**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**Faculdade de Tecnologia da Baixada Santista**

**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**Alexandre Gomes da Silva**

**Rodrigo Vasconcelos da Silva**

**Portal da vida política**

**SANTOS**

**05/2014**

**Alexandre Gomes da Silva**

**Rodrigo Vasconcelos da Silva**

**PORTAL DA VIDA POLÍTICA**

Trabalho de Conclusão de Curso apresentada à Faculdade de Tecnologia da Baixada Santista – FATEC Rubens Lara, como exigência parcial para a obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

**Orientador:** Prof. Rosemeire Cardozo Vidal

**Santos**

**05/2014**

**Alexandre Gomes da Silva**

**Rodrigo Vasconcelos da Silva**

**PORTAL DA VIDA POLÍTICA**

Trabalho de Conclusão de Curso apresentada à Faculdade de Tecnologia da Baixada Santista – FATEC Rubens Lara, como exigência parcial para a obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

**Orientador:** Prof. Rosemeire Cardozo Vidal

BANCA EXAMINADORA:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nome do examinador:

Titulação:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nome do examinador:

Titulação:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nome do examinador:

Titulação:

**Santos:** Faculdade de Tecnologia da Baixada Santista

**Santos**

**05/2014**

*Dedicamos*,

Primeiramente a Deus por me amparar nos momentos em que precisei e não foram poucos, minha mãe Vilma Aparecida Vasconcelos e meu pai Aparecido Roque da Silva, que sempre me apoiaram e me deram uma estrutura familiar que precisei em meus momentos de aflição, meus amigos que sempre estiveram do meu lado me dando forças para continuar seguindo em frente.

*Rodrigo Vasconcelos da Silva*

**AGRADECIMENTOS**

Ao amigo e parceiro de TCC Alexandre Gomes, que foi de fundamental importância no meu aprendizado, pelos incentivos e por acreditar em meu empenho e me oferecer a oportunidade de iniciar minha carreira profissional na área de TI.

Á minha noiva Fabiane Mondim, pela paciência e compreensão nas vezes que tomei nosso tempo para o desenvolvimento da aplicação e documentação do trabalho.

Aos professores dedicados que colaboram com o desenvolvimento dos meus estudos e minha formação.

Á professora Rosemeire Cardozo Vidal pelas orientações e sugestões ao longo do desenvolvimento do relatório de qualificação e o trabalho de conclusão de curso. ***Rodrigo Vasconcelos da Silva***

“Se hoje fosse o último dia de minha vida, queria fazer o que vou fazer hoje? E se a resposta fosse Não muitos dias seguidos, sabia que precisava mudar algo”.

***Steve Jobs***

RESUMO

O foco principal desta monografia de conclusão de curso é criação de um portal colaborativo voltado a disponibilizar informações obtidas dos próprios eleitores através de fontes diversas sobre a vida política dos candidatos, utilizando na criação do portal, técnicas de engenharia de software, padrões de projeto e linguagem Java e várias outras ferramentas, o software desenvolvido cujo nome é pvp 1.0 tem as funcionalidades semelhantes às de um fórum cujos princípios e regras são regidos por moderadores, o colaborador que descumpri-las está sujeito a penalidades temporárias ou perpétuas que no caso seria a exclusão, o usuário poderá publicar, comentar noticias; o moderador poderá publicar comentar excluir e punidos usuários infratores.

**Palavras Chaves:** Portal Colaborativo, Colaborador, Informações, Vida política,Engenharia de Software, Padrões de projeto.

Sumário

[1. INTRODUÇÃO 12](#_Toc386363862)

[1.1. OBJETIVOS 16](#_Toc386363863)

[1.1.1. Gerais 16](#_Toc386363864)

[1.1.2. Específicos 16](#_Toc386363865)

[1.1.3. Quantitativa 17](#_Toc386363866)

[2. **Desenvolvimento do Software** 18](#_Toc386363867)

[2.1. Introdução 18](#_Toc386363868)

[2.2. Processos de Desenvolvimento 19](#_Toc386363869)

[2.3. Modelos de Ciclo de vida 19](#_Toc386363870)

[2.4. Modelo Cascata 19](#_Toc386363871)

[3. Arquitetura de Software e Framework Web 21](#_Toc386363872)

[3.1. Introdução 21](#_Toc386363873)

[3.2. Padrões no desenvolvimento de software e Design Arquitetural 22](#_Toc386363874)

[3.3. Categorias de padrões e Relacionamentos 22](#_Toc386363875)

[3.4. Framework Web 23](#_Toc386363876)

[3.4.1. Introdução 23](#_Toc386363877)

[3.4.2. Spring MVC 23](#_Toc386363878)

[3.4.3. Dispatche Servlet 23](#_Toc386363879)

[3.4.4. WebAplicationContext 24](#_Toc386363880)

[3.4.5. HandlerMapping 24](#_Toc386363881)

[3.4.6. Controller 25](#_Toc386363882)

[3.4.7. ViewResolver 25](#_Toc386363883)

[3.4.8. FrontController 26](#_Toc386363884)

[4. Conceitos Programáticos de Desenvolvimento 27](#_Toc386363885)

[4.1. Banco de dados Relacional 28](#_Toc386363886)

[4.2. Chaves primarias e estrangeiras 29](#_Toc386363887)

[4.3. Integridade 30](#_Toc386363888)

[4.3.1. Integridade de Entidade 30](#_Toc386363889)

[4.3.2. Integridade Referencial 31](#_Toc386363890)

[4.4. Conceito de orientação a objetos 31](#_Toc386363891)

[4.5. Mapeamento Objeto-relacional 32](#_Toc386363892)

[4.6. Padrões de Projeto 34](#_Toc386363893)

[4.6.1. Introdução 34](#_Toc386363894)

[4.6.2. Histórico 35](#_Toc386363895)

[4.6.3. Tipos de padrões de projeto 35](#_Toc386363896)

[4.6.4. DAO 36](#_Toc386363897)

[37](#_Toc386363898)

[4.6.5. Singleton 39](#_Toc386363899)

[4.6.6. Delegate 40](#_Toc386363900)

[4.6.7. Strategy 41](#_Toc386363901)

[4.6.8. Builder 42](#_Toc386363902)

[4.6.9. Abstract Factory 43](#_Toc386363903)

[4.7. Chain of Responsibility 44](#_Toc386363904)

[5. Especificação do Projeto 45](#_Toc386363905)

[5.1. Levantamento de Requisitos 45](#_Toc386363906)

[5.2. Descrição geral do software 45](#_Toc386363907)

[5.2.1. Descrição Resumida do Serviço 46](#_Toc386363908)

[5.2.2. Objetivos deste documento 46](#_Toc386363909)

[5.3. Definição de Papéis 46](#_Toc386363910)

[5.3.1. Usuário 46](#_Toc386363911)

[5.3.2. Moderador 47](#_Toc386363912)

[5.3.3. Administrador 48](#_Toc386363913)

[5.3.4. Anônimo 48](#_Toc386363914)

[5.4. Escopo do Software 48](#_Toc386363915)

[5.5. Requisitos Específicos 49](#_Toc386363916)

[5.5.1. Requisitos Funcionais 49](#_Toc386363917)

[5.5.2. Requisitos não Funcionais 51](#_Toc386363918)

[5.5.3. Descrição dos tipos de usuário 51](#_Toc386363919)

[6. Modelo de analise de software 52](#_Toc386363920)

[6.1. Regras adotadas para o Portal 52](#_Toc386363921)

