAUB B. **Pro Git**. 2.Ed., 2014.

COIMBRA, PMP. **Termo de Abertura do Projeto – PMBOK 4ª Edição**. <https://projetoaseti.com.br/termo-de-abertura-do-projeto-pmbok4-edicao/>, 2012.

EPA DEVELOPERS GUIDANCE. **Agile Framework – SCRUM, Kanban and Extreme Programming**. <https://developer.epa.gov/guide/templates-guides/agile/agile-frameworks/>, 2017.

FORMAGGIO, E. **Gestão de integração em projetos**. <http://miauproject.com.br/blog/gestao-da-integracao-em-projetos/>, 2014.

MACORATTI, J. C. **Conceitos Básicos de modelagem de dados**. <http://www.macoratti.net/cbmd1.htm>, 2013.

MANIFESTO AGIL. **Manifesto para Desenvolvimento Ágil de Software**. <http://agilemanifesto.org/>, 2001.

MIRANDA, W. **Modelagem de Dados**. <http://aprendaplsql.com/modelagem-de-dados/modelagem-de-dados-parte-01/>, 2017.

MONTES, E. **Introdução ao Gerenciamento de Projetos**. <https://escritoriodeprojetos.com.br/o-que-e-um-projeto#!/ccomment>, 2017.

NEVES, M. **Curso de BPMN 2.0**. https://www.youtube.com/watch?v=_KdCwaOp7zE&list=PL7NDvV6PnYOCCaZyza08NI0AgbaTNmT8D, 2015.

POHL, K. e RUPP, C. **Fundamentos da Engenharia de Requisitos**. Pág. 34 a 59, 2012.

PROJECT BUILDER. **Como identificar os stakeholders-chave para seu projeto?**. <https://www.projectbuilder.com.br/blog/como-identificar-os-stakeholders-chave-para-seu-projeto/>, 2017.

PROJECT MANAGEMENT INSTITUTE. **O que é Gerenciamento de Projetos**. <https://brasil.pmi.org/brazil/AboutUs/WhatIsProjectManagement.aspx>, 2018.

RODRIGUES, J. **Modelo Entidade relacionamento (MER) e Diagrama Entidade-Relacionamento (DER)**. <https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>, 2017.

ROGER S. **Engenharia de Software – Uma abordagem profissional**, Pressman - Bruce R. Maxim 8ª edição – 2016.

SOMMERVILLE, IAN. **Engenharia de SOFTWARE**. 9. Ed. São Paulo, 2011.

TESTES DE SOFTWARE. **Tipos de teste de software**. <http://testesdesoftware.com/tipos-de-teste-de-software/>, 2017.

VINÍCIUS, T. **Introdução a modelagem de dados**. <https://www.devmedia.com.br/introducao-a-modelagem-de-dados/24953>, 2016.

YOUTUBE BIZAGI. **Bizagi**. <https://www.youtube.com/channel/UCeVjOJMxih3lCXS3-R26n4Q>, 2009.

JAN/2018

CURSO DE FÉRIAS ANÁLISE DE NEGÓCIOS

COLABORADORES

Vinicius Mussak
mussak@smn.com.br
Coordenador geral

Felipe Pomini
felipe@smn.com.br
Coordenador do curso de análise de negócios

Maria Rita Benate
maria.rita@smn.com.br
Prototipação / Analista de negócio
vs. Garçom de requisitos

Ericson Silva
edoni.smn@gmail.com
Técnicas de levantamento de requisitos

André Assunção
andre.luiz@smn.com.br
Banco de dados

Felipe Araújo
felipe.aparecido@smn.com.br
BPMN

Andrelino Faria
andrelino@smn.com.br
Homologação e testes

Ezequiel Alves
ezequiel@smn.com.br
Introdução a Git

Jhonatan Gonçalves
jhonatan@smn.com.br
Gerenciamento de Projetos

Rafael Borges
rafael.borges@smn.com.br
Metodologias ágeis / Scrum

Wilson Guiraldelli
wilson@smn.com.br
User Stories / Behavior Driven Development