**FUNDAMENTAÇÃO TEÓRICA**

Quando desenvolvemos um software, são várias as tecnologias e recursos existentes. Para escolher uma tecnologia, uma linguagem de programação dentre as disponíveis, exigiu uma análise criteriosa sobre a finalidade do sistema. Neste capitulo serão apresentadas as tecnologias utilizadas ao longo do desenvolvimento do projeto.

* 1. **Gerenciamento de Projetos**

Segundo a NBR ISO 10006 (2000), projeto é um “*processo único, constituído de um grupo de atividades coordenadas e controladas com datas para início e termino para alcançar objetivos conforme requisitos específicos, incluindo limitações de tempo, recurso e custos*”. Esta afirmação concorda que o projeto busca um único resultado, com início e fins bem definidos.

Desta forma, tornasse evidente a importância do planejamento e estruturação do projeto, pois evita desperdício de tempo e recursos assim como assegura o atendimento das expectativas do cliente.

* 1. **Github**

O *Git* nasceu, quando o DVCS chamado *BitKeeper* desfez o contrato com o Linux, foi onde surgiu a ideia de fazer uma própria ferramenta, com o que tinham aprendido usando o *BitKeeper*. Criado em 2005, o sistema foi evoluindo e amadurecendo a ponto de se tornar um sistema de fácil de usar. Extremamente rápido e eficiente com grandes projetos, possuí um branching para desenvolvedores não-lineares.

É um versionador de arquivos, utilizado para gerenciar versões de software onde pode-se implementar novas funcionalidades, e sempre vai haver um histórico das operações, sendo possível retroceder, e todos os usuários podem enviar atualizações, quando enviadas as alterações, a versão anterior fica salva, e o administrador escolhe se aquela atualização pode ou não ser efetuada.

A princípio usa a mesma ideia de repositório local, a diferença é que está hospedado em servidores da internet, quase todas as operações usam recursos e arquivos locais, onde o computado já possua um histórico dos projetos, as operações serão instantâneas, pois não é requisitado ao servidor e sim ao banco de dados local, com o *Git* pode trabalhar off-line realizando commits, e quando se reestabelecer a conexão um simples upload, em outros sistemas isso é quase impossível.

*Git* tem sua integridade, que é *checksum* (valor para a integração de integridade), nenhum dado pode ser alterado ser que o *Git* tenha conhecimento, caso tenha algum arquivo corrompido ou perdido o sistema avisara, mas isso é praticamente impossível de acontecer.

* 1. **Kanban**

Criado pelo *Taiichi Ohno*, que se inspirou nos supermercados americanos, onde quando se esvaziava um produto era reposto somente aquela quantidade, pois não havia muito espaço. Desenvolvido para organizar os processos das empresas, tem o objetivo de controlar a movimentação e produção.

O *Kanban* permite um controle mais detalhado da produção com as informações de *quando, quanto e o que produzir*, o método pode ser aplicado para sistemas de produção repetitiva, segundo o criador o *Kanban* tem que respeitar algumas regras:

“*O processo subsequente deve retirar, no processo precedente, os produtos necessários nas quantidades certas e no tempo correto;*

*O processo precedente deve produzir os seus produtos nas quantidades requisitadas pelo processo subsequente;*

*Nenhum item pode ser produzido ou transportado sem um kanban;*

*Produtos com defeito não devem ser enviados ao processo seguinte;*

*O número de kanbans deve ser minimizado continuamente.”*

O *Kanban* é uma ferramenta de metodologia ágil as vezes usado juntamente com o *Scrum*, que tem o objetivo de aumentar a produtividade.

* 1. **Estrutura Analítica do Projeto (EAP)**

Uma *EAP* (Estrutura Analítica do Projeto) ou em inglês *WBS* (Work Breakdown Structure) é utilizada para evidenciar as entregas do projeto. Ele inclui todo o escopo do projeto, ou seja, o trabalho necessário para terminar o projeto e atender os requisitos das partes interessadas.

Segundo PMBOK 4

*A EAP é uma decomposição hierárquica orientada*

*á entrega do trabalho a ser executado pela equipe*

*para atingir os objetivos do projeto e criar as entregas*

*requisitadas, com cada nível descendente da EAP*

*representando uma definição gradualmente mais detalhada*

*do trabalho do projeto. É finalizada pela determinação*

*de contas de controle (ver PMBOK). Esses identificadoes*

*produzem uma estrutura para sumarização hierárquica*

*de custos, cronograma e informações sobre recursos.*

O primeiro retângulo da EAP, contém o nome do projeto. Os níveis mais a baixo representam “o quê” será entregue. Os níveis somados, representarão o projeto como um todo.

* 1. **Iterativo Incremental**

Incremental é realizado um planejamento, onde a estratégia é desenvolver várias partes em paralelas e integrar quando estão prontas. Seu objetivo é desenvolver o sistema somente com uma integração.