[RN01 52](#_Toc386363922)

[Nome 52](#_Toc386363923)

[Descrição 52](#_Toc386363924)

[RN02 52](#_Toc386363925)

[Nome 52](#_Toc386363926)

[Descrição 52](#_Toc386363927)

[RN03 52](#_Toc386363928)

[Nome 52](#_Toc386363929)

[RN04 52](#_Toc386363930)

[Nome 52](#_Toc386363931)

[Descrição 52](#_Toc386363932)

[6.2. Restrições 53](#_Toc386363933)

[N 53](#_Toc386363934)

[6.3. Descrição geral de funcionamento 54](#_Toc386363935)

[6.3.1. Diagrama de contexto 54](#_Toc386363936)

[6.3.2. Diagrama de Classe 55](#_Toc386363937)

[Figura 2 - Diagrama de Classes representando as principais entidades do Portal. 55](#_Toc386363938)

[6.3.3. Diagrama de Caso de Uso 56](#_Toc386363939)

[Figura 5 - Caso de uso do usuário Moderador 57](#_Toc386363940)

[6.3.4. Diagrama de Atividade 58](#_Toc386363941)

[Figura 9 - Exclusão de usuários. 61](#_Toc386363942)

[6.3.5. Telas do Portal 62](#_Toc386363943)

[N 62](#_Toc386363944)

[Nome da Tela 62](#_Toc386363945)

[Ator 62](#_Toc386363946)

[Caso de uso 62](#_Toc386363947)

[Descrição 62](#_Toc386363948)

[6.3.6. Especificação do Cenário 63](#_Toc386363949)

[7. Divisão de Camadas 68](#_Toc386363950)

[7.1. Aplicações monolíticas 68](#_Toc386363951)

[7.2. Aplicações de duas camadas 68](#_Toc386363952)

[7.3. Aplicações de três camadas 69](#_Toc386363953)

[7.4. MVC 70](#_Toc386363954)

[1. Ambiente de desenvolvimento 72](#_Toc386363955)

[Orientação a objetos 72](#_Toc386363956)

[Histórico da Linguagem Java 72](#_Toc386363957)

[Arquitetura MVC 73](#_Toc386363958)

[Eclipse IDE 73](#_Toc386363959)

[Hibernate Tools 74](#_Toc386363960)

[Scrum 74](#_Toc386363961)

[REFERÊNCIAS BIBLIOGRÁFICAS 76](#_Toc386363962)

**Figuras**

[Figura 1 - Gráfico demonstrativo das fontes de informações eleitorais .......................... 7](#_Toc386123796)

[Figura 2 - Diagrama de Classes representando as principais entidades do Portal ..........11](#_Toc386123797)

[Figura 3 - Casos de uso do usuário Administrador.................................................11](#_Toc386123798)2

[Figura 4 - Caso de uso do usuario Colaborador 13](#_Toc386123799)

[Figura 5 - Caso de uso do usuário Moderador 14](#_Toc386123800)

[Figura 6 - Envio de mensagem 17](#_Toc386123801)

[Figura 7 - Inserção de novo tópico 18](#_Toc386123802)

[Figura 8 - Interação de Moderador com tópico e comentário 20](#_Toc386123803)

[Figura 9 - Exclusão de usuários 2](#_Toc386123804)9

[Figura 10 - Representação dos conceitos do modelo relacional 23](#_Toc386123805)

Figura 11 - Representação dos conceitos de chave estrangeira  [24](#_Toc386123806)

[Figura 12 - Representação do conceito Objeto 25](#_Toc386123807)

Figura 13 - Representação dos conceitos ORM

Figura 14 - Representação do conceito do padrão de projeto DAO

Figura 15 - Representação do Conceito de Strategy

Figura 16 - Representação do Conceito de Builder

Figura 17 - Representação do Conceito de Abstract Factory

Figura 18 - Representação de Conceito Chain of Responsability

Figura 19 - Exemplo de conceito de camada Monolítica

Figura 20 - Exemplo de conceito de duas camadas

Figura 21 - Exemplo do Conceito de três camadas

Figura 22 - Representação do Conceito MVC

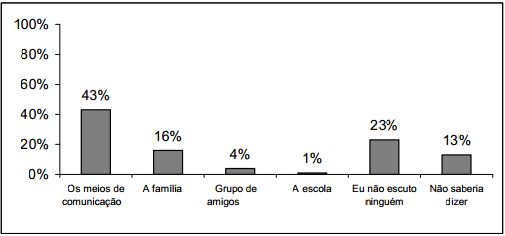
Figura 23 - Representação do modelo de Cascata

Figura 24 - Representação do fluxo do modelo Iterativo Incremental entre uma fase e a seguinte

Figura 25 - Ilustração global de atores e casos de uso

# INTRODUÇÃO

À medida que as eleições se aproximam as propagandas políticas, principalmente televisivas passam a fazer parte do nosso dia-a-dia, o que possibilita aos eleitores conhecer melhor os candidatos, suas propostas e ideias. Atualmente, segundo estatísticas, grande parte dos eleitores conhece o candidato apenas pelos meios midiáticos tradicionais, uma forma que mostra apenas o que o candidato fez de bom em sua vida pública. Analisando dessa maneira, a fonte de informação que chega até o eleitor é pouco diversificada, consequentemente por estar acostumado a utilizar esta fonte acaba sendo forte influenciado, sendo mostrado a ele apenas informações a favor do candidato ou do partido. Diante do exposto, a internet pode ser uma forte aliada na obtenção de informações reais sobre os candidatos, que por ser uma rede mundial possibilita a circulação de uma série de informações, que poderiam ser centralizadas e filtradas. Um portal colaborativo com uma base de dados, alimentada com informações confiável obtidas pelos próprios eleitores sendo uma alternativa para meios midiáticos tradicionais, composto de evidências e fontes diversificadas, de maneira que o eleitor possa tirar suas próprias conclusões sobre o candidato e posteriormente fazer sua escolha baseado nas informações obtidas.



**Figura 1 - Gráfico demonstrativo das fontes de informações eleitorais.**

Fonte: Pesquisa sobre o perfil político dos jovens em Londrina (PR), 2006. N= 270.

Quando votamos em um candidato, depositamos nele nossa confiança, imaginamos que este será diferente dos demais, muitas vezes não nos importamos se quer em avaliarmos o passado dele e suas ações mediante ao cenário publico, votamos simplesmente por costume. Em propagandas eleitorais os candidatos impressionam e conquistam o eleitor com planos e projetos de governo que apresentam, porém conforme o tempo passa vemos que as propostas e metas são intangíveis e o plano de governo é inconsistente.

É preciso trazer a verdade ao eleitor, verdade esta que na maioria das vezes não é relatadas em campanhas que mostram apenas ‘o bom politico’, deixando de lado processos que este possa estar respondendo, os projetos que participou e que não deram resultado nenhum, os desperdícios de verbas publicas em obras inacabadas ou superfaturadas.

As mídias tradicionais, na grande maioria das vezes, trazem batalhas entre partidos e candidatos, com troca de acusações e informações duvidosas, fazendo com que o eleitor fique ainda mais confuso em sua opção de voto. Buscar informações sobre os candidatos gera algum trabalho, pois na maioria dos casos estas estão muito descentralizadas, e geralmente o eleitor não tem conhecimento de como fazer essa pesquisar.

No inicio o portal será divulgado em vários *links* em redes sociais e fóruns que poderá obter mais visibilidade e acessos, trará informações sobre políticos que aparentemente já são conhecidos pelo eleitor ou que participaram das ultimas eleições.

