

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE SÃO PAULO**

Campus São João da Boa Vista

Trabalho Final de Curso

4º ano – Curso Técnico em Informática

Prof. Breno Lisi Romano

**Criação de uma website e introdução no desenvolvimento
front-end e back-end**

Aluno: Alexandre de Freitas Filho

Prontuário: 1420372

São João da Boa Vista – SP

2017

Resumo

Atualmente as fontes de informação estão cada vez mais concentradas na internet. Este projeto de conclusão de curso teve como objetivo oferecer a comunidade uma metodologia de desenvolvimento web, e também princípios básicos de desenvolvimento *front-end* e *back-end*. Por meio dos documentos de visão, foi possível deixar uma boa documentação para consultas futuras e adaptação para cada caso específico. Através de códigos *PHP*, *JavaScript* e o padrão *MVC* foi possível deixar uma base introdutória simples para possíveis desenvolvedores. O resultado foi um site que permite a comunicação entre estabelecimentos e consumidores, com um design amigável e poucos passos para execução das operações.

Palavras-chave: *Back-end*; *PHP*; *Front-end*; *Website*; Reclame São João.

Lista de figuras

Figura 1 - Arquitetura cliente/servidor [3].	9
Figura 2 - Fases do modelo em cascata [6].	10
Figura 3 - Camadas, exemplo de padrão estrutural [8].	11
Figura 4 - SOA, exemplo de padrão de sistemas distribuídos [9].	12
Figura 5 - MVC, exemplo de padrão de sistemas interativos [10].	13
Figura 6 - Micro Kernel, exemplo de padrão de sistemas adaptáveis [11].	14
Figura 7 - Comparativo de PHP frameworks [13].	15
Figura 8 - Exemplo de um projeto sendo organizado no Kanban [14]	16
Figura 9 - Exemplo de um projeto sendo organizado no Redmine [16]	17
Figura 10 - Mostra como é o funcionamento estrutural do SVN [18].	18
Figura 11 - Projeto sendo feito utilizando o Notepad++	18
Figura 12 - Sistema dividido em módulos	19
Figura 13 – Página Inicial	20
Figura 14 – Estrutura inicial de uma página html [20].	21
Figura 15 - Código dos primeiros protótipos	22
Figura 16 - Página de reclamações	23
Figura 17 - Código utilizado para criação do alerts e popup	24
Figura 18 - Estrelas da avaliacao	24
Figura 19 - Código captura o voto do consumidor conforme o clique	25
Figura 20 - Banco de dados	26
Figura 21 - Código do banco de dados para tabela consideracao_final	26
Figura 22 - Tratamento de dados do banco da Figura 20 e 21	27
Figura 23 - Página gerenciar reclamação responsável por exibir as reclamações	28
Figura 24 – Comparação de layout fixo e flexível	29
Figura 25 - Uso de CSS para imagens	29
Figura 26 – Código do css adaptativo	30

Sumário

1	Introdução.....	5
1.1	O Projeto	5
1.2	Justificativa	5
1.3	Objetivos	6
1.4	Metodologia	6
1.5	Referencial teórico	6
1.5.1	Etapas de desenvolvimento de um site	6
1.5.2	Desenvolvimento Back-end e Front-end.....	7
1.6	Organização do trabalho	7
2	O Projeto.....	8
2.1	Tecnologia Utilizada.....	8
2.2	Estrutura de funcionamento	8
2.3	Segurança	9
2.4	Processos de software	10
2.5	Arquitetura de Software	11
2.6	Framework	14
2.7	Ferramentas auxiliares	16
2.7.1	KANBAN.....	16
2.7.2	REDMINE	16
2.7.3	SVN.....	17
2.7.4	NOTEPAD++.....	18
2.8	Organização do desenvolvimento	19
3	Desenvolvimento Back-end e Front-end	20
3.1	Criação e Formação das páginas com HTML/CSS	20
3.2	Comportamento com Java Script	23
3.3	Linguagem SQL.....	25
3.4	Estrutura MVC.....	26
3.5	Linguagem PHP	27
3.6	Responsivo.....	28
4	Conclusão e Recomendações	31
5	Referências Bibliográficas.....	32

1 Introdução

Quando pensamos em contratar profissionais para desenvolver um site, loja virtual, portal ou blog, não paramos para perceber que antes do site estar finalizado e publicado, ele passará por diversas etapas. As etapas de um projeto não existem por acaso elas servem para que o produto seja finalizado com o mínimo de problemas possíveis e uma alta satisfação do cliente.

As etapas de um projeto podem variar de uma empresa para outra. Contudo geralmente elas são divididas em: planejamento, criação, estruturação e desenvolvimento. Nesse trabalho será evidenciado as etapas de estruturação e desenvolvimento, tendo como base o projeto Reclame São João.

A etapa de estruturação é responsável pelos estudos e as escolhas de um site. Enquanto a etapa de desenvolvimento consiste na execução de linguagens *front-end* e *back-end*. A linguagem front-end é responsável pela interface gráfica do site, já a linguagem *back-end* lida com a estrutura lógica, ou seja, é aquela que trabalha no lado servidor [1].

1.1 O Projeto

O projeto Reclame São João se iniciou em uma conversa entre o professor Breno Lisi Romano (graduado em Engenharia de Computação) e a prefeitura de São João da Boa Vista, em que a prefeitura expôs um problema de comunicação entre os consumidores e os estabelecimentos do município. Tendo conhecimento do problema os alunos do quarto ano do curso técnico de informática integrado ao ensino médio, do Instituto Federal de Ciência, Tecnologia e Educação – Campus São João da Boa Vista, desenvolveram uma website para facilitar a comunicação entre ambos.

1.2 Justificativa

No Brasil segundo a pesquisa TIC Domicílios 2016, realizada pelo Centro de Estudos sobre as Tecnologias da Informação e da Comunicação (Cetic.br), a internet está presente em 54% dos domicílios [2].

Diante desse contexto em que a grande fonte do conhecimento dos jovens está cada vez mais concentrada na internet, o compartilhamento de conhecimento estrutura em estudos científicos se torna algo essencial para o desenvolvimento tecnológico do país e o aprimoramento dos jovens em diversos setores.

Tendo em vista está grande demanda por conhecimento através da internet, viu-se necessário um meio de compartilhamento do projeto, esse que se baseia em estudos teóricos e experiências práticas das últimas quatro turmas do Instituto Federal - Campus São João da Boa Vista na disciplina de Prática e Desenvolvimento de Sistemas. Para isso foi feito esse relatório com o propósito de compartilhar o conhecimento adquirido ao longo do projeto e além disso aprofundar a experiência do autor na área.

1.3 Objetivos

Este trabalho tem como objetivo oferecer referência a desenvolvedores web de como seria o desenvolvimento de uma website em uma grande empresa. Para isso é transcorrido o projeto Reclame São João que tem todas as dimensões e parâmetros de um projeto como este.

Pretende-se também passar informações necessárias para a utilização do desenvolvimento *Back-end* e *Front-end* em aplicações em geral.

1.4 Metodologia

Os materiais que serão utilizados para a realização desse trabalho são os documentos de requisito e visão do projeto Reclame São João, e os protótipos das páginas de desenvolvimento do módulo de Reclamações, além de projetos similares ao mesmo e artigos científicos, no qual possibilitarão uma maior explicação sobre o tema delimitado.

Como procedimentos para realização deste trabalho, foram feitos estudo científicos sobre etapas de desenvolvimento de um software e aplicações práticas dos códigos *Back-end* e *Front-end* nas páginas de reclamações do portal.

1.5 Referencial teórico

1.5.1 Etapas de desenvolvimento de um site

Etapas de desenvolvimento de um site pode ser entendido como seu ciclo de vida que é um processo que passa desde seu surgimento até o momento no qual ele não será mais utilizado (COSTA, 2000).

De acordo com LOPES (2006), o ciclo de vida serve para diminuir os riscos do projeto, busca entregar o serviço no menor tempo e custo possível. Portanto seguir essas etapas é importante para equilibrar esforços em preço, produto, distribuição e promoção de modo a melhorar o relacionamento cliente-fornecedor.

1.5.2 Desenvolvimento Back-end e Front-end

O desenvolvimento web engloba todas as atividades necessárias para o desenvolvimento de um site. Para PRESSMAN (1995) ele é dividido basicamente em dois segmentos o desenvolvimento front-end e o back-end.

Enquanto o primeiro trabalha na interface gráfica o segundo trabalha no servidor. Quando falamos em interface gráfica estamos nos referindo a aplicação que permite ao usuário enviar comandos (clique em links de um menu, digitar um termo de busca) ou seja, a parte visual de um software.

Na internet existem o cliente-side e o server-side, os dois trabalham juntos, o cliente faz as requisições por meio de navegadores enquanto o servidor envia as respostas das requisições.

Segundo Mendonça (2013), “o desenvolvimento web é uma área que está em constante evolução, para isso os desenvolvedores precisam se manter atualizados sobre as tecnologias e procurar a melhor forma de desenvolver, tanto no back-end como no front-end”.

1.6 Organização do trabalho

Visando a complexidade do tema e a necessidade de uma abordagem sobre o assunto, o presente trabalho foi dividido em cinco capítulos, estruturados da seguinte forma:

O primeiro capítulo é introdutório. Nele são explanadas algumas considerações iniciais, os objetivos a serem alcançados, a justificativa da pesquisa, a metodologia utilizada e o referencial teórico.

O segundo capítulo fala do projeto em si, o que é, como foi feito, de que maneira, utilizando de quais linguagens. Nele é definida a tecnologia utilizada, a estrutura de funcionamento, a segurança por trás da *website*, a organização do desenvolvimento, os processos de software, a arquitetura de software, o estudo sobre *framework* e as ferramentas auxiliares.

O terceiro capítulo contempla a parte prática do trabalho. Nele é apresentado o desenvolvimento *Back-end* e *Front-end* das páginas do módulo de Reclamações do projeto Reclame São João.

No quarto capítulo são apresentadas as considerações finais e algumas recomendações para realização de trabalhos futuros acerca do tema.

2 O Projeto

Como dito anteriormente o projeto se trata de um portal de comunicação entre cliente e estabelecimento, esse portal demandou no início alguns estudos e escolhas.

Abaixo são apresentadas algumas informações importantes estabelecidas no primeiro *Sprint* do projeto.

2.1 Tecnologia Utilizada

Em qualquer projeto seja com pretensões comerciais ou não, é de suma importância a escolha da tecnologia empregada para o desenvolvimento do projeto. Por esse projeto não ser financiado pela prefeitura, foi necessário a avaliação dos custos agregados à compra e manutenção de softwares nele utilizados. Com essa avaliação, tomou-se a decisão de basear o desenvolvimento em tecnologias sem custo comercial nenhum, as denominadas “livres”. Portanto foi definido que o site seria composto essencialmente por três linguagens de programação: o *PHP* (um acrônimo recursivo para "PHP: Hypertext Preprocessor"), uma linguagem de script de código fonte livre e de uso geral, muito utilizada e especialmente criada para o desenvolvimento de aplicações *Web* (Php.net, 2005); o *SQL* uma linguagem padrão, especificamente concebida para permitir que as pessoas criem Bancos de Dados, adicionem novos dados a essas bases, manipulem os dados, e recuperem partes selecionadas dos dados; e a linguagem de marcação *HTML* junto com o *CSS* que é uma folha de estilo em cascata, permitindo a criação de toda parte visual do projeto.

2.2 Estrutura de funcionamento

Foi escolhido para esse site uma estrutura de funcionamento baseada no conceito de arquitetura cliente/servidor. Nessa estrutura, o banco de dados é armazenado em um servidor, que tem como papel compartilhar essas informações para os computadores, *desktops*, *laptops* ou celulares da população conforme solicitado. Dessa forma o site tem uma maior integridade de dados, pois todos os usuários acessam as mesmas informações. Essa arquitetura está representada na figura abaixo:

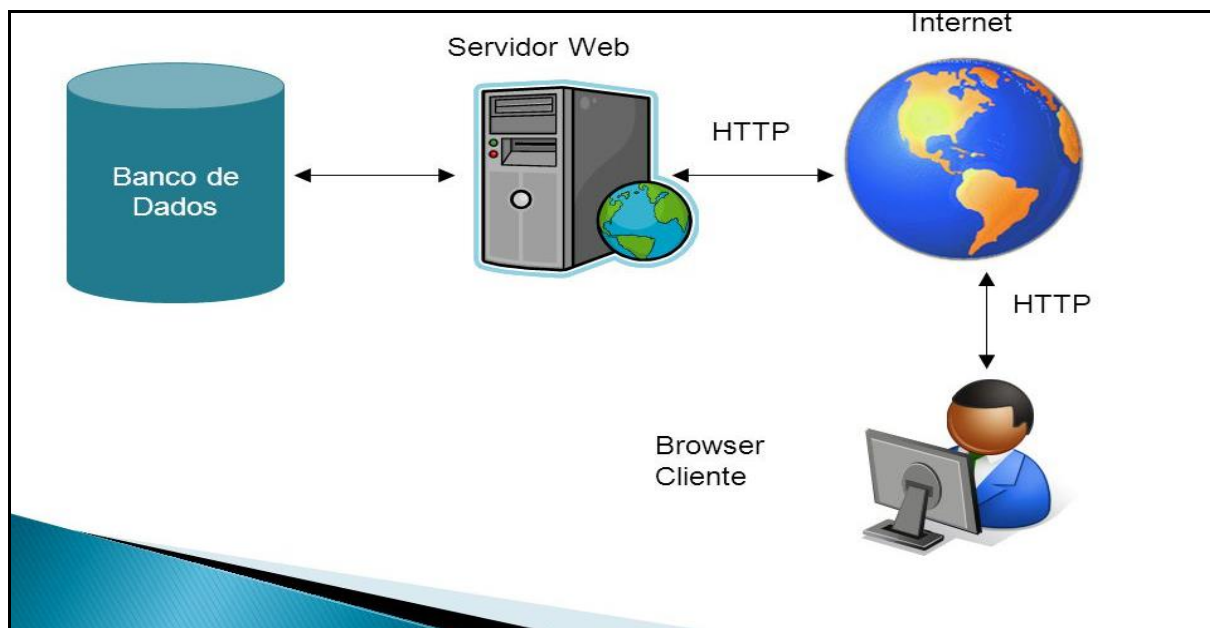


Figura 1 - Arquitetura cliente/servidor [3].

Utilizando isso, portanto é necessário apenas uma instalação do software em um servidor dentro da prefeitura ou em algum provedor de serviços de hospedagem. Dessa maneira o acesso aos dados pode ocorrer através de qualquer dispositivo conectado à internet ao redor do planeta. Além disso é necessário um microcomputador onde será instalado um sistema operacional, preferencialmente o Microsoft Windows devido a facilidade em lidar com a maioria das ferramentas livres atualmente, configurado para funcionar com o servidor de aplicativo e dados. Essa mesma máquina deverá conter um aplicativo servidor *Web*, que irá permitir a interação do usuário através do seu navegador com a *website* [4].

2.3 Segurança

Nesse quesito as especificações foram para o controle de acesso através do *login* do usuário, tendo quatro níveis de acesso: o usuário não logado, o consumidor logado, o estabelecimento logado e o administrador, que seriam protegidos através de um sistema de usuário e senha com criptografia, além de que seria preciso um sigilo no armazenamento de dados, sigilo na exibição dos dados, controle da imagem e a fins do usuário cadastrado.

Para isso seria necessário para cada sessão do site o usuário informar seu *e-mail* de usuário único e intransferível e também sua senha, que serão validados pelo sistema e, caso estejam corretos, será liberado o acesso apenas ao conteúdo adequado a sua categoria. Para se proteger de usuários “fantasmas” o sistema exige que a validação seja confirmada no *e-mail* logo após o cadastro de um novo usuário.

2.4 Processos de software

Atualmente existem diversos modelos de processo de software, portanto foi feita uma análise e baseado nos projetos dos anos anteriores, foi seguido o modelo sequencial linear:

- Modelo sequencial linear ou modelo em cascata:

Este modelo foi proposto por Royce em 1970 e tem como principal característica a sequência linear dos passos: sugere que o progresso de uma fase para a próxima se dá de uma forma puramente sequencial. O *software*, portanto, é feito ao longo de todo esse processo e entregue no final deste. O autor sugere o desenvolvimento em um modelo iterativo, com *feedback* de cada fase influenciando as próximas, de modo similar a muitos métodos amplamente utilizados hoje. Críticas ao modelo Cascata sugerem a inadequação deste a processos reais; e também criticam sua falta de iteração e o fato de ser pouco flexível. Além disso, o modelo Cascata não leva em consideração questões modernas importantes ao desenvolvimento: prototipação, aquisição de software e alterações constantes nos requisitos, por exemplo [5].

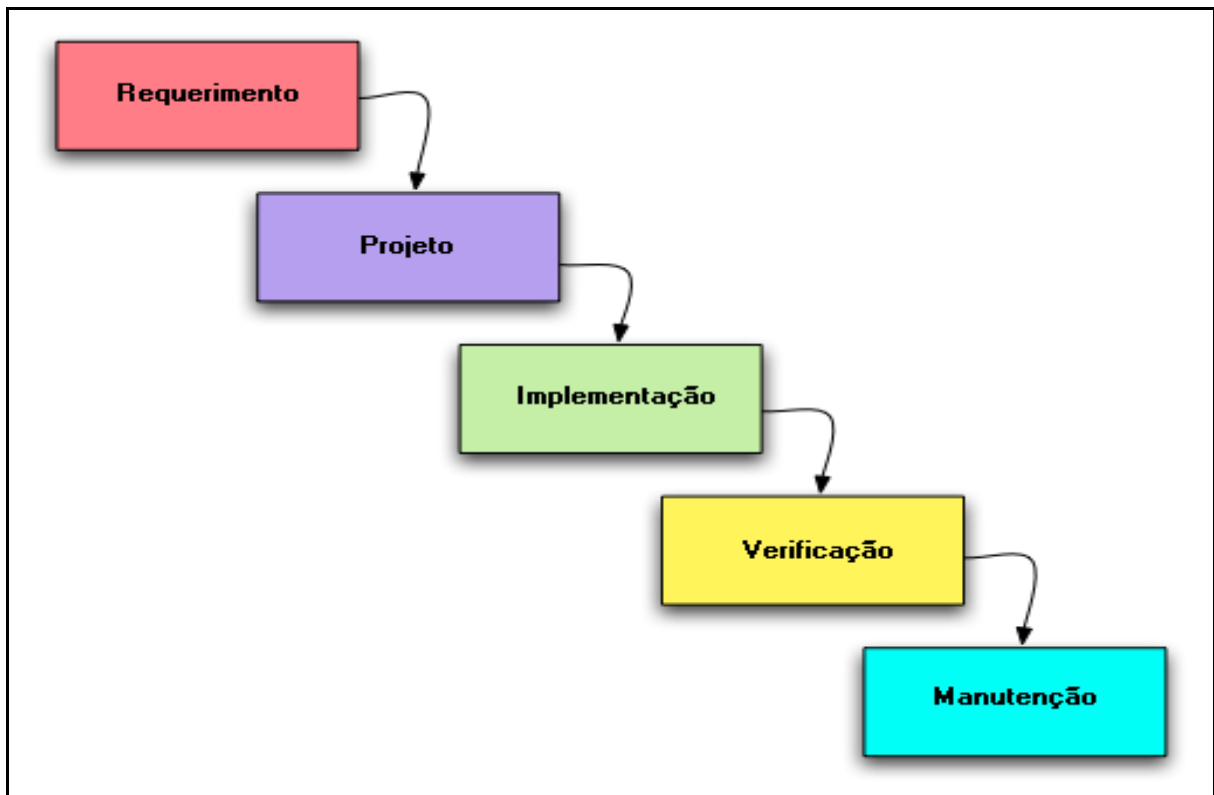


Figura 2 - Fases do modelo em cascata [6].

2.5 Arquitetura de Software

Escolher uma arquitetura de um *software* é mais um passo importante para o sucesso do projeto, sendo ela a organização fundamental de um sistema, compreendida pelos seus componentes, seus relacionamentos entre si, seus relacionamentos com o ambiente e os princípios que guiam seu desenho e evolução; é importante defini-la segundo as necessidades do projeto. Para facilitar a definição e escolha da arquitetura, alguns padrões foram criados. Os padrões definem um conjunto de regras de projeto que identificam tipos de componentes, conectores e restrições existentes para a sua composição, os quais podem ser usados para compor uma família de sistemas e subsistemas [7]. Eles podem ser divididos em quatro categorias:

Estrutural

- Uma opção da categoria estrutural é o padrão de Camadas, em que os componentes ficam nas camadas que controlam a interação e esses componentes se comunicam as camadas vizinhas.

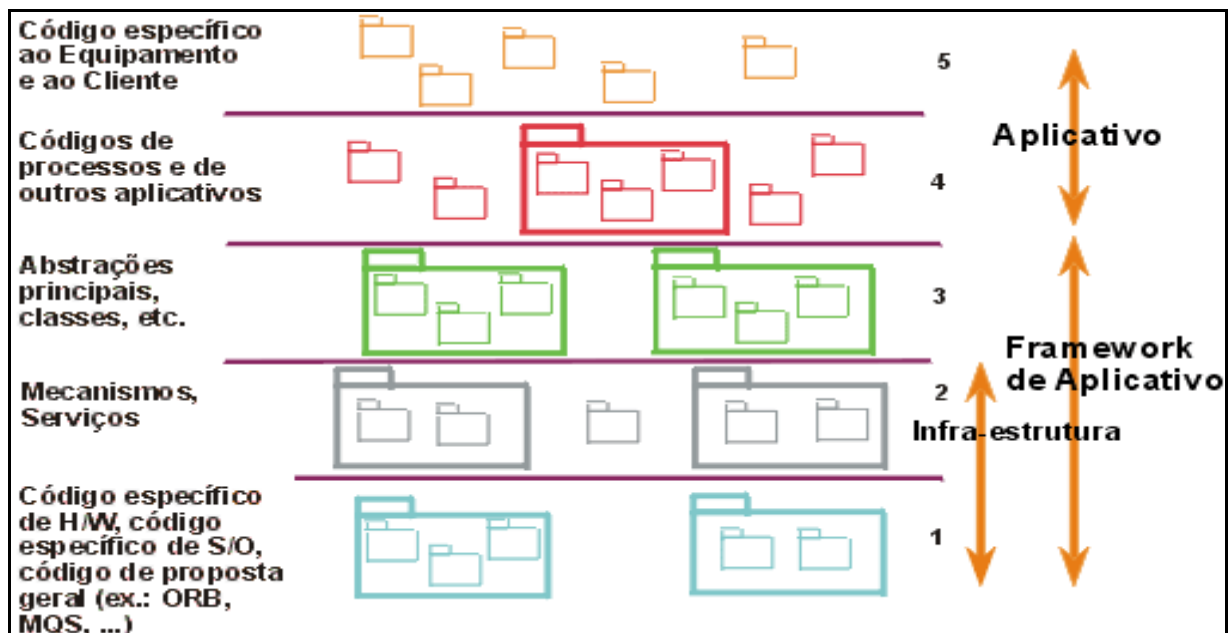


Figura 3 - Camadas, exemplo de padrão estrutural [8].

Sistemas Distribuídos

- Um exemplo é o padrão Broker que permite estruturar aplicações distribuídas através de componentes desacoplados, que interagem via invocações remotas, coordenando toda a

comunicação, de forma a fazer o encaminhamento de requisições, coordenando os resultados e as exceções.

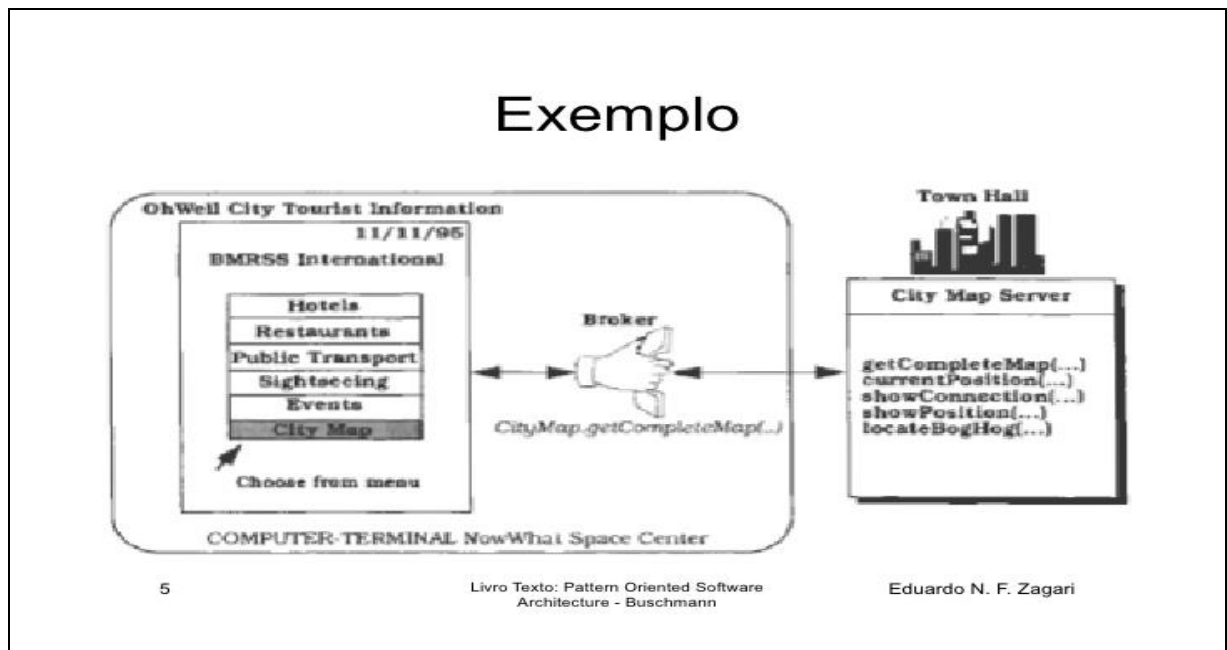


Figura 4 - SOA, exemplo de padrão de sistemas distribuídos [9].

Sistemas Interativos

- Um exemplo é o *MVC* (Model-View-Controller) que divide o sistema em:
 - Model: define a semântica da aplicação e define seu comportamento.
 - View: viabiliza uma apresentação visual da aplicação.
 - Controller: gerencia as interações do usuário com os modelos e visões da aplicação.

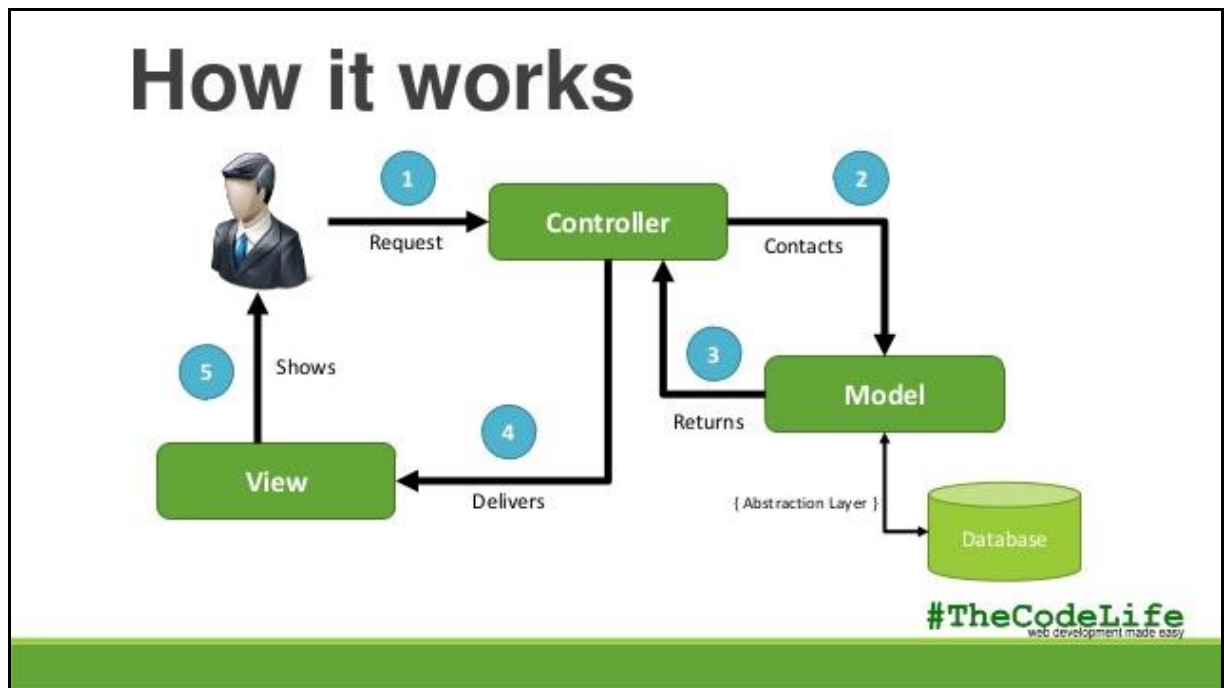


Figura 5 - MVC, exemplo de padrão de sistemas interativos [10].

Sistemas Adaptáveis

- Um exemplo é o Micro Kernel, que se aplica a sistemas de *software* que devem ser capazes de se adaptar às mudanças de requisitos do sistema. Ele separa um núcleo funcional mínimo de funcionalidade estendida e partes específicas do cliente. O Micro Kernel também serve como uma tomada para ligar essas extensões e coordenar a sua colaboração.

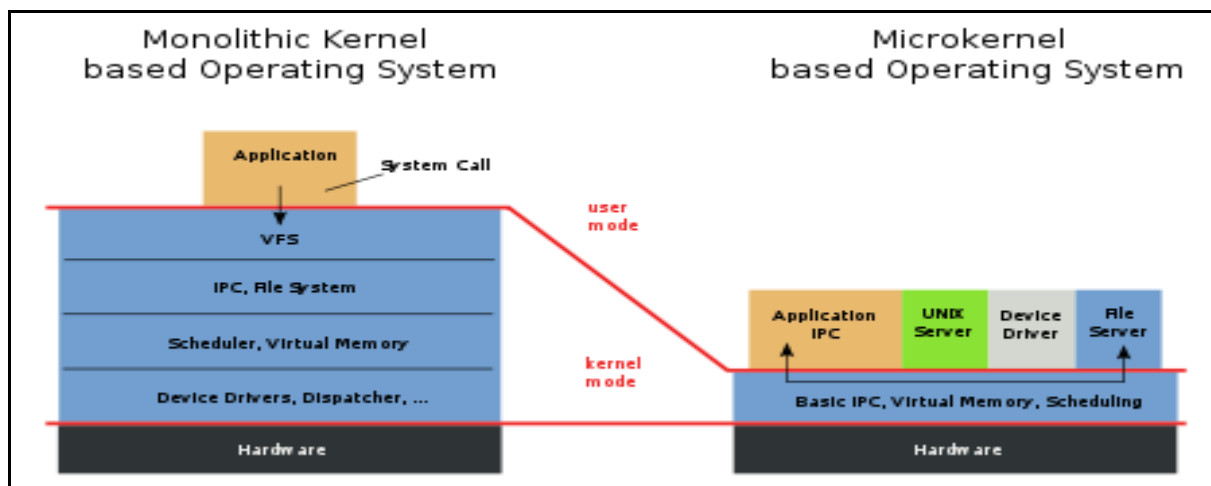


Figura 6 - Micro Kernel, exemplo de padrão de sistemas adaptáveis [11].

Dentre os padrões citados acima [9], pareceu ao projeto o mais adequado a ser seguido o de sistemas interativos. Visto que o projeto tende a atender muitas requisições de usuário por ser uma plataforma *web*, parece sensato utilizar um padrão como o MVC.

2.6 Framework

De acordo com Fayad e Schmidt, um *framework* é “um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação”. Um *framework* é responsável por determinar o fluxo de controle da aplicação e apresenta um conjunto de classes que facilitam o desenvolvimento [12]. A fim de agilizar o processo de programação e também possibilitar uma base mais documentada sobre o projeto, foi decidido buscar um *Framework PHP*.

A fim de agilizar o processo de programação e também possibilitar uma base mais documentada sobre o projeto, foi decidido analisar os *frameworks PHP* mais utilizados.




















PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Auth Module	Modules	EDP
Akelos 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ash.MVC 	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP 	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter 	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY 	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components 	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox 	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX 	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-
PHPDevShell 	-	✓	-	-	-	-	✓	-	-	✓	✓	✓	-
PhpOpenbiz 	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-
Prado 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP 	✓	✓	✓	✓	-	✓	✓	-	✓	✓	✓	✓	✓
Seagull 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Symfony 	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
WACT 	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	-
WASP 	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	-
Yii 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ZooP 	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-

Figura 7 - Comparativo de PHP frameworks [13].

A figura 7 mostra um comparativo de quase todos os frameworks php existentes e de diversos quesitos de comparação, porém deu importância apenas os quesitos e frameworks necessário para o projeto. Os quesitos de comparação priorizados são:

- MVC: indica se o framework tem suporte ao padrão *MVC*.
- Multiple DB's: indica o suporte a múltiplos bancos de dados sem exigir alterações
- DB Objects: indica se existe a possibilidade de diversos objetos de banco de dados.
- Templates: indica se o *framework* possui uma ferramenta de template pré-construída.
- Validation: indica se *framework* tem uma ferramenta de validação ou filtro de componentes.
- Ajax: indica se o framework suporta *Ajax*.

Após um Sprint realizado, chegou-se à conclusão que conforme as necessidades do projeto, os frameworks mais completos e que satisfaziam as necessidades eram: Zend e o Laravel. Portanto no projeto, o Zend foi apresentado como framework escolhido, pois, na época, foi indicado por programadores o de menor complexidade e completo. Após o levantamento das necessidades, foi percebido que utilizar um *framework* no meio do projeto acarretaria uma corrente de problemas que poderiam atrasar e colocar tudo a perder, portanto optou-se por não utilizar de nenhum *framework* específico, porém é reconhecido a utilização

para desenvolvimento *front-end* do *framework*. Durante o trabalho com o *Bootstrap*, percebeu-se que uma de suas grandes vantagens é o grande número de informações compartilhadas pela comunidade de *programadores php* e isso acabou facilitando na pesquisa e aprimoramento da turma.

2.7 Ferramentas auxiliares

2.7.1 KANBAN

O projeto utilizou o *software* Kanban Project Management Software, que é um *software* baseado no sistema Kanban, algumas vantagens na utilização desse *software* são: personalizar suas placas de acordo com suas atividades comerciais, múltiplos projetos com a capacidade de arrastar e soltar tarefas e sua ferramenta de relatórios e análises. Dessa forma o Kanban foi utilizado para registro das histórias na metodologia utilizada e também para controle do desenvolvimento das *Sprint*, existem alguns *softwares* proprietários que oferecem diversas funcionalidades, como o Burn Down Chart da metodologia; no entanto, existem muitos custos atrelados a eles. No intuito de manter uma aplicação sem custos, uma ferramenta um pouco mais limitada foi escolhida [14].

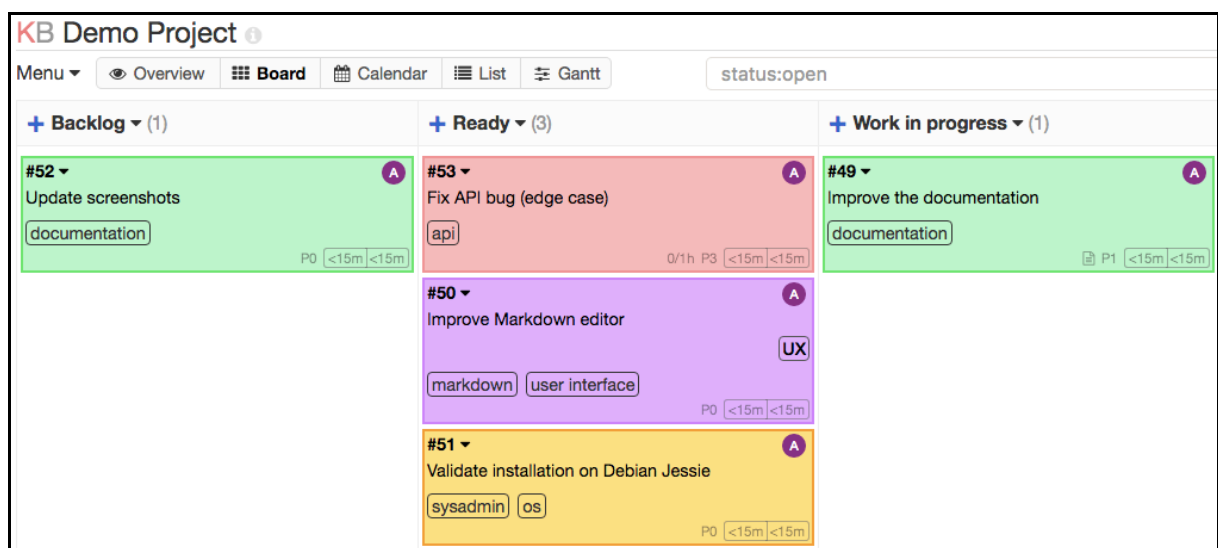


Figura 8 - Exemplo de um projeto sendo organizado no Kanban [14]

2.7.2 REDMINE

Redmine é um *software* livre, também como o Kanban é um gerenciador de projetos baseados na web e ferramenta de gerenciamento de *bugs*. Ele contém calendário e gráficos de *Gantt* para ajudar na representação visual dos projetos e seus deadlines (prazos de entrega). Ele pode também trabalhar com múltiplos projetos.

Um dos seus diferenciais é ser um software multilíngue possibilitando o uso integrado com vários repositórios tais como Svn, Git, Mercurial, Darcs, Cvs e Bazaar. Como optamos por trabalhar com repositórios para controle de versões foi interessante utilizar o Redmine. Foi escolhido trabalhar com dois softwares de gerenciamento, para dar mais experiência aos docentes [15].

The screenshot displays the Redmine web application interface. At the top, there is a navigation bar with links: Home, My page, Projects, Administration, and Help. Below this is a search bar and a 'Jump to a project..' dropdown. The main content area is titled 'Issues' and contains a table of issues. The table has columns for #, Tracker, Status, Priority, Subject, Assigned to, and Updated. A context menu is open over issue #79, showing options like Edit, Status, Priority, Assigned to, Copy, Move, and Delete. The right sidebar contains links for 'New issue', 'Tracker', 'Issues', 'View all issues', 'Summary', 'Change log', and 'Custom queries'.

#	Tracker	Status	Priority	Subject	Assigned to	Updated
127	Bug	New	Normal	Ticket with attachments		12/22/2007 12:11 PM
116	Bug	New	Low	Keep playing audio when rw/ff and preserve pitch.	John Smith	12/17/2007 09:56 PM
88	Feature	Assigned	Low	HTTP Challenge-MD5 authentication		12/22/2007 04:33 PM
83	Feature	Assigned	Low	Export the parameters of an input	John Smith	12/17/2007 09:56 PM
82	Feature	New	Low	Formatted text rendering support	Dave Loper	12/17/2007 06:58 PM
81	Feature	New	Normal	DVTS support		12/17/2007 06:58 PM
79	Feature	New	Low	QuickTime capturing		12/17/2007 06:58 PM
78	Feature	New	Low	Full H323 videoconferencing		12/17/2007 06:58 PM
77	Feature	Assigned	Low	Closed captions / Teletext support		12/17/2007 06:58 PM
74	Feature	New	Low	Progressive download playing		
73	Feature	New	Low	Dshow tuning support		
72	Feature	New	Low	V4L tuning support		
70	Feature	New	Low	Electric Program Guide		
69	Bug	New	Low	SDL vout cleaning		
65	Feature	New	Low	Protocol rollover support		
64	Feature	New	Normal	Improve ZLM functionality		12/22/2007 04:33 PM
63	Feature	New	Low	Gstreamer and Helix integration		12/17/2007 06:58 PM
62	Feature	New	Low	Gnutella servlet		12/17/2007 06:58 PM
59	Feature	New	Low	Finalization of Pocket PC port		12/17/2007 06:58 PM
58	Bug	Assigned	Low	Re-write of the AppleScript bindings		12/22/2007 04:33 PM
57	Feature	New	Low	MacOS X SVCD support	Dave Loper	12/17/2007 06:58 PM
51	Bug	New	Low	Better Mozilla plugin control		12/17/2007 06:58 PM

Figura 9 - Exemplo de um projeto sendo organizado no Redmine [16]

2.7.3 SVN

O Subversion (SVN) é um sistema de controle de versão *open-source* que gerencia arquivos e diretórios controlando as alterações realizadas ao longo do tempo. Além disso, é possível recuperar versões anteriores ou visualizar o histórico de alterações. Tem como ponto forte a utilização em rede, possibilitando que vários usuários possam trabalhar colaborativamente.

Permite que você recupere versões antigas de seus dados, ou que examine o histórico de suas alterações. É um sistema de caráter geral que pode ser usado para gerenciar quaisquer conjuntos de arquivos (código-fonte, arquivos de edição de vídeo, etc.).

Essa ferramenta foi essencial, já que possibilitou que umas equipes de mais de 30 pessoas conseguissem trabalhar em um único projeto, contudo problemas de controle de versões sempre ocorriam e foi, portanto, preciso que organizasse conversas com os interpersonais dos módulos para decidir sobre as versões e alterações indesejadas que ocorriam [17].

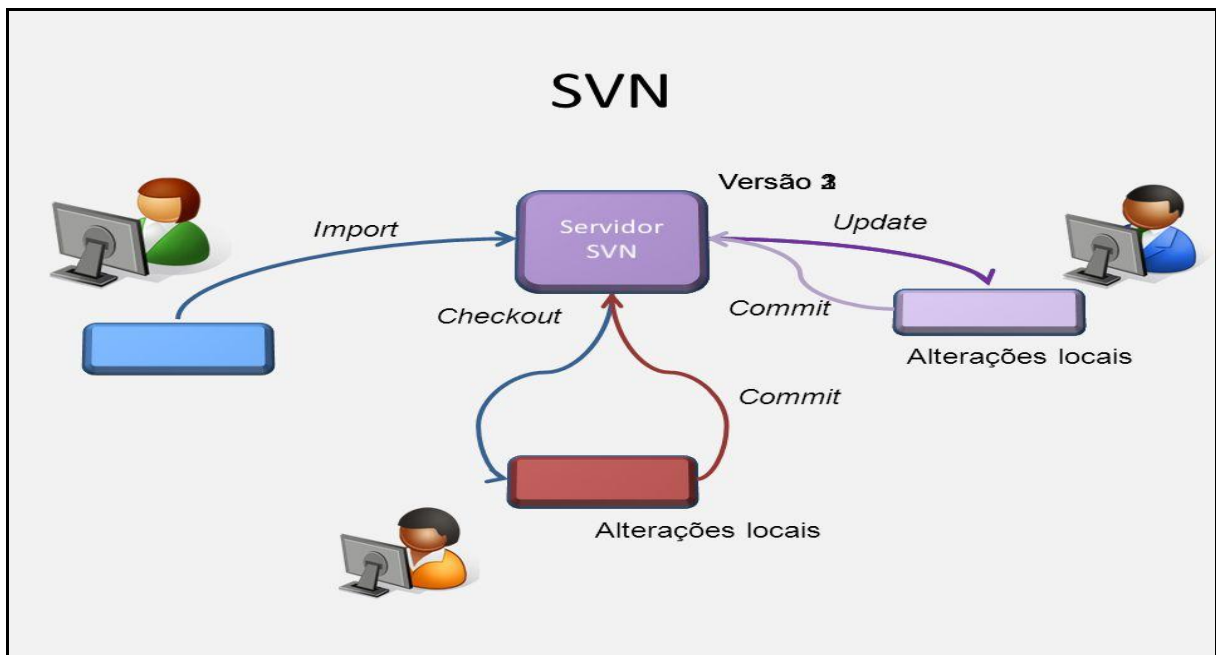
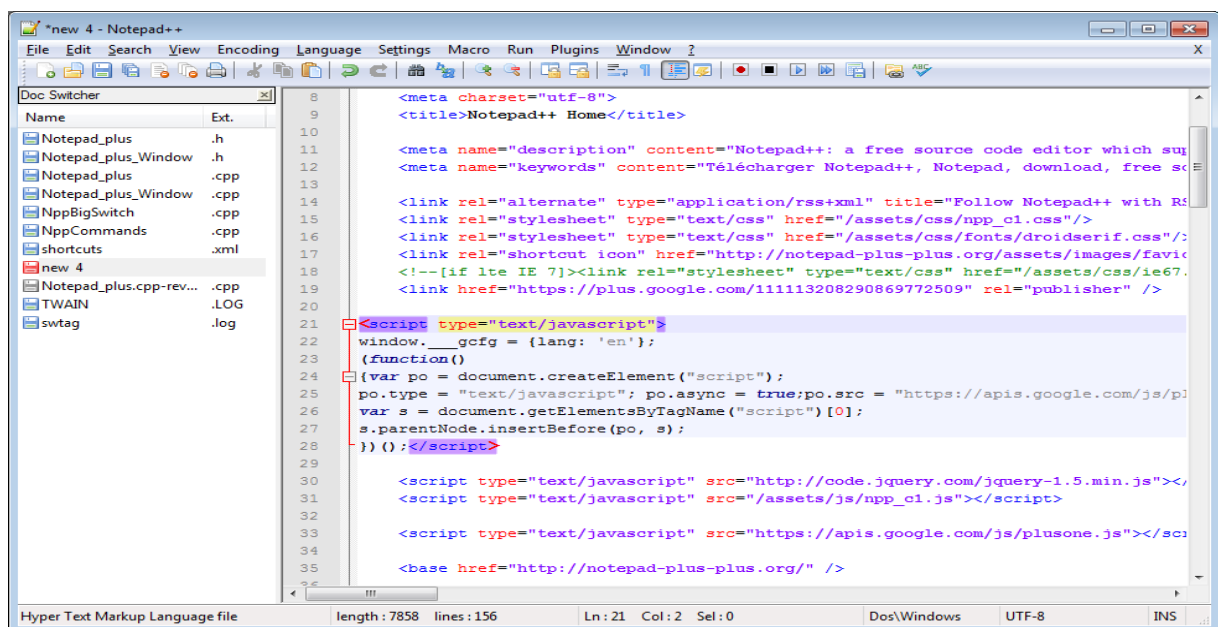


Figura 10 - Mostra como é o funcionamento estrutural do SVN [18].

2.7.4 NOTEPAD++

Notepad++ foi o editor de texto rápido e simples, escolhido pela equipe para desenvolver o projeto, por ser compatível com Windows e poder trabalhar com diversas linguagens de programação, ele foi a melhor ferramenta encontrada, contudo é reconhecido que alguns desenvolvedores utilizaram outras ferramentas, como: o Netbeans e o Adobe Dreamweaver CC [19].



**Figura 11 - Projeto sendo feito utilizando o Notepad++
Fonte: o autor.**

2.8 Organização do desenvolvimento

No início do projeto foi feito um planejamento, no qual dividiu-se o projeto em cinco módulos, cada módulo representa uma parcela do projeto final, sendo eles: Usuários, Reclamações, Estabelecimentos, Relatórios e Administrativo. Tendo por finalidade cada módulo cumprir os requisitos que foram definidos no início do projeto.

Essa modularização permitiu uma enorme versatilidade no desenvolvimento do site, onde cada conjunto de desenvolvedores, analistas e os administradores do banco de dados responsáveis por esse módulo pode se aprofundar e desenvolver com tranquilidade dentro do prazo, além é claro desse formato facilitar a comercialização do software futuramente, pois cada empresa poderá adquirir o pacote de módulos que melhor atende às suas necessidades. Outra vantagem desta opção modular é a possibilidade oferecida ao administrador do sistema de apontar as permissões de acessos dos usuários para cada módulo. Pode-se, por exemplo, permitir que somente os usuários do setor administrativo acessem os Relatórios Tabulares e Gráficos. Ou com um controle ainda mais específico, permitir que apenas os usuários cadastrados possam excluir suas reclamações feitas, veja a figura abaixo onde é exibido as características de cada módulo:

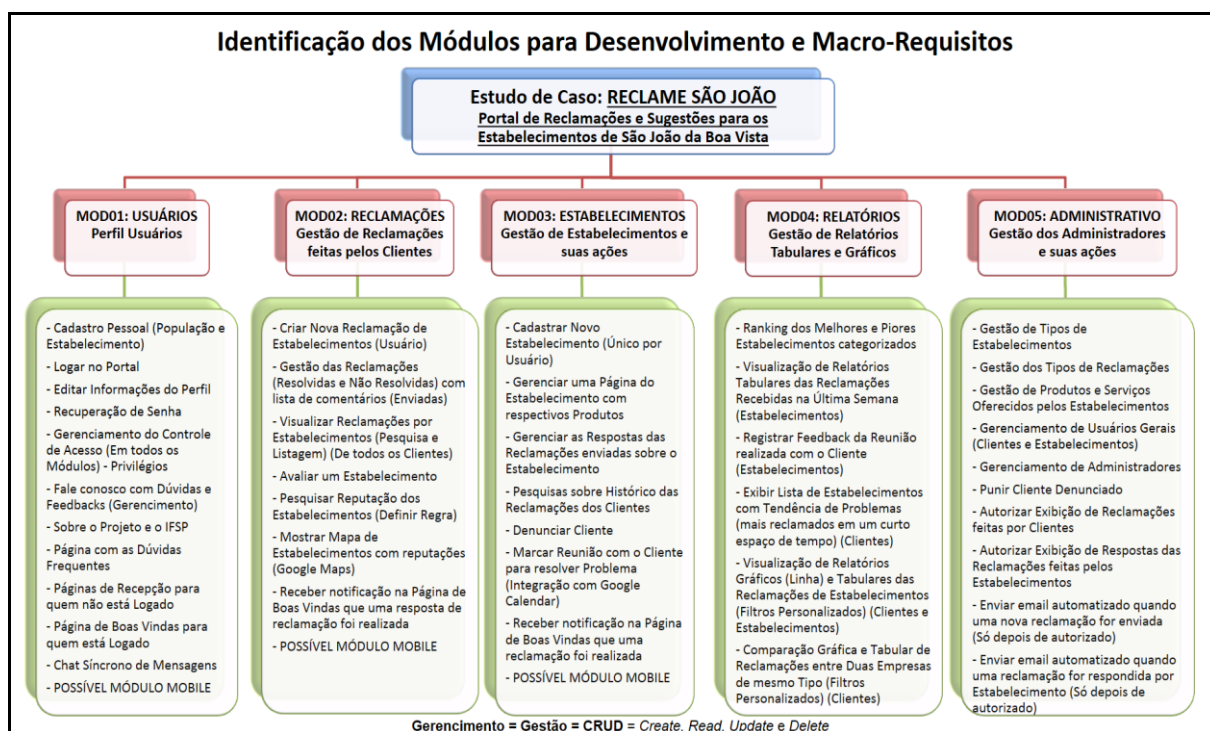


Figura 12 - Sistema dividido em módulos

Fonte: o autor.

3 Desenvolvimento Back-end e Front-end

3.1 Criação e Formação das páginas com HTML/CSS

O projeto começou a ser desenvolvido com a criação e formatação das páginas em *html*, nessa etapa, portanto definimos quais serão as cores do site e como os elementos serão distribuídos na tela. Inicialmente, desenvolvemos apenas a página inicial do site, que foi apresentada para o cliente para aprovação.



Figura 13 – Página Inicial

Fonte: o autor.

Após aprovação da página inicial começamos com o desenvolvimento das páginas internas. Nessa etapa cada módulo ficou responsável pelas suas páginas do site, o módulo de reclamações ficou responsável pelas páginas de reclamação e avaliação do projeto, portanto começou-se o desenvolvimento dessas páginas, primeiramente com a sua estrutura inicial, composta por seu *elemento raiz* mais os *metadados*:

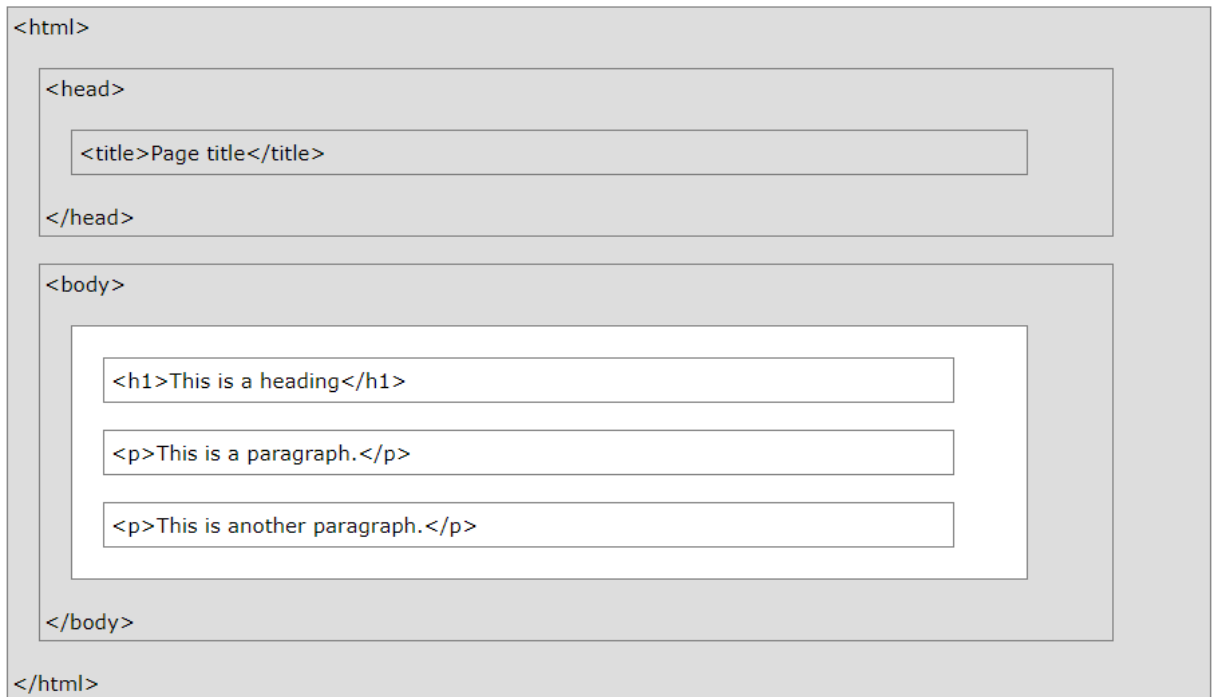


Figura 14 – Estrutura inicial de uma página html [20]

Após a criação dessa estrutura inicial começou o cumprimento de requisitos de cada página, populando a mesma com: *scripting*, *seções*, *tags de formatação*, *input*, entre outras.

```

<!-- Conteudo -->

<center><table id="rec_table_titulo">
<tr>
<td><center><span class="glyphicon glyphicon-fire" id="form_icon"></span></center></td>
<td><center><span class="form_titulo">Preencha o formulário para <br/>fazer sua <strong>reclamação!</strong></span></center></td>
</tr>
</table></center>

<center><table id="rec_table_titulo_responsivo">
<tr>
<td><center><span class="glyphicon glyphicon-fire" id="form_icon"></span></center></td>
</tr>
<tr>
<td><center><span class="form_titulo">Preencha o formulário para fazer sua <strong>reclamação!</strong></span></center></td>
</tr>
</table></center>

<div class="rec_div_formulario">
<div class="rec_div_conteudo_formulario">

<p class="rec_label">É muito importante que você faça a reclamação com o que está ocorrendo, relate tudo pois a empresa precisa saber que houve a
relação de consumo ou uma tentativa. Seja detalhista, mas de uma forma direta e objetiva.</p>

<br/>

<div class="widget" id="rec_checkbox">
<fieldset>
<table id="rec_table_checkbox">
<form action="php/mod02/cadastrarReclamacao.php" method="POST" enctype="multipart/form-data">
<label for="produto" class="rec_label" style="width: 100%;"><div class="popup" onclick="myFunction()"><i class="fa fa-info-circle" aria-hidden="true"></i>
<span class="popuptext" id="myPopup"><a id="rec_popup" href="rec_duvidas.php" target="_blank">Clique Aqui!</a> E saiba Mais</span>
</div>Qual é o tipo de reclamação?</label><br/>

<tr>

</tr>

</div>
<div class="rec_label">
<div class="rec_label_checkbox" for="checkbox-".$cont."".$row["TRC_CATEGORIA"]."></div>
<input type="checkbox" name="REC_NOTA[]" id="checkbox-".$cont."".$row["TRC_CATEGORIA"].">
</div>

</tr>


</div>
</table>

</div>

```

Figura 15 - Código dos primeiros protótipos
Fonte: o autor

Essas páginas foram feitas inicialmente utilizando apenas da linguagem *html*, juntamente com a folha de estilo *css*. O código da imagem acima é apenas uma parte do código *html* responsável pela seguinte página:

Preencha o formulário para
fazer sua **reclamação!**

É muito importante que você faça a reclamação com o que está ocorrendo, relate tudo pois a empresa precisa saber que houve a relação de consumo ou uma tentativa. Seja detalhista, mas de uma forma direta e objetiva.

Qual é o tipo de reclamação?

<input type="checkbox"/> Mal atendimento
<input type="checkbox"/> Produtos específicos
<input type="checkbox"/> Tempo de espera
<input type="checkbox"/> Erro na entrega
<input type="checkbox"/> Pessoal
<input type="checkbox"/> Sujeira
<input type="checkbox"/> Preço abusivo
<input type="checkbox"/> Acústica
<input type="checkbox"/> Empresa com produtos ruins

Título da reclamação

Figura 16 - Página de reclamações
Fonte: o autor

3.2 Comportamento com Java Script

O site utilizou da linguagem *JavaScript* para fazer o comportamento da linguagem *HTML* e *CSS* do site, possibilitando assim o surgimento de páginas mais “inteligentes”, com recursos adicionais e animações.

O código *JavaScript* foi escrito em um documento externo da página, do tipo *js*, nesse documento foi utilizado na grande maioria conceito básico de *JavaScript*, como funções *alerts* e *popup*:

```
function exibirMensagem()
{
    alert("Reclamação feita com sucesso!");
}

function myFunction() {
    var popup = document.getElementById("myPopup");
    popup.classList.toggle("show");
}
```

Figura 17 - Código utilizado para criação do alerts e popup
Fonte: o autor

Porém para cumprir alguns requisitos mais específicos foi necessário um estudo um pouco maior da linguagem *back-end JavaScript*. Um desses requisitos foi a criação da avaliação por estrelas da página de avaliação:

Figura 18 - Estrelas da avaliacao
Fonte: o autor

O seguinte código é responsável pelas animações da estrela:


```

$(function(){
    var average = $('#ratingAverage').attr('data-average');
    function avaliacao(average){
        average = (Number(average)*20);
        $('#bg').css('width', 0);
        $('#barra .bg').animate({width:average+'%'}, 500);
    }

    avaliacao(average);

    $('#star').on('click', function(){
        var idArticle = $('#article').attr('data-id');
        var voto = $(this).attr('data-vote');
        $.post('votar.php', {votar: 'sim', artigo: idArticle, ponto: voto}, 'JSON');
    });
});

```

Figura 19 - Código captura o voto do consumidor conforme o clique
Fonte: o autor

3.3 Linguagem SQL

A linguagem *SQL* foi desenvolvida pela IBM no final dos anos 70, e com seu enorme sucesso ganhou grande mercado no mundo do desenvolvimento de sites. O *SQL* é uma linguagem padrão, especificamente concebida para permitir que as pessoas a criem Bancos de Dados, adicionem novos dados a essas bases, manipulem os dados, e recuperem partes selecionadas dos dados [21].

Utilizando o *Xampp* foi possível fazer uma conexão com o banco de dados para que houvesse a comunicação da linguagem *php*, *html* e *sql*. O banco de dados do projeto foi estruturado conforme os documentos de requisitos de cada módulo, depois foi realizado *sprints* para formar um grande banco integrado unindo todo o site.

Abaixo é a imagem da versão final do banco de dados do projeto:

```

1  -- phpMyAdmin SQL Dump
2  -- version 4.5.1
3  -- http://www.phpmyadmin.net
4  --
5  -- Host: 127.0.0.1
6  -- Generation Time: 09-Out-2017 às 19:43
7  -- Versão do servidor: 10.1.13-MariaDB
8  -- PHP Version: 5.6.21
9
10 CREATE DATABASE reclame_sao_joao;
11 USE reclame_sao_joao;
12
13 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
14 SET time_zone = "+00:00";
15
16
17 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
18 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
19 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
20 /*!40101 SET NAMES utf8mb4 */;
21
22 --
23 -- Database: 'reclame_sao_joao'
24 --
25 -- site utilizado para os inserts http://www.generatedata.com
26 --
27 -----
28 --
29 -- Estrutura da tabela 'administradores'
30 --
31
32 CREATE TABLE `administradores` (
33   `ADM_TIPO_PRIVILEGIO` int(11) NOT NULL,
34   `ADM_ID` int(11) NOT NULL,
35   `USU_ID` int(11) NOT NULL,
36 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
37
38 --
39 -- Extrair dados da tabela 'administradores'
40 --
41
42 INSERT INTO `administradores` (`ADM_ID`, `ADM_TIPO_PRIVILEGIO`, `USU_ID`) VALUES
43 (1,1,3),
44 (2,2,4),
45 (3,2,41),
46 (4,2,42);
47
48 -----

```

Figura 20 - Banco de dados
Fonte: o autor

```

99 --
100 -- Estrutura da tabela 'consideracao_final'
101 --
102
103 CREATE TABLE `consideracao_final` (
104   `COF_ID` int(11) NOT NULL,
105   `COF_DESCRICAO` varchar(32) NOT NULL,
106   `REC_ID` int(11) NOT NULL,
107   `COF_STATUS_RESOLVIDO` int(11) NOT NULL,
108 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
109
110 --
111 -- Extrair dados da tabela 'consideracao_final'
112 --
113
114 INSERT INTO `consideracao_final` (`COF_ID`, `COF_DESCRICAO`, `COF_STATUS_RESOLVIDO`, `REC_ID`) VALUES
115 (1,"Tudo foi resolvido",1, 1),
116 (2,"Foi solucionado em parte",1, 2),
117 (3,"Nao foi resolvido",0, 3);
118
119 -----

```

Figura 21 - Código do banco de dados para tabela consideracao_final
Fonte: o autor

3.4 Estrutura MVC

Esse banco integrado foi utilizado pelos desenvolvedores tratar os dados do site, o padrão para desenvolvimento foi o *MVC*, com ele foi possível uma integração dos módulos e um encurtamento do código desenvolvido, vejamos a figura a baixo:

```

<?php
require_once($_SERVER['DOCUMENT_ROOT'] . '\RECLAME_SAO_JOAO-INTEGRAR\modelo\ConsideracaoFinal.php');
require_once 'Conexao.php';

class ConsideracaoFinalDAO {

    private $conexao;
    private $sql;
    private $post;
    private $resultado;
    private $tabela;

    public function __construct()
    {
        $conn = new Conexao();
        $this->conexao = $conn->getConexao();
        $this->tabela = "CONSIDERACAO_FINAL";
    }

    public function inserirConsideracaoFinal($consideracao)
    {
        $this->sql = "INSERT INTO $this->tabela (COF_DESCRICAO, REC_ID, COF_STATUS_RESOLVIDO)
        values (:COF_DESCRICAO, :REC_ID, :COF_STATUS_RESOLVIDO)";
        $this->resultado = $this->conexao->prepare($this->sql);
        $this->resultado->bindValue(':COF_DESCRICAO', $consideracao->getCOF_DESCRICAO());
        $this->resultado->bindValue(':REC_ID', $consideracao->getREC_ID());
        $this->resultado->bindValue(':COF_STATUS_RESOLVIDO', $consideracao->getCOF_STATUS_RESOLVIDO());
        $this->resultado->execute();
        return (($this->resultado->rowCount() > 0) ? true : false;
    }

    public function listarInformacoes($rec_id)
    {
        $this->sql="select * from $this->tabela WHERE REC_ID=:rec_id";
        $this->resultado= $this->conexao->prepare($this->sql);
        $this->resultado->bindParam(':rec_id',$rec_id);
        $this->resultado->execute();
        foreach ($this->resultado->fetchAll() as $linha)
        {
            return $linha['REC_ID'];
        }
    }
}

```

Figura 22 - Tratamento de dados do banco da Figura 20 e 21
Fonte: o autor

O código da Figura 22, é responsável por criar funções para inserção de variáveis e o seu retorno para a manipulação no site. Ele faz isso através de uma conexão criada com estrutura constructo, com a tabela consumidores_final do banco.

3.5 Linguagem PHP

A linguagem *php* foi demasiado utilizado nesse projeto, por tratar de um portal onde o cliente interage diversas vezes com o servidor, ela é frequente em todos as páginas desenvolvidas. O *php* que significa Personal Home Page é uma linguagem interpretada livre, ou seja, funciona e atua no lado do servidor, para então consegui-la acessa-la foi necessário o *Xampp* assim como o *SQL* [22].

Nas páginas desenvolvidas do módulo de reclamações, principalmente na página de gerenciar reclamações, o *php* foi utilizado em praticamente toda página. Essas páginas têm como objetivo exibir as reclamações do consumidor logado, portanto foi utilizado a estrutura

foreach que através da chamada do método (esse que continha o select na página DAO) permitiu a exibição dos dados, conforme ilustra a figura abaixo:

```
87 <?php
88 $id = 1;
89 $cont=0;
90 $cont2=0;
91 $status="";
92 foreach ($resultado as $row):
93
94     echo "<tr>";
95     echo "<td>".$id."</td>";
96
97     $REC_ID = $row["REC_ID"];
98     $REP_STATUS_APROVADO = $RespostaReclamacaoDAO->selecionarStatus($REC_ID);
99     $resultado_consideracao_final = $ConsideracaoFinalDAO->listarInformacoes($REC_ID);
100
101     echo "<td>" . ucfirst(substr($row["REC_TITULO_RECLAMACAO"], 0, 40)) . "</td>";
102     echo "<td>" . substr($row["COM_MOM"], 0, 40) . "</td>";
103
104     $cont++;
105     $id++;
106     if ($row["REC_APROVADO"] == 0): echo "<td><span class='label label-warning'>Pendente</span></td>";
107     $status = 'Pendente';
108     elseif ($row["REC_APROVADO"] == 1):
109         if ($resultado_consideracao_final!=null):
110             echo "<td><span class='label label-default'>Finalizada</span></td>";
111             $status = 'Finalizada';
112         else:
113             if ($REP_STATUS_APROVADO==1):
114                 echo "<td><span class='label label-default'>Respondida</span></td>";
115                 $status = 'Respondida';
116             else:
117                 echo "<td><span class='label label-default'>Publicada</span></td>";
118                 $status = 'Publicada';
119             endif;
120         endif;
121     elseif ($row["REC_APROVADO"] == -1):
122         echo "<td><span class='label label-danger'>Suspensa</span></td>";
123         $status = 'Suspensa';
124     endif;
125
126 >?
127 <td>
128 <?php if($status == 'Pendente'): ?>
129
130     <button type="button" class="btn btn-xs btn-primary" data-toggle="modal" data-target="#myModal">?>Visualizar</button>
131     <button type="button" class="btn btn-xs btn-warning" data-toggle="modal" data-target="#exampleModal" data-whatever="#?php echo $row['REC_ID']; ?>" data-whatevername="#?php echo
132     <button type="button" class="btn btn-xs btn-danger" data-toggle="modal" data-target="#removeModal" data-whatever="#?php echo $row['REC_ID']; ?>">Apagar</button>
133     <a href="rec_notificacao.php?REC_ID=?php echo $REC_ID;" target="_blank"><button type="button" class="btn btn-xs btn-info disabled">?</button></a>
```

Figura 23 - Página gerenciar reclamação responsável por exibir as reclamações
Fonte: o autor

3.6 Responsivo

Como requisito do cliente era um sistema responsivo, foi necessário um planejamento para que todo o site fosse desenvolvido de modo que seja compatível com qualquer dispositivo, seja ele um *tablet*, um celular, um *desktop* ou qualquer outro. Portanto foi feito Sprint no começo no meio e no fim do projeto para acompanhamento do design responsivo.

Para o sistema funcionar foi necessário utilizar algumas técnicas, como:

- Páginas flexíveis – essa técnica é basicamente estrutura em trocar as medidas fixas, conhecidas como pixel, por uma medida que seja adaptável a qualquer resolução, portanto o tipo de medida utilizada foi a porcentagem (%).

```

/* Antes */
.coluna {
    width: 264px; /* 264px / 1128px */
    float: left;
    margin-right: 24px; /* 24 / 1128px */
}

/* Depois */
.coluna {
    width: 23.404255319149%; /* 264px / 1128px */
    float: left;
    margin-right: 2.127659574468%; /* 24 / 1128px */
}

```

Figura 24 – Comparação de layout fixo e flexível
Fonte: o autor

- Imagens e recursos flexíveis – através de técnicas variadas, é possível fazer com que os recursos como imagens e vídeos, por exemplo, também sejam flexíveis para garantir que a experiência do visitante prime pela excelência [Zemel, 2012]. Uma das formas de alcançar esse objetivo é utilizando um recurso disponível em CSS, conforme exemplifica o trecho de código a seguir:

```

img{
    max-width: 100%;
}

```

Figura 25 - Uso de CSS para imagens
Fonte: o autor

- Css adaptativo: através de declarações *media*, é possível fazer com que a tela se adapte conforme o dispositivo.

```

1
2 @media (max-width: 1024px) {
3   #rec_table_titulo{
4     display:none;
5   }
6   .form_titulo{
7     font-size: 20px;
8     font-family: Calibri;
9   }
10
11   #form_icon{
12     font-size: 20px;
13   }
14 }
15
16 @media (min-width: 1024px) {
17   #rec_table_titulo_responsivo{
18     display:none;
19   }
20   .form_titulo{
21     font-size: 28px;
22     font-family: Calibri;
23   }
24   #form_icon{
25     position: relative;
26     transform: translateX(75%);
27     font-size: 25px;
28   }
29 }

```

Figura 26 – Código do css adaptativo
Fonte: o autor

Utilizando esses conceitos foi possível fazer com que o site Reclame São João fosse compatível com qualquer dispositivo, desde celulares a *desktops*

4 Conclusão e Recomendações

Passo o desenvolvimento do trabalho, vê que foi possível alcançar um dos objetivos principais, que era apresentar o projeto Reclame São João, sua metodologia de desenvolvimento e a introdução dos códigos *back-end* e *front-end*. Para tanto, foi realizada pesquisa em livros, artigos, teses, dissertações, sites de internet, entre outros.

Além disso, também foi possível a revisão do desenvolvimento do software ainda em execução, fato que proporcionou a aplicação prática de correções no site. Entretanto, a disponibilidade de tempo limitada para a dedicação ao projeto não permitiu que fossem mencionados todas as técnicas e códigos de desenvolvimento do módulo de reclamação, sendo reproduzido somente uma introdução daquilo que foi desenvolvido.

Para uma pré-aprovação do projeto foi realizado testes práticos de funcionamento, a website foi disponibilizada para comunidade na Semana da Tecnologia da escola e foi utilizada sistematicamente pelas escolas participantes. De acordo com dados da reitoria, cerca de 1000 alunos participaram do evento, sendo que uma parte significativa utilizou do site. Ainda segundo os membros participantes da apresentação do projeto não foram transcorridas nenhuma falha.

Diante deste teste de uso, é possível determinar que a *website* tem capacidade de atender perfeitamente a população de São João da Boa Vista, desde que o desenvolvimento seja levado adiante e conclua o restante do projeto.

Por fim, como recomendações, sugere-se a realização de novos trabalhos com o mesmo objetivo, considerando que a programação é um setor em expansão e exige uma cooperação da comunidade. A ampliação desse sistema em outros municípios poderá ser útil para uma melhor comunicação cliente/estabelecimento e ações voltadas para a divulgação de técnicas *back-end* livres para comunidade de desenvolvedores.

5 Referências Bibliográficas

[1] VIANA, Daniel. O que é front-end e back-end?. Disponível em:

<https://www.treinaweb.com.br/blog/o-que-e-front-end-e-back-end/>

Acesso em: 11 set. 2017.

[2] TIC Domicílios – 2016 Indivíduos.

Disponível em: <http://www.cetic.br/pesquisa/domicilios/indicadores>.

Acesso em: 12 set. 2017.

COSTA, JAIR (2000). O Processo de Desenvolvimento de Software.

Disponível em: <https://www.dimap.ufrn.br/~jair/ES/c2.html>.

Acesso em: 12 out. 2017.

LOPES A. R.; Planejamento e Controle da Produção: Um Estudo de Caso no Setor de Artigos Esportivos de uma Indústria Manufatureira. XXVIII ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO. Rio de Janeiro, 2008. MART

PRESSMAN, R. S. (1995). Engenharia de Software. Makron Books.

MENDONÇA, Filho. Estrutura de início de uma Página HTML5 (Semântica). 2013.

Disponível em:

http://web.unipar.br/~seinpar/2015/_include/artigos/Eduardo_Laguna_Rubai.pdf

Acesso em: 22 out. 2017.

PHP. Documentação oficial da linguagem PHP.

Disponível em: http://php.net/manual/pt_BR/

Acesso em: 07/10/2017.

[3] SLIDERPLAYER. XIV Jornada de Cursos. Slide 4 – Arquitetura Cliente/Servidor

Disponível em: <http://slideplayer.com.br/slide/3990076/>

Acesso em: 05/10/2017

[4] MOREIRA, THIAGO. Monografia Desenvolvimento de software de auxílio ao fluxo e ao compartilhamento de informações administrativas em ambientes empresariais de Software.

Disponível em:

http://www.ufjf.br/ep/files/2014/07/2005_3_Thiago.pdf

Acesso em: 02/10/2017.

[5] WIKIPEDIA. Modelo em Cascata. Disponível em: Acesso em: 28/10/2017.

[6] CASADACONSULTORA. Modelo Cascata.

Disponível em: <http://casadaconsultoria.com.br/modelo-cascata/>

Acesso em: 06/10/2017

[7] SOUZA, JESSÉ. Arquitetura De Software Sua importância no desenvolvimento de sistemas. Disponível em:

<https://pt.linkedin.com/pulse/arquitetura-de-software-sua-import%C3%A2ncia-sistemas-jess%C3%A9-souza>

Acesso em: 28/10/2017.

[8] Conceitos: Arquitetura de Software. Disponível em:

http://www.inf.pucrs.br/jornada.facin/jafacin_2010/palestras/ArquiteturaDeSoftware.pdf

Acesso em: 11/10/2017.

[9] BACELO, A. P. TERRA. Arquitetura de Software: conceitos e tendências. Disponível em:

http://www.inf.pucrs.br/jornada.facin/jafacin_2010/palestras/ArquiteturaDeSoftware.pdf

Acesso em: 11/10/2017.

[10] SlideShare. MVC, exemplo de padrão de sistemas interativos. Disponível em:

<https://www.slideshare.net/icapetillos/model-view-controller-mvc-31065496>

Acesso em: 12/10/2017

[11] WIKIPEDIA. Microkernel.

Disponível em: <https://en.wikipedia.org/wiki/Microkernel>

Acesso em: 14/10/2017

[12] DEVMEDIA. Função do Framework.

Disponível em: <https://www.devmedia.com.br/introducao-ao-laravel-framework-php/33173>

Acesso em: 17/10/2017

[13] NXNETBLOG. ¿La Comparativa Frameworks PHP. Disponível em:

<https://nxnethosting.com/blog/la-comparativa-frameworks-php-ya-esta-disponible/>

Acesso em: 12/10/2017

[14] KB PROJECT. Kanban Project Management Software.

Disponível em: <https://kanboard.net/>

Acesso em: 21/10/2017

[15] REDMINE. Visão Geral.

Disponível em: <http://www.redmine.org/projects/redmine>

Acesso em: 30/10/2017

[16] Flexible issue tracking system

Disponível em: <http://www.redmine.org/projects/redmine/wiki/Features>

Acesso em: 30/10/2017

[17] TORTOISE SVN. About TortoiseSVN. Disponível em: <https://tortoisesvn.net/>

Acesso em: 01/11/2017

[18] Controle de Versão SVN. Disponível em: <http://slideplayer.com.br/slide/395924/>

Acesso em: 01/11/2017

[19] Notepad. Disponível em: <https://notepad-plus-plus.org/download/v7.5.1.html>

Acesso em: 02/11/2017

[20] W3SCHOOL. HTML Introduction. Disponível em:

https://www.w3schools.com/html/html_intro.asp

Acesso em: 02/11/2017

[21] WIKIPEDIA. SQL. Disponível em: <https://pt.wikipedia.org/wiki/SQL>

Acesso em: 02/11/2017

[22] WIKIPEDIA. PHP. Disponível em: <https://pt.wikipedia.org/wiki/PHP>

Acesso em: 02/11/2017

ZEMEL, T. (2012) Web Design Responsivo Páginas Adaptáveis para todos os dispositivos. 1. ed. Casa do Código, 2012.

VOLPATO, TIAGO. Técnicas de Estruturação para Design Responsivo: Ampliando a Usabilidade no Ambiente Web. Disponível em:

http://web.unipar.br/~seinpar/2015/_include/artigos/Tiago_Volpato.pdf

Acesso em: 02/10/2017.