Iterativo é onde partes do sistema, melhorias e revisão são pré-definidos, que funciona com o incremental, pois no incremental as saídas não são pré-definidas, e o teste do usuário não são utilizados para revisão, já no iterativo sua saída é utilizada para modificação e revisão. São gerados vários ciclos, no desenvolvimento por exemplo podemos identificar a fase de análise, projeto, teste e implementação, onde os ciclos são considerados um subconjunto de requisitos, que são desenvolvidos e alocados a um ciclo de desenvolvimento, o próximo ciclo o novo conjunto de requisitos que será desenvolvido, produzira um novo incremento que contém extensões e refinamentos do ciclo anterior, sendo assim o desenvolvimento vai evoluindo, até que o sistema esteja completo.

* 1. **Casos de Uso**

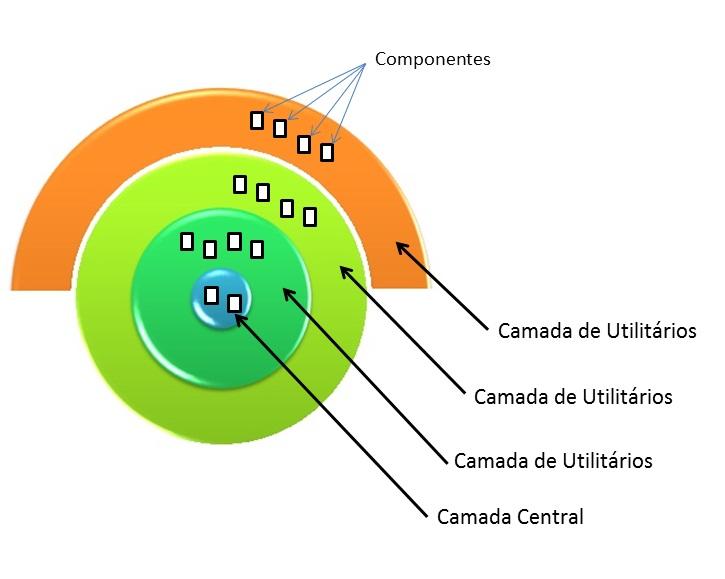
*Caso de uso* é uma técnica de modelagem com a finalidade de descrever os requisitos funcionais do sistema de forma clara e consistente.

Segundo Ivar Jacobson, podemos dizer que um caso de uso é um “*documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo”*

Os Diagramas de *Casos de Uso* são representações dos mesmos e podem ser representados por uma elipse contendo, internamente, o nome do caso de uso. Esta elipse pode estar ligada por um traço a um boneco, onde o boneco indica o ator e o traço indica a interação entre eles.

* 1. **Arquitetura em Três Camadas (3 TIER)**

A estrutura básica de uma arquitetura em camadas estará ilustrada na figura a baixo. Um certo número de camadas diferentes é definido, cada uma realizando operações que se tornam progressivamente mais próximas do conjunto de instruções de máquina. Na camada exterior; os componentes servem as operações da interface do usuário. Na camada mais interna os componentes realizam a interface com o sistema operacional. As camadas intermediarias fornecem serviços utilitários e funções do software de aplicação.



* 1. **Microsoft Visual Studio**

Conjunto de ferramentas utilizado para desenvolver aplicativos web e desktop. Foi desenvolvida pela *Microsoft*, e foi dedicado ao *framework.net* e *Asp.Net* (plataforma web), onde pode ser desenvolvido aplicativos, sites e serviços para esta plataforma.

* 1. **C# (C Sharp)**

*C#* lê- se como “cê chárp” no português e “si sharp” para o inglês. Foi criado pela Microsoft para a plataforma .NET, é uma linguagem orientada a objetos que foi baseado nas linguagens C++ e Java, C# é considerada uma linguagem símbolo de .Net pois, criaram do zero sem preocupação de com o código legado, desenvolvida para rodar em uma plataforma nova e foi criado primeiro o seu copilador. A linguagem foi desenvolvida por vários programadores, mas o principal foi Anders Hejlsberg, que é o criador de Turbo Pascal e Delphi.

Na linguagem não é inclusa nenhuma biblioteca de classes ou funções, a mesma está ligada ao framework.Net onde obtém suas classes e funções de execução. Ao desenvolver na linguem percebesse que é organizado por conjuntos de namespaces, exemplo *System.Collections (estruturas de dados) e System.Drawing (para gráficos)* e suporta encapsulamento, polimorfismo e herança.

C# foi submetida a ECMA, para padronização formal, onde em 2001 foi liberada a especificação ECMA - 334, e em 2003 tornou padrão ISSO (ISO\IEC 23270). A Microsoft continua induzindo mudanças nas linguagens, onde C# faz parte de sua estratégia de software.

* 1. **Designer Patterns**

É um padrão de projeto de software, onde as soluções são reutilizadas em problemas recorrentes, em situação especifica.

O *Designer Patterns* teve início no trabalho de um engenheiro civil, chamado Chistopher Alexander, onde estabelecerão as características que o padrão deveria ter como Nome, Exemplo, Contexto, Problema e Solução. Chistopher, descreveu como solucionar os problemas, que obteve com a sua experiência.

No dano de 1995, um grupo de programadores ficou conhecido a *gangue dos quatros*, ao publicar o livro *"Design Patterns: Elements of Reusable Object-Oriented Software*". O livro apresentava 23 padrões baseados em suas próprias experiências, onde organizarem em três famílias:

***°*** *Padrões de criação: relacionados à criação de objetos;*

***°*** *Padrões estruturais: tratam das associações entre classes e objetos;*

***°*** *Padrões comportamentais: tratam das interações e divisões de*

*º Responsabilidades entre as classes ou objetos.*

* 1. **Singleton**

Considerado um padrão criacional, o *Singleton* garante que apenas uma instância do objeto seja instância em todo o projeto, além de ter um ponto de acesso global a essa instância.

Outra funcionalidade é quando precisamos de uma classe de *Log* para todo o sistema, sem precisar instanciar a classe, utilizando os recursos do sistema, pois não é preciso chamar o construtor para a criação da instância, onde seu acesso é restrito (método privado).

Para criar a instancia da classe usa o método público, onde o mesmo vai fazer o controle da instancia, permitindo que ela só possa ser feita uma vez.

* 1. **Windows Presentation Foundation (WPF)**

Tecnologia criada pela Microsoft para desenvolver aplicativos e interfaces, não depende de plataforma, sendo assim o código-fonte funciona para web e desktop. O programa que utiliza o Wpf tem duas partes, um que utiliza o arquivo XML que contém uma característica chamada XAML (eXtended Aplication Markup Language, pronuncia-se “zémel”) e um código .Net.

Suas principais características são:

º Flexibilidade na interface, independente do código;

º Utilizada em desktop e web;

º Utiliza XAML e código. Net

º Melhora a interface do hardware do usuário através do sistema operacional

* 1. **Model View Controller (MVC)**

O *Mvc* é composto pelo modelo, visão e controlador, sendo que no começo servia para mapear o processo de entrada, processamento e saída, assim era possível ter os três processos mapeados.

*Controller* é responsável por controlar as entradas do mouse e teclado, mapear as ações e enviar para o Model ou para a *View*, onde pode ser realizada a alteração, o Model também pode gerenciar vários dados, sendo assim o mesmo responde o estado e como mudar o estado. A *View* fica responsável para apresentar ao usuário as informações que são geradas por gráficos e textos, sendo assim ela apenas recebe as informações do *Model* e *Controller* e exibe.

* 1. **Object-Relational Mapping (ORM)**

O *Orm* é um *framework* ou conjunto de classes que facilita o do desenvolvedor, pois não precisa escrever códigos a todo o momento para conexão com bancos, *querys* de *Sql*, mesmo assim as características da orientação ao objeto são preservadas.

Facilita a manutenção, o código fica mais limpo e se obtém uma ótima performance da padronização.

* 1. **.Net Frameworks**

O *.Net Frameworks* é considerado um ambiente de tempo de execução, gerencia aplicativos que utilizam o *.Net Framework.*

Consistem em uma linguagem comum, é fornecido gerenciamento de memória, administra uma imensa biblioteca de classes, o que permite os programadores aproveitar ao máximo o código.

Possui dois componentes principais, *Common Language Runtime (CLR)* - manipula aplicativos, que são executados na biblioteca de classe. A biblioteca fornecida é de código testado, os desenvolvedores podem reutilizar, e chamar a partir de seus próprios aplicativos.

* 1. **Sql Server Express 12**

Programa de gerenciador de banco de dados, é da *Microsoft* que foi criado a partir de uma parceria com a *Sybase* em 1988, onde inicialmente era um complementar do Windows NT, com o decorrer do tempo, foi sendo aperfeiçoado e é um sistema robusto e usado por diversas corporativas e comporta diversas plataformas. O *Sql* (Structured Query Language) está sendo um dos mais usados do mundo, competindo fortemente com o *Mysql* e *Oracle*.

O backup exige menos do I\O (entrada\saída), e diminui os custos para manter online, existe a computação de bloqueio que o usuário pode aumentar a profundidade de hierarquia. Umas das principais características é que a linguagem trabalha com conjuntos de registros de dados, enquanto as outros, foram feitas para tratar um registro de cada vez, pois o *Sql* não é procedural sendo assim não precisa detalhar a tarefa que precisa, a linguagem descreve o que o usuário final deseja.

A linguem se divide em duas: DDL (linguagem de Definição de Dados) - Usa instruções para descrever o esquema das tabelas, e a DML (linguagem de Manipulação de Dados) - Usa instruções para manipular os dados.

* 1. **Testes**