O portal conterá informações pré-cadastradas de alguns políticos conhecidos e seus projetos com isso atiçará a curiosidade dos eleitores, fazendo com que o eleitor conheça o candidato que em outrora votou, e os demais candidatos podendo compara-los, definindo critérios de voto que antes não tinha possibilidade de efetuar, deixando para traz a forma intuitiva induzida pelas propagandas eleitorais. O resultado é um eleitor mais consciente conhecedor parcial ou até mesmo completo do candidato, com dados confiáveis.

Em tempos de manifestações e protestos, cujos objetivos são os mais variados, o eleitor espera do governo, resultados que foram prometidos em tempos de eleição. Desta forma há a necessidade de mais transparência sobre a vida política dos candidatos, para que na hora de decidir o voto, o eleitor esteja ciente em quem realmente está votando, com um conhecimento baseado em informações verídicas e divulgado por vários outros eleitores, podendo a qualquer momento ter acesso a essas informações de forma fácil, rápida e centralizada.

Hoje temos acesso a uma massa de informação muito grande, disponibilizadas pelos mais variados meios de comunicação, porém quando se fala de política, a informação se torna essencial para uma escolha consciente do candidato, pois o poder de escolha permite o poder de mudança. Confiar apenas nas informações que nos são trazidas por propagandas televisivas, por exemplo, não garante que essas não tiveram algum tipo de manipulação, seja pessoal ou partidária, principalmente em tempos de eleição onde são apresentadas no horário politico pesquisas de pretensão de votos de origens diversas nem sempre confiáveis, indicando candidato do partido ao cargo, com índices que podem não ser verídicos.

Os meios atuais para obtenção de informação na escolha do voto satisfazem a necessidade de informação do eleitor. Um espaço de comunicação confiável, onde todos os eleitores pudessem ter acesso a informações do candidato, mandato atual ou anteriores, processos em que está sendo julgado. O eleitor poderia se manter atualizado e se conscientizar, sabendo tudo o que o possível eleito em questão, fez e está fazendo para prover melhorias e demonstrar ser merecedor de confiança e voto nos tempos de eleição.

Hoje há informação de forma a prover ao eleitor a oportunidade de se auto informar sobre os candidatos, com isso, se faz necessário um portal de informações centralizadas, organizada e coerente, para que o eleitor obtenha o conhecimento que necessita antes de votar e durante o mandato do eleito, para assim poder cobrar a confiança que foi depositada no político.

Um portal com publicações sobre os candidatos nas eleições traz para o eleitor a opção de conhecer melhor seus candidatos, com isso poderá votar com maior propriedade. O portal permitirá que o eleitor forneça informações que forem de seu conhecimento. Havendo moderadores que terão como principal papel verificar a origem da publicação feita pelo usuário, assim o eleitor terá informações trazidas por eleitores.

## 

## OBJETIVOS

### Gerais

O portal terá como medida apenas informar, cabendo ao eleitor formar sua própria opinião com base nas publicações e noticias do portal, tendo o(s) moderador (es) um cuidado em analisar as matérias disponibilizadas, fazendo uma verdadeira investigação sobre a legitimidade da informação. O objetivo é conscientizar o eleitor sobre o(s) candidato(s) que pretende votar, mostrando suas ações praticadas e omitidas perante o cenário político antes, durante e depois das eleições. O portal será uma importante ferramenta de monitoramento sobre tudo o que venha a ter uma relevância política e seja direito do eleitor saber.

Com a ideia de tornar publico o que não é divulgado nas propagandas políticas, o portal levará informação de eleitores para eleitores tendo como a internet um veículo de comunicação. O portal não terá partido, não terá bandeira, seguirá uma única ideologia, a verdade. A internet possui a liberdade necessária para esse tipo de conteúdo, divulgando informações com relevância, para a formação de opinião dos eleitores, com uma linguagem fácil onde até mesmo os mais leigos em política, poderão compreender. Muitas dessas informações já estão postadas em *sites* governamentais, porém com uma linguagem maçante e de difícil entendimento, nada atrativa e dificultando assim os eleitores de se manterem informados.

### Específicos

Este projeto iniciará com a elaboração de documentação funcional para atender as necessidades de controle de um projeto, com foco principal em tarefas, recursos e custos. Posteriormente serão elaborados os descritivos técnicos, modelo de dados e diagramas UML. Terminada a primeira fase de documentação, será iniciada a fase de desenvolvimento do sistema, que engloba metodologia e ferramentas técnicas.

### Quantitativa

Conforme mostra a Figura 1 os meios de comunicação são onde o eleitor mais buscam informações sobre os candidatos. O portal será uma alternativa para o eleitor se informar sobre seus candidatos, e assim comparar com as informações que chegam até ele por propaganda eleitoral, contendo sempre a origem da informação disponível para que o próprio eleitor possa conferi-las. Através do número de acesso poderemos monitorar a aceitação do portal pelos internautas e apresentar o interesse dos eleitores em conhecer melhor suas opções de voto

1. Desenvolvimento do Software

## Introdução

Com a crescente importância que os computadores ganharam no mundo corporativo desde suas primeiras distribuições, abriu-se mercado para o desenvolvimento de softwares para diversas áreas profissionais, tendo as empresas o interesse em soluções computadorizadas gerando enormes demandas.

Nos anos de 1960 o desenvolvimento de softwares por encomenda começou a enfrentar sérios problemas recorrentes as altas demandas. Estimativas de tempo e custos eram necessárias para desenvolvedores e cliente, com a intenção de medir a viabilidade da aquisição e desenvolvimento. A qualidade do software estava depreciada e o cliente insatisfeito com as garantias e custos de manutenções. Em uma maneira de contornar os complexos problemas de desenvolvimento surgiu à engenharia de software, uma ciência voltada a estudo detalhado das etapas de desenvolvimento de software, tendo como foco a qualidade, a organização e a produtividade, especificando as etapas, aplicando boas práticas de mercado e tecnologias da ciência da computação.

A engenharia de software segue alguns princípios científicos em sua abordagem, sendo eles disciplinar, quantificável e sistemático. Disciplinar, pois segue o principio que os processos definidos e detalhados serão seguidos; quantificável, pois tem medição da qualidade do software, produtividade em seu desenvolvimento e operação e manutenção, controle do desenvolvimento dentro dos custos, prazos e qualidade acordado; sistemático, devendo haver um processo de desenvolvimento com as definições de atividade a serem executadas pelos programadores.

## Processos de Desenvolvimento

O processo de desenvolvimento de softwares trata-se do conjunto de atividades organizadas que se relacionam e são executadas ao longo do ciclo de vida de um software, ou seja, desenvolvimento, manutenção e testes. Esses processos têm o intuito mapear os passos necessários para a definição, planejamento e a gestão do software á ser desenvolvido. Como já é de se esperar existem diversos processos de desenvolvimento de software, mas há alguns processos básicos comuns à maioria dos processos, como levantamento de requisitos, análise de requisitos, projeto, implementação, testes e implantação.

## Modelos de Ciclo de vida

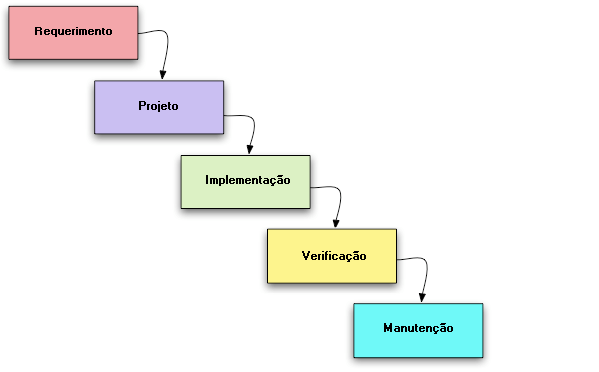
A ideia de ciclo de vida trata-se da descrição do conjunto de atividades e como elas interagem entres si. Esse modelo auxilia os gestores e suas equipes de desenvolvimento a ter um escopo amplo do projeto entendendo todas as etapas de desenvolvimento.

O ciclo de vida refere-se a um encadeamento das fases da construção do sistema. Há diversos modelos de ciclos de vida mais como estão encadeadas, os modelos mais utilizados são os modelo em cascata e o modelo Iterativo e Incremental (BEZERRA, 2002).

## Modelo Cascata

O termo cascata se refere ao fluxo corrente do desenvolvimento através de etapas de analise de (requisitos, projetos, implementações e testes, integração e manutenção) como é mostrado na figura 2, também conhecido como clássico ou linear tem como fundamento progressões entre as etapas.

É um modelo que requer uma abordagem sistemática, sequencial ao desenvolvimento do software, que se inicia no nível do sistema e avança ao longo da análise, projeto, codificação teste e implantação (BEZERRA, 2002).



**Figura 2 - Representação do modelo de Cascata.**

Fonte: <http://pt.wikipedia.org/wiki/Ficheiro:Modelo_em_cascata.png>

O modelo em cascata foi o que melhor atendeu as necessidades do desenvolvimento do portal em vista do seu fluxo, desde sua especificação até sua verificação, passando por todas as etapas recorrentes do modelo.

# Divisão de Camadas

## Aplicações monolíticas

Antigamente um software era produzido para ser utilizado integralmente em uma máquina, com isso sua criação se continha em um modulo robusto, que possuía todas as funcionalidades pertinentes, gerando uma grande quantidade de código, contendo a interface com usuário, verificação, regra de negócio e manipulação do banco de dados presentes em um único lugar conforme mostra a figura 19, tornando a manutenção do software uma tarefa árdua.(MACORATTI, 2004)



**Figura 19 - Exemplo de conceito de camada Monolítica.**

FONTE: <http://www.macoratti.net/vbn_mvc.htm>

## **Aplicações de duas camadas**

Com a necessidade do compartilhamento de informações entre vários usuários simultaneamente, fez surgir à necessidade da arquitetura de duas camadas. Nessa arquitetura o banco de dados não está alocado mais localmente na estação de cada usuário, mas sim em uma maquina especifica, separada das máquinas que executam o software como mostra na figura 20, sendo acessada por vários usuários simultaneamente ficando na estação a lógica de apresentação e a lógica de negócio (clientes ricos). Entretanto surgi um novo problema, que é a gestão de versões do software, para cada atualização realizada para software, será necessário atualizar todas as máquinas cliente. (MACORATTI, 2004)



**Figura 20 - Exemplo de conceito de duas camadas.**

FONTE: <http://www.macoratti.net/vbn_mvc.htm>

## Aplicações de três camadas

Com a evolução da internet surgiu à ideia de separar a regra de negócio da interface com o usuário como mostra na figura 21, com a intenção de que os usuários da Web acessem a aplicação sem a necessidade de instalar a aplicação em sua(s) máquina(s). Com essa arquitetura os aplicativos passaram para um Servidor Web podendo ser acessado pelos clientes (cliente pobre), que passaram a ser os navegadores web.



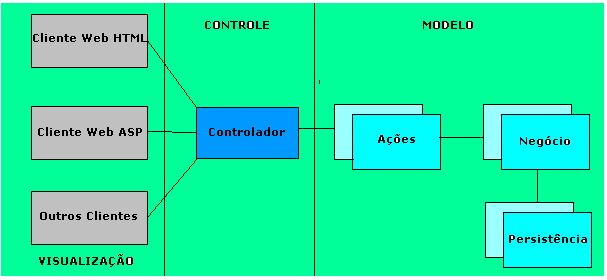
**Figura 21 - Exemplo do Conceito de três camadas**

FONTE: <http://www.macoratti.net/vbn_mvc.htm>

Nesse modelo existe a separação lógica e física dos componentes de apresentação (layout) da regra de negócio, que se encontra no servidor Web e a separação física do acesso aos dados. A separação lógica do aplicativo em camadas, a torna flexível permitindo alterações de forma independente. Havendo a divisão da regra de negócio em várias classes, reduz a dependência entre as classes e pacotes, sendo possível reaproveitamento maior do código que pode ser reutilizado em diferentes lugares da aplicação e até mesmo em aplicações distintas. (MACORATTI, 2004).

O modelo de três camadas é um padrão no desenvolvimento de softwares corporativos baseados na Web. Utiliza uma modelagem baseada na orientação objeto, auxilia na modularidade, no encapsulamento das propriedades e estados do objeto. Sendo aplicado da maneira correta, a dependência entre os objetos são reduzidas diminuindo o acoplamento e melhorando o código em compreensão manutenção.(MACORATTI, 2004)

## MVC



**Figura 22 - Representação do Conceito MVC.**

FONTE: <http://www.macoratti.net/vbn_mvc.htm>

Conforme é mostrado na Figura 22 segue a sua descrição:

A camada de apresentação exibe apenas a informação, não sendo de seu interesse onde ela foi obtida ou como a informação foi obtida.

* Tecnologias utilizadas no cliente: CSS, JSP, HTML, JSF;
* Camada que o usuário interage;
* Responsável por receber os dados e exibir um resultado ao usuário.

A Camada de Controle é responsável por controlar o fluxo da apresentação, sendo uma camada intermediária entre a camada de apresentação e a lógica.

A camada modelo é divida em três subcamadas que são

* A subcamada da regra de negocio é o core do software.

Define a regra de negocio que o software conterá em seu funcionamento sendo definido pela empresa na especificação.

* Modela comportamento por atrás do processo de negócios e os dados;

A subcamada de persistência é responsável pelas principais interações com o banco

* É responsável apenas com o armazenamento, manipulando e gerando de dados;

A subcamada de ações tem as funções de:

* Encapsula os dados e seu comportamento independe da apresentação.
* Controla e mapeia as ações. (MACORATTI, 2004).

# Arquitetura de Software e Framework Web

## Introdução

A complexidade da informação é um problema que acompanha o campo da computação desde sua criação. A partir disso começou-se a ser desenvolvidas formas de se particionar os sistemas tendo em vista uma compreensão mais precisa do software, o reuso de funções ou partes do código, melhor gerenciamento do sistema em execução minimizando suas complexidades assim o conceito de arquitetura de software começou a ganhar espaço.

O termo engenharia de requisitos também pode ser usado na documentação de arquitetura de software, tendo como papel o mapeamento dos requisitos iniciais do projeto em alto nível, o reaproveitamento de padrões e componentes entre projetos e a facilitação da comunicação entre os *stakeholders.*

Um dos conceitos de arquitetura de software trata-se da identificação dos componentes de software, suas propriedades externas e sua interação com os demais sistemas. A arquitetura de software especifica de forma abstrata os componentes e suas relações, tendo como objetivo visualizar e documentar o funcionamento da estrutura separadamente da linguagem de programação.

A arquitetura pode ser descrita por diversos diagramas proporcionando visões distintas sendo estruturas lógicas ou físicas, estática ou dinâmica de seu funcionamento. As estruturas lógicas englobam conceitos lógicos como classes, funções, variáveis; estruturas físicas correspondem a conjunto de arquivos fonte, bibliotecas, arquivos fontes executáveis que formam a estrutura física do software.

xxxxx

## Padrões no desenvolvimento de software e Design Arquitetural

A partir de experiências colaborativas de programadores, as soluções para problemas similares e recorrentes formam à base dos Padrões.

Os problemas não são isolados contendo componentes (propriedades, restrições, requisitos) externos que estão associados aos problemas. A solução obedece aos componentes do problema, suas interconexões configuração e estrutura.

Os padrões podem ter desde nível conceitual, a nível de arquitetura de software, sendo abstratos apresentam soluções de desenvolvimento, podendo ser instanciados quando encontram problemas únicos e similares em diferentes projetos.

Os métodos são ações que o objeto faz ao longo do sistema sendo independentes do problema, assim os métodos não abordam como resolver certos problemas, mas sim, mostram os passos do desenvolvimento e as notações para descrição do sistema, diferentemente dos padrões que são dependentes do problema.

## Categorias de padrões e Relacionamentos

Cada padrão depende de padrões menores que ele contém e de padrões maiores no qual ele está contido.

* **Padrões arquiteturais**

Expressam o esquema de organização estrutural fundamental para um sistema de software.

Assemelham-se aos Estilos Arquiteturais descritos por [Shaw & Garlan 96].

* **Padrões de design**

Provê um esquema para refinamento dos subsistemas ou componentes de um sistema de software.

* **Idiomas**

São padrões de baixo nível, específicos para o desenvolvimento em uma determinada linguagem de programação, descrevendo como implementar aspectos particulares de cada componente.

## Framework Web

### Introdução

Seu por objetivo balancear sites e aplicações das sobrecargas exigidas nas atividades de desenvolvimento como acesso a banco, gerenciamento de sessões, através de suas bibliotecas proverem uma serie de facilidades inclusive a reutilização de código. Muitos frameworks utilizam a arquitetura MVC, que permite aplicar a divisão da aplicação em camadas sendo elas modelo de dados, negocio e a interface sendo uma boa pratica de mercado já que gera reutilização e modularização do código e implementação de diversas interfaces.

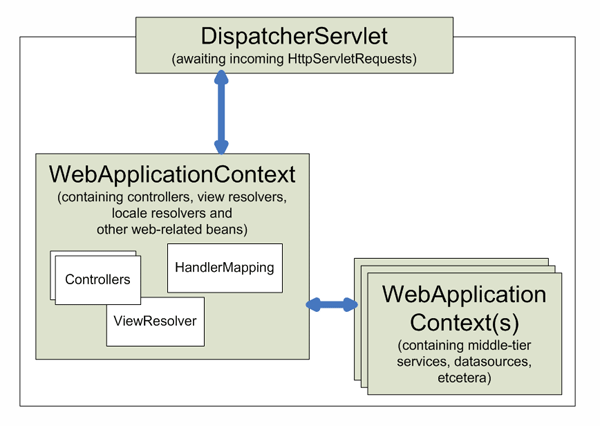
### Spring MVC

Um framework popular e open source que auxilia no desenvolvimento de aplicações web, contendo um Contêiner, um conjunto de snap in para interface com usuário, um framework que gerencia componentes e transações e persistências Web, baseado no Spring MVC podemos citar Model, Controller e as View no caso jsp

### Dispatche Servlet

O dispatche servlet é um dos componentes do spring, responsável por gerenciar todas as requisições recebidas da aplicação, sendo configurado na estrutura da aplicação através de um arquivo xml, é ele o core do Spring MVC.

* + 1. WebAplicationContext



**Figura 23 – Ilustração do funcionamento dos componentes do spring**

Fonte:

## HandlerMapping

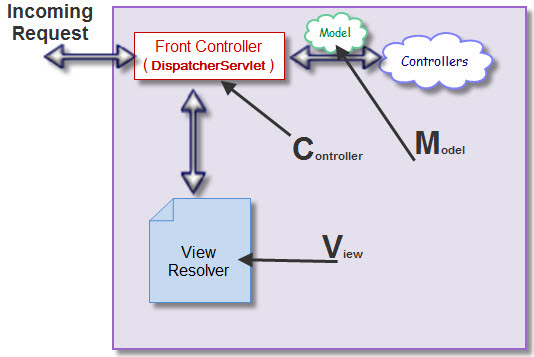
Quando o Dispatche Servlet recebe a requisição ele analisa sua assinatura e encaminha para o HandlerMapping que verifica qual controlador deve ser acionado.

## Controller

A finalidade do controller é receber requisições e retornar uma resposta ao usuário, é ele que encaminha para view o que ela exibira e recebe os parâmetros da view que o usuário requisitou, respondendo o que foi requisitado seja um html, JASON, XML sendo muito útil em formulários.

## ViewResolver

Após a execução do controlador é enviado ao Dispatche Servlet um nome lógico do template de visualização e o conjunto de variáveis (model) que será renderizado como resposta ao usuário (view), assim o ViewResolver retorna ao Dispatche Servlet qual elemento de visualização deverá ser rende rizado de volta ao usuário da requisição.



**Figura 24 – Ilustração do funcionamento do FrontController**

Fonte

## FrontController

Como é mostrado na figura ele fornecer um único ponto de entrada para todas as requisições direcionada para a aplicação como é mostrado na figura. Sendo sua função interpretar os requisitos e direcionar para o componente que irá processa-lo dando retorno ao usuário se houver.

# Conceitos Programáticos de Desenvolvimento

A melhor maneira de se escolher uma ferramenta é conhecendo suas necessidades, com isso é possível escolher a que melhor se encaixa no projeto.

Segundo Csillag (1995), “Uma ferramenta utilizada para reduzir custos de produção de bens e serviços e aumentar o valor para o usuário. Consiste basicamente em identificar as funções de determinado produto avaliá-las e finalmente propor uma forma alternativa de desempenhá-las de maneira mais conveniente do que a conhecida”.

Com as atuais tecnologias de mercado, o desenvolvimento de software está cada vez mais automatizado, tornando os esforços de criação e manutenção cada vez menor, com isso várias dificuldades que os desenvolvedores enfrentavam no passado, hoje estão sendo superadas pelas novas tecnologias que surgem todo dia no mercado.

As implementações de conceitos criaram tecnologias que vão desde tratamento de acesso e armazenamento de dados, até layouts sofisticados e modernos. Com todo esse aparato tecnológico no mercado é necessário ter claro entendimento do conceito das tecnologias antes de implementa-las para podermos identificar se está alinhada com o desenvolvimento e se é uma solução viável para a necessidade do software.

Fazendo analogia a um prédio, para construirmos um software é necessário começarmos pela base, havendo uma base bem estruturada com bons alicerces conseguimos erguer um prédio com seus andares, assim é a base de dados de um software precisa ser criada de forma que possa ser expandida, de fácil manutenção e flexibilidade nas mudanças de requisitos de negócio.

Diante das diversas dificuldades que envolvem os ciclos de vida de software um dos problemas mais propensos a falhas e que traz muitas dificuldades para os desenvolvedores e analistas é a evolução e a definição de acesso à base de dados para aplicações orientadas a objeto, pensando nisso é necessário uma maneira de mapear a base de dados relacional para a linguagem de programação que no caso é orientado a objeto.

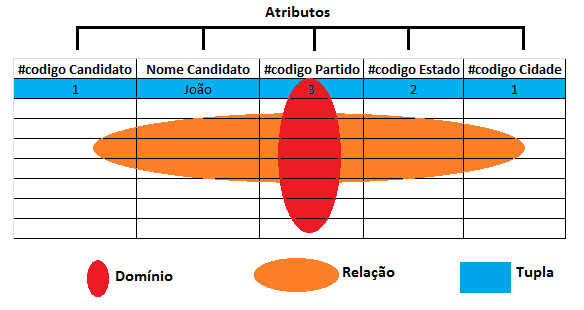
Existe um conceito chamado de mapeamento objeto-relacional, porém para melhor compreensão é necessário algumas definições que serão abordadas a seguir.

## Banco de dados Relacional

Obter informações de forma ágil e confiável é vital para as empresas atualmente, com isso se faz necessário um modelo de dados confiável que já esteja consolidado no mercado, como o modelo de dados relacional. O modelo de dados relacional é caracterizado por organizar os dados em relações, relações que podem ser considerada tabelas, colunas como atributos da relação, as linhas representam as *tuplas* ou elementos da relação e o domínio é os tipos de dados que descrevem os tipos de valores que pode ser representado em cada coluna. Segundo Elmasri e Navathe (2005), o modelo relacional representa o banco de dados como uma coleção de relações.

Um conceito muito importante em banco de dados relacional são os atributos chaves, é com isso que ele pode diferenciar as *tuplas* uma das outras e manter relações entre múltiplas tabelas, também existe a possibilidade de acelerar o acesso às consultas usando índices.

Segundo Kaufeld (1996), o modelo de banco de dados relacional possui a capacidade de lidar com grandes volumes de informações, eliminando dados redundantes.



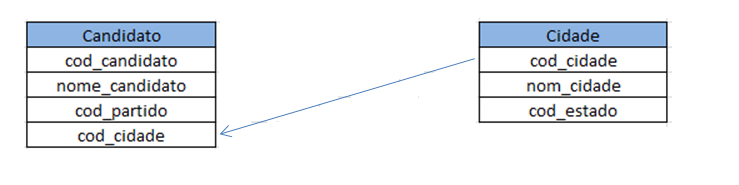
**Figura 10 - Representação dos conceitos do modelo relacional.**

Fonte: Elaborado pelo autor.

Um banco de dados relacional é composto por inúmeras tabelas que diariamente são alimentadas por sistemas em funcionamento, uma massa de dados que a empresa manuseia para obter informações a partir de dados e elaborar relatórios, gráficos e etc. Dependendo do numero de informações que se deseja obter uma empresa pode ter vários bancos de dados em diferentes domínios de aplicações. (Date, 2000).

## Chaves primarias e estrangeiras

Um banco de dados relacional caracteriza-se pela utilização de chaves sendo os tipos: identidade e referencial é isso que mantem a restrição de integridade, havendo entre as tabelas relacionamentos através de um campo identificador como chave. Uma chave primária pode ser representada por uma ou mais colunas, sendo que sua principal característica é que ela não pode ter valores duplicados dentro de uma tabela. Uma chave estrangeira pode ser representada por uma ou mais colunas ela se se caracteriza por ter o mesmo valor da chave primaria de outra tabela, sendo a chave estrangeira que definir o relacionamento entre as tabelas. Uma chave primaria é escolhida a partir de um conjunto de candidatas de uma entidade que por definição tenha valor exclusivo das tuplas e não nulo. (Date, 2000).



**Figura 11 - Representação dos conceitos de chave estrangeira**

Fonte: Elaborado pelo autor.

Como mostra a Figura 11 à entidade chamada Candidato conte a chave primaria “*cod\_candidato*” e as chaves estrangeiras “*cod\_partido*” e ”*cod\_cidade*”, já a outra entidade contem a chave primaria “*cod\_cidade*” e a chave estrangeira “*cod\_estado*”, nessa situação a chave estrangeira “*cod\_cidade*” da entidade Candidato contem o mesmo valor da chave primaria da entidade Cidade formando assim um relacionamento entre as duas entidades, o mesmo se aplica para as chaves estrangeiras “*cod\_partido*” da entidade Candidato e a “*cod\_estado*” da entidade Cidade.

## Integridade

O conceito de integridade diz respeito à credibilidade de uma ou mais informações em banco de dados. Ela é necessária para que o banco possa tratar os diferentes tipos de dados de forma especifica minimizando assim a probabilidade de erro. As regras de restrição de banco de dados é que determina que tipo de dados pode ser armazenado em determinada coluna de uma tabela. (Date, 2000).

### Integridade de Entidade

É especificado em relações individuais, nenhum valor de chave primaria pode ser nulo, pois sendo nulo não saberíamos distingui-las. Um exemplo é se uma ou mais *tuplas* tivesse valores nulos em suas chaves primarias não saberíamos distingui-las umas das outras. (Date, 2000).

### Integridade Referencial

A restrição de integridade referencial é especificada entre duas relações e é utilizada para manter a consistência entre *tuplas* destas relações. Informalmente, a restrição de integridade referencial declara que uma *tupla* em uma relação que se refere à outra relação deve se referir a uma *tupla* existente naquela relação (Elmasri e Navathe, 2005).

## Histórico da Linguagem Java

Java surgiu na década de 90, oriundo da linguagem *Oak* ( que foi criada para o projeto \*7 da Sun), Java foi a adaptação do Oak para internet e foi lançado em 1995 por James Gosling e sua equipe. A linguagem Java foi amplamente adotada após seu lançamento e em 2004 já atingia o número de 3 milhões de desenvolvedores em todo o mundo. CAELUM (2005)

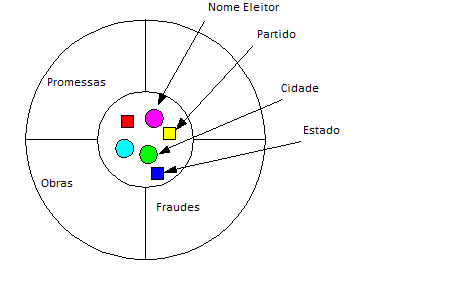
Java se destaca, principalmente, pela sua interoperabilidade, por ser compilada para um byte code e não para um código nativo, como a grande maioria das linguagens, Java pode ser usado em qualquer dispositivo que executa uma máquina virtual Java. CAELUM (2005)

Em 1999 foi criado o Java EE (Java *Enterprise* *Edition*), ele consiste em uma série e especificações que demonstram como deve ser implementado um *software* que utilize uma infraestrutura complexa, composta por: persistência em base de dados, transação, acesso remoto, web *services*, gerenciamento de threads, gerenciamento de conexões HTTP, cache de objetos, balanceamento de carga, entre outros(Arun Gupta,2013). A ideia principal do Java EE é entregar pronta toda essa infraestrutura para o desenvolvedor, possibilitando assim que o foco seja voltado para a implementação das regras de negócio do software. CAELUM (2005)

## Conceito de orientação a objetos

É baseado na composição e interação entre diversas unidades do software chamados de objeto, que possuem responsabilidades e funções e ao ser implementado tenta modelar os elementos no sistema na mesma forma que são percebidos no mundo real.

Segundo Campos (2009), o paradigma de orientação a objeto tenta desenhar os elementos em um sistema da mesma maneira que são percebidos no mundo real, desta forma os objetos possuem responsabilidades e funções para atingirem um objetivo comum. Como vantagens a orientação objeto é mais fácil de pensar do que em procedures e funções, pois os objetos estão mais próximos do mundo real e o reaproveitamento do código com componentes bem modularizados com funções bem definidas servindo a propósitos claros e delimitados, podem ser reaproveitados minimizando o tamanho do código.



**Figura 12 - Representação do conceito Objeto**

Fonte: Elaborado pelo autor.

No exemplo da Figura 12 temos dois círculos um dentro do outro, o circulo interno representa os atributos, o circulo externo representa as ações ou métodos do objeto.

## Eclipse IDE

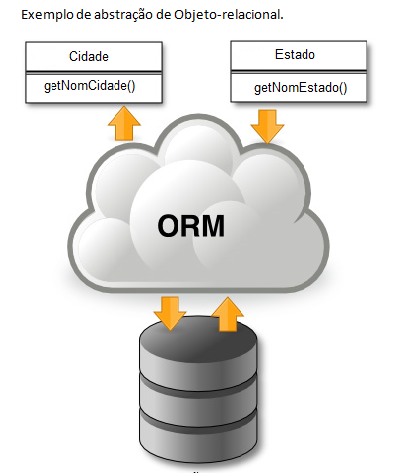
IDE (*Integrated development environment*) são softwares que auxiliam no desenvolvimento de aplicações, eles possuem uma série de ferramentas que possibilitam e agilizam o desenvolvimento de aplicações, tais como: compilados, editor, linker, depurados, modelagem, geração de código e refatoração. As principais IDE’s que dão suporte a linguagem Java atualmente são: Eclipse, NetBeans e InteliJ. O Eclipse é largamente utilizado, pois além de Java possui suporte a outras linguagens, como PHP, Pearl, Ruby, Groove e outras. Possui uma série de ferramentas padronizadas em suas versões, um dos grandes diferenciais da IDE Eclipse, são os vários plug-ins que ela possui, para uma infinidade de funções, com essa condição, cada desenvolvedor pode customizar a IDE de acordo com as suas necessidades, o que ajuda tanto em termos de organização, como também no consumo de recursos de hardware.

## Mapeamento Objeto-relacional

O mapeamento objeto relacional (ORM) com o intuito de atingir as vantagens competitivas durante o desenvolvimento de software e evitando o trabalhoso acesso aos dados da forma tradicional, caracteriza-se por ser uma técnica que abstrai a solução de uma forma que as relações passam a ser representadas pelas classes, sendo as “*tuplas”* desta tabela originadas de cada instancia da classe em questão e utilizando banco de dados relacionais reduzindo os impedimentos da programação orientada a objeto. Conforme mostra na figura 13, o ORM é que faz a tradução das Classes da linguagem OO em tabelas da base de dados relacional, tendo um papel fundamental na interação de conceitos distintos como a orientação a objeto e o banco de dados relacional.

Softwares desenvolvidos com ORM tendem a serem econômicos no seu desenvolvimento, melhores no desempenho e fácil manutenção e evolução.

Exemplo de abstração de Objeto-relacional.



**Figura 13 - Representação dos conceitos ORM**

Fonte: http://www.edzynda.com/use-laravels-eloquent-orm-outside-of-laravel/

No mercado existem vários produtos que são utilizados hoje que usam o conceito de ORM que *são Toplink, Ibatis* *e Hibernate*.

## Hibernate Tools

Uma ferramenta com o objetivo de facilitar o mapeamento das classes e a criação de suas tabelas, a partir de um arquivo de configuração. Permitindo ao programador gerar e visualizar as classes mapeadas em suas respectivas tabelas do banco.

Havendo diversos pluguins no mercado compatíveis com diversas IDEs e sua utilização já consolidada no mercado, traz evidencias de ser uma ferramenta confiável e com risco zero no desenvolvimento do software

## Arquitetura MVC

Para desenvolver uma aplicação, é necessário definir qual a arquitetura será utilizada, existem alguns modelos tradicionais de arquitetura de software, como: cliente-servidor, monolítica, orientada a serviço, orientada a busca e o modelo em três camadas, onde encaixa-se a arquitetura MVC. MVC é o acrônimo de *Model*, *View* e *Controller* (Modelo, visão e controlador). Esta arquitetura visa a separação do código de uma aplicação nessas três camadas, na camada de modelo, temos as informações do domínio que a aplicação opera, um exemplo são as classes POJO (*Plain old java objects*), classes que mapeia todas as entidades com que a aplicação irá interagir. Na camada *View* são exibidas as informações do domínio para o usuário, por exemplo os dados de um usuário do sistema, que estão mapeados em uma tabela na base de dados são exibidos através de uma página web, o código dessa página faz parte da camada de *View.* A camada *Controller* realiza o processamento e responde aos eventos gerados pela *View,* podemos dizer que ela é uma intermediária entre as outras duas camadas, também é nela que são realizadas validações dos dados e filtros impostos pelo usuário.

## Scrum

Atualmente existe uma série de metodologias de desenvolvimento de software, como, RUP, XP e outras. Dentre elas existe uma categoria de metodologias ágeis, sendo o *Scrum* uma das mais conhecidas e utilizadas. No *Scrum* o processo de desenvolvimento é iterativo e incremental, constituído de ciclos de feedback entre equipe e clientes. Uma das grandes características do *Scrum* é a entrega contínua de funcionalidades, o que permite acompanhar bem de perto a evolução do que está sendo desenvolvido. De forma básica, temos os *sprints*, que são as unidades básicas de desenvolvimento, onde são criados incrementos de produto potencialmente entregável, algo como pequenos protótipos. Antes de iniciar cada *sprint*, há um *sprint planning*, que basicamente é uma reunião que vai definir os itens que devem ser entregues ao final desse *spring,* esses itens fazem parte do *Sprint Backlog*, uma lista de atividades que devem ser concluídas nesse spring. Durante os *sprints*, temos as stand-up meetings, reuniões diárias onde a equipe conversa sobre o trabalho que está sendo desenvolvido.

**GitHub**

Em um ambiente de desenvolvimento é preciso se ter o controle das versões de software, assim você tem o controle do código fonte e o que foi agregado ao software ao longo do tempo, lhe permitindo volta a versões anteriores a um bug por exemplo. Com a automatização desse processo não é mais necessária a criação de pastas com nomes das versões para cada nova tarefa compilada, mas sim um repositório centralizado e compactado de forma segura e documentado. O GitHub é um site que foi lançado em fevereiro de 2008, com o intuito de hospedar projetos, atua como um servidor centralizado de versões, sendo um de seus benefícios a disponibilidade, onde um programador a qualquer momento tendo o acesso a internet pode acessar seus fontes, sem a necessidade de configurar servidor e abrir portas no firewall da empresa para obter seus fontes.

## MySQL

Um SGBD (sistema de gerenciamento de banco de dados) que utiliza em sua interface a linguagem SQL. O MySQL se tornou um dos bancos de dados de código aberto mais populares de mercado, fácil de usar, excelente performance, confiável e consistente. Uma das opções mais procuradas da nova geração de aplicações LAMP (Linux, Apache, MySQL, PHP), sendo compatível com mais de 20 plataformas, transparecendo flexibilidade e gerenciamento. (MYSQLBRASIL,2006).

## Padrões de Projeto

### Introdução

Uma forma de obter qualidade e custos baixos no desenvolvimento de projetos com prazos curtos e custos reduzidos é adotando padrões em seu desenvolvimento, tendo isso em vista é necessário conhecer os conceitos e definições dos padrões de projetos para saber como e onde aplicá-los, Tendo a ciência que o seu uso não gerenciado pode tornar um problema para o desenvolvedor.

Padrões de projetos são modelos de soluções que foram documentadas e adaptadas à necessidade do software.

Atualmente softwares com desenvolvimento voltado a orientação a objeto vem se destacando no mercado, por agregar qualidades importantes para os sistemas desenvolvidos, como reaproveitamento do código e a capacidade de ser extensível, mas apenas utilizar orientação objeto não garante que essas qualidades sejam obtidas.

Analisando o cotidiano de desenvolvimento de software é possível identificar que muitas soluções para problemas específicos que foram identificados em projetos anteriores são semelhantes ou até iguais aos de novos projetos, mas por deficiência no processo de documentação ou compreensão do problema, não são aplicados nos novos projetos gerando uma perda de tempo e recursos com o desenvolvimento de uma nova solução para um problema que anteriormente já havia sido encontrado ou resolvido. Uma boa pratica que os projetistas costumam aplicar é não resolver os problemas do zero, buscando reutilizar soluções já aplicadas em problemas anteriores. Então podemos dizer que padrões de projeto são soluções de problemas anteriormente enfrentados, aplicando a partir de um modelo documentado e que pode ser adaptado dependendo da necessidade de sua solução.

### Histórico

Segundo Alexander (1977), “cada padrão descreve um problema no nosso ambiente e o núcleo da sua solução, de tal forma que você possa usar esta solução mais de um milhão de vezes, sem nunca faze- lo da mesma maneira”.

É importante ter uma definição clara do problema e conhecer o padrão de projeto que melhor se encaixa no problema, sabendo qual usar, como usar, porque usar e analisar se ele realmente se enquadra no caso em foco.

A princípio a iniciativa de projetar soluções a partir de algo conhecido e documentado não partiu de projetos de software e sim da arquitetura civil. Christopher Alexander, publicou uma catálogo com mais de 250 soluções de projetos de arquitetura. Em seu catálogo foram descritos os problemas e justificados as soluções comuns enfrentados pelos arquitetos no cotidiano. Sua analise consistia em diminuir o foco do problema procurando estruturas que resolvam problemas similares, assim ele podia entender as similaridades entre projetos distintos, similaridades essas que ele nomeou como padrões. Com o tempo a comunidade de software aderiu a suas ideias que obteve atenção na conferencia sobre orientação a objetos, em 1995 Erich Gama, Richard Helm, Ralph Johnson, John Vlissides publicaram o livro Design Patterns G4 que ganhou sua versão na língua portuguesa chamada “Padrões de Projeto”- Soluções reutilizáveis de software orientado a objetos, um catalogo com 23 padrões de projetos para soluções de problema de software contendo três categorias padrões de criação, estruturais e comportamento.

### Tipos de padrões de projeto

O padrão deve ser adaptado conforme a necessidade do projeto. Observa-se a mesma solução diversas vezes, para problemas com similaridades.

### DAO

Um dos problemas enfrentado por desenvolvedores é organizar no código do sistema os comandos SQL, os problemas mais comuns são duplicidade em query que possuem a mesma finalidade. Outro problema comum é query com finalidades iguais localizada no código em lugares distintos, atendendo dois objetos diferentes. Situações com essa onde as querys estão dispersas e desordenadas torna o código pouco compreensível, difícil manutenção e evolução para os programadores.

A melhor prática quando se desenvolve um sistema com banco de dados é definir de forma eficaz e organizada onde agrupar os comandos *sql*. A partir disso uma solução seria agrupar todo o código *sql* em um objeto para que a sua manutenção permaneça em um único ponto (conforme é mostrado na figura 14), é essa solução que se baseia o padrão de projeto de persistência de dados *Data Access Object* (DAO). Segundo Aéce (2005), isso leva a uma produtividade muito grande e facilidades em manutenções futuras.

Genericamente todo o sistema (conjunto de objetos) é composto por relações com a base de dados que são Salvar, Atualizar, Apagar e Listar todos se for o caso, sabendo disso e seguindo a solução proposta pelo DAO fazemos o agrupamento dos comando SQL em classes DAO de acordo com cada objeto, assim as Classes DAO serão o ponto único de manutenção dos comandos *sql.* Segundo, Aéce (2005), é um padrão de projeto que abstrai e encapsula os mecanismos de acesso aos dados.

Segundo Sun (2007), o DAO esconde completamente os detalhes da execução da origem dos dados de seus clientes, conforme mostra.

# 

**Figura 14 - Representação do conceito do padrão de projeto DAO.**

Fonte: <http://spiproductsplatformsolution.wordpress.com/2012/10/30/hibernate-generic-data-access-object/>

A grande vantagem em desenvolver com o padrão de projeto DAO é que há uma separação rigorosa entre as partes de persistência e regras de negócio, podendo assim evoluir frequentemente independente uma da outra. Quando se altera a lógica da regra de negócio, espera-se apenas que se implemente uma interface e quando altera-se a regra de persistência, não se altera a lógica da regra de negócio desde que a interface entre ambas não seja modificada.

#### Segurança nos processos de desenvolvimento

A segurança que envolve o acesso aos dados muitas vezes é negligenciada no desenvolvimento de uma aplicação, O sql Injection é uma técnica de ataque baseado em inserção e manipulação de consultas geradas pelo software, que são encaminhada diretamente para a base de dado.(Cristiano, 2012)

Uma falha visível que pode gerar esse tipo de ataque é o não tratamento do conteúdo inserido pelo usuário nos campos específicos.

Exemplo:

var sql= "SELECT cod, nome, sobrenome FROM politico WHERE nome ='" + jo'sé + "' AND sobrenome = '" + serra + "'";

Uma vez que o usuário pode inserir em um campo tipo texto qualquer tipo de caractere, está aberta a vulnerabilidade que o atacante precisa. No exemplo anterior resultaria em um erro de sintaxe, pois a inserção das aspas simples entre 'jo' e 'sé' fecha as aspas simples da consulta, através do problema exposto o atacante já consegue manipular os dados de inserção de forma a gerar um comportamento na base inesperado sabendo sobre a concatenação interna.

var sql = "SELECT \* FROM usuario WHERE nome = '" + username + "' and password = '" + password + "'";

Ao acessar a tela de autenticação, como não a nenhuma validação dos tipos de dados de inserção, apenas a concatenação de string o usuário mal intencionado utiliza um artificio.

Pressupondo que conheça o nome de um usuário valido:

Usuario: admin'--

Password:

o sistema vai tratar

var sql = "SELECT \* FROM usuarios WHERE nome = '" + username"

O sql irá tratar o ‘—‘ como comentário, não exigindo o Password e liberando o acesso ao software.

O usuário mal intencionado pressupondo que não conheça o nome de um usuário valido entra com

Usuário: or 1=1

Password:

O sistema vai tratar

var sql = "SELECT \* FROM usuario WHERE nome = "''" or '"+ 1=1 + "'"

O sistema retornará todos os registros da tabela e liberando o acesso ao software.

Nos casos mais extremos o usuário mal intencionado pode fazer o seguinte em uma consulta:

Nome = jo'; DROP TABLE Usuarios; --

Sobrenome = silva

ou seja

SELECT cod, nome, sobrenome FROM Usuarios WHERE nome = 'jo'; DROP TABLE Usuarios;

Deletar todos os usuários da tabela Usuários

Para um tipo de vulnerabilidades como essa são necessários tratamento de inserção de dados e a utilização do Design Partner DAO também é recomendado.

### Singleton

A técnica de aplicar soluções reutilizáveis em softwares através de padrões de projeto resulta objetