**4 Fundamentação Teórica**

A Web 1.0, era uma plataforma, onde usuários, no caso espectadores, podiam apenas navegar entre as páginas e visualizar seu conteúdo; não sendo permitido nenhum tipo de interação com o serviço, como por exemplo, reeditar ou alterar algum conteúdo exposto na página.

        Além disso, a grande maioria de serviços prestados na Internet, eram liberados mediante a pagamentos, ou seja, para usufruir de algum conteúdo, era necessário realizar um pagamento e uma licença única e restrita àquele usuário era gerada para que o mesmo tivesse acesso a tais serviços.

         Apesar do vasto conteúdo de informações transmitidas para o usuário, a Internet era aberta para todos, sem  nenhum dono para controlar os dados acessados.

Eis que surge a segunda fase da Internet que foi o marco da evolução da Internet, onde Tim O’Reilly no ano de 2005 diz as seguintes considerações:

*A web 2.0 é a mudança para uma Internet como plataforma, e um entendimento     das regras para obter sucesso nesta nova plataforma. Entre outras, a regra mais importante é desenvolver aplicativos que aproveitem os efeitos de rede para se tornarem melhores quanto mais são usados pelas pessoas, aproveitando a inteligência coletiva. (REILLY, 2005). [5]*

      Nesta mudança, obteve-se um enorme avanço em relação à interação do usuário, publicações de conteúdo, edições, criações de páginas online sem nenhuma dificuldade em nível de programação e também pode-se citar que o utilizador disponibilizava servidores gratuitamente.

Blogs, PodCast, Hi5, De.li.cio.us são alguns termos que podem ser citados como exemplo da grande variedades de serviços que estão hoje disponíveis graças ao avanço que a Web 2.0 propôs. [6]

       Outras características importantes, são as informações que se atualizam constantemente a fim de diminuir e obter correções para maior comodidade do utilizador, interações com pessoas do mundo inteiro, nisso, incluímos aplicações como Messenger, Googletalk, Skype e Voip, algumas ferramentas de escrita online como Google Docs e Podcast e o avanço do acesso a vídeos que cresceu muito nos últimos anos com o surgimento do Youtube, GoogleVideos e YahooVideos.

Sendo assim, a demanda de programação para a web foi crescendo cada vez mais, e problemas como prazos de entrega que não eram respeitados por vários fatores determinantes e até mesmo o não entendimento por parte da parte de programação que ficava um pouco confusa na hora de ser feita a interpretação dos textos da documentação, surgiu um novo modo de ser feita a organização dessas documentações, algo que poria a prática na implementação do software em algo muito mais eficiente e com bastante clareza que foi o manifesto agile. (colocar definição no glossário)

Antes do manifesto, vários fatores agravavam o não cumprimento de metas e prazos de entrega dos softwares aos clientes, tais como poucas reuniões com os clientes, onde ficavam faltando informações sobre o projeto, advogados que brigavam por causa de contratos, documentação muito extensa que acarretava o desentendimento dos desenvolvedores na hora de analisar a documentação

O manifesto agile, que significa desenvolver com agilidade teve como precursor os próprios desenvolvedores ao tentar dominar um pouco o processo produtivo nas empresas em que trabalhavam, não aceitando as regras impostas de algum gerente que muitas vezes não sabiam como funcionava o processo da fábrica de software, então eles começaram a mudar a maneira como trabalhavam, fazendo o que achavam que seriam melhor para os projetos em que trabalhavam e a partir dessas ideias, outros desenvolvedores se juntaram ao manifesto.

O manifesto agile tem como seus valores, quatro itens. Sendo eles:

* Os indivíduos e suas interações acima de processos e ferramentas.
* O software em funcionamento acima da documentação abrangente.
* A colaboração dos clientes acima da negociação de contratos.
* A capacidade de resposta a mudanças acima de se seguir um novo plano.

A expressão “Metodologias Ágeis” tornou-se conhecida em 2001, quando especialistas em processos de desenvolvimento de software representando entre outros, os métodos Scrum e Extreme Programming (XP), foram estabelecidos princípios e características comuns destes métodos. Assim foi criada a “Aliança Ágil” e efetuou-se o estabelecimento do “Manifesto Ágil”.

Extreme Programming (XP) é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente, sendo diferente dos métodos de desenvolvimento comum o constante feedback, a obordagem incremental e o encorajamento da comunicação entre pessoas.

Muitas de suas regras causam estrenheza em seu primeiro contato ou até mesmo não fazem sentido se aplicadas isoladamente, porém é a porça de seu conjunto que sustenta o sucesso da XP, trazendo a verdadeira revolução no desenvolvimento de software, sendo como seu objetivo dar agilidade ao processo de desenvolvimento e buscando sempre garantir a satisfação do cliente. Suas práticas são conduzidas por quatro princípios básicos, sendo eles:

1. **Princípio de comunicação** – busca manter o relacionamento com o cliente sempre da melhor forma possível, com reuniões práticas e esclarecedoras, prezando pelo contato pessoal ao invés de outros meios de comunicação.
2. **Princípio da simplicidade** - busca-se alcançar o objetivo do projeto de software com o menor número possível de classes e métodos, também está em sua caracterísitca a procura da implementação de requisitos atuais, evitando adicionar funcionalidades que podem ser importante apenas no futuro. A aposta do XP é fazer algo simples hoje do que implementar algo complicado hoje que talvez não venha a ser usado.
3. **Princípio do feedback** – o desenvolvedor terá constantes feedback do código que está sendo construiído se informando num geral sobre todo o projeto e recebendo também feedback do cliente para qual o projeto é destinado. A informação do código é conseguida através de constantes testes em que indicam os erros que o código possui, tanto erros pessoais quanto erros de código integrado.
4. **Princípio da coragem** – Sabe-se que nem todas as pessoas tem facilidade de comunicação e possuim bom relacionamento interpessoal, sendo um método voltado para a simplicidade, a equipe pode experimentar a buscar novas soluções, além disso, é preciso coragem para obter e cobrar constantes feedback do cliente. [referência]

As práticas do modo XP começam pelo planejamento, sendo definido o que é, e o que não é necessário ser feito no projeto, baseando-se em requisitos atuais, deixando que requisitos futuros sejam implementados somente quando haja a necessidade deles no projeto. A entrega do software acontece conforme os requisitos vão sendo implementados, a cada requisito há uma atualização do software, cada versão entregue ao cliente deve conter o menor tamanho possível, contendo os requisitos de maior valor para o negócio.

A descrição do projeto fica por conta de metáforas, sendo assim, a descrição não conta com termos técnicos, o que possibilita um maior entendimento para o cliente, sendo muito importante a participação do cliente em todo o processo de desenvolvimento para sanar dúvidas de requisitos, evitando atrasos ou até mesmo construções erradas.

A Extreme Programming (XP) prioriza a validação do projeto durante todo o processo de desenvolvimento. Os desenvolvedores implementam o software criando primeiramente os testes, sendo a implementação de código feito em duplas, ou seja, dois desenvolvedores trabalham em um único computador, procurando identificar erros sintáticos e semânticos, pensando em conjunto em como melhorar o código que está sendo implementado.

Muito importante também é a refatoração do código, tornando um código lapidado, limpo, para que não haja problema na hora que for feita a leitura deste código que perterce a todos os membros da equipe. Isso significa que qualquer pessoa pode perceber alguma necessidade no código, pode fazê-lo mesmo sem ter sido a pessoa propriamente dita quem desenvolveu aquele código, mas desde que faça os testes necessários e não prejudique as funcionalidades atuais. A grande vantagem de todos ter acesso a todo o código, é que se algum membro abandona o projeto, outro membro não terá grandes dificuldades ao se encarregar das tarefas que foram deixadas pelo outro membro do projeto, pois todos conhecem todas as partes do software mesmo que não seja de forma detalhada.

A XP também assume, que não se deve trabalhar mais que 40 horas semanais. Caso seja necessário trabalhar mais que 40 horas, significa que existe um problema sério no projeto que deve ser resolvido não com o aumento de horas trabalhadas, mas sim como melhorar o planejamento. [referência]

Além da evolução em como a internet era vista pelos navegadores, após a evolução da web 1.0 para web 2.0 como citado acima, o desenvolvimento também sofreu uma evolução, a maioria dos sites antigos eram todos escritos em HTML.

HTML foi projetado para definir a estrutura de um documento da web, onde o código <p> é um elemento estrutural que se refere à "parágrafo", <LI> também é um elemento estrutural que se refere à "listar item" enquanto o HTML expandia, mais elementos foram adicionados. Estes elementos, definem como o texto que será exibido, será formatado.

Em outras palavras, forma e conteúdo se tornaram inseparáveis em HTML, porém o XML foi projetado para fazer exatamente isso onde <title> não define a forma e sim o conteúdo, o mesmo com <link> e <description>, assim os dados podem ser exportados livres de restrições de formatação.

Com a forma separada, usuários não precisavam saber linhas de código para publicar informações na internet, nascem então os blogs, onde a programação já era toda definida pelo servidor em que hospedava os blogs, o usuário a partir de sua conta, somente tinha que enviar informações para ser exibida.

Mesmo sendo um avanço tecnológico grande, para se construir um site, o programador dedicava muito tempo do seu tempo para digitar inúmeras linhas de comando, as vezes para se construir pouca coisa, necessitava de um grande número de linhas.

Com o aprimoramento das tecnologias, novas linguagens de programação para web foram surgindo, e que satisfazem essa questão da necessidade do programador digitar várias linhas de comando. Exemplo dessas tecnologia de desenvolvimento ágil é a linguagem Ruby que foi escolhida por ser uma linguagem dinâmica e atual, com constante crescimento no mercado de trabalho, segue abaixo suas definições e ferramentas que podem ser usadas juntamente com a linguagem.

Ruby é uma linguagem de programação interpretada multiparadigma, de tipagem dinâmica e forte, com gerenciamento de memória automático, originalmente planejada e desenvolvida no Japão em 1995 por Yukihiro “Matz” Matsumoto, para ser usada como linguagem de script. Matz queria uma linguagem de script que fosse mais poderosa do que Perl e mais orientada a objetos do que Python.

A linguagem também suporta programação funcional, orientada a objetos, imperativa e reflexiva. Foi inspirada principalmente por Python, Perl, Smalltalk, Eiffel, Ada e Lisp, sendo muito similar em vários aspectos a Python. Atualmente, Ruby é a 11ª (décima primeira) linguagem de programação mais popular do mundo, de acordo com o Índice Tiobe [usar referencia do tcc]

Ruby on Rails é seu framework, foi escrito por David Heinemeier Hansson em Ruby, organizado em MVC (Model-View-Controller). Lançado para o público em 2004, sua característica é a facilidade na hora de programar, tornando códigos menores o que leva o programador programar mais em menos tempo.

Este framework surgiu para trazer uma forma mais dinâmica para o desenvolvimento de sites orientado a banco de dados, focando na agilidade para aumentar a produtividade em uma quantidade de linhas menores.

Esta é uma plataforma com opinião, ela impõe que tem um modo certo de se fazer as coisas e foi desenvolvido para encorajar o programador a seguir este modo.

Rails segue algumas filosofias, e uma delas é denominada DRY (Don't Repeat Yourself) ela diz que não é bom que não é bom escrever o mesmo código mais de uma vez.

Em uma plataforma de arquitetura MVC (Model-View-Controller) o ROR utiliza um recurso chamado Scaffolding que gera automaticamente controllers e views a partir de um certo modelo mostrado para o programador. RoR também provê abstração de SGBDs (Sistema de Gerenciamento de Banco de Dados), o usuário usa apenas uma sintaxe para MySQL, PostgreSQL, SQL Server, Oracle, Firebird, entre outros

O Rails faz suposições sobre o que você quer fazer e como você estará fazendo isto, em vez de deixá-lo mudar cada minúscula coisa através de intermináveis arquivos de configuração. [9]

Esta ferramenta poderosa de programação para Web pode ser rodado em Windows, Linux e Mac OS e a permissão para o uso do software é gratuitamente, podendo modificar sem restrição, com direitos a usar, copiar, distribuir e até vender.

A linguagem disponibiliza o uso do coffescript que é uma pequena linguagem de script que compila para JavaScript para desenvolvimento de websites, é apenas JavaScript, compilando de um-para-um para o js (JavaScript) equivalente.. CoffeScript tenta mostrar as boas partes do JavaScript em um modo simplificado. Esta linguagem converte os scripts .coffe em scripts .js (JavaScript). [10] Segue abaixo exemplos de sintaxes escrita em JavaScript e a mesma sintaxe escrita em CoffeScript.



Além do CoffeScript, o uso do HAML também é utilizado. A idéia de HAML é somente a estética da sintaxe, não funcionalidades. É uma sintaxe simplificada do HTML, renderizado para HTML que não necessita o fechamento de tags. Segue abaixo exemplo de sintaxe escrita em HTML e a mesma escrita em HAML.



Quando o assunto diz respeito a CSS, a tecnologia utilizadada chama-se Sass.que é uma extensão para o CSS3, desenvolvida [Nathan Weizenbaum](http://en.wikipedia.org/w/index.php?title=Nathan_Weizenbaum&action=edit&redlink=1), adicionando regras aninhadas, variáveis, mixins, herança de seletor. É traduzida para o formato CSS padrão. [11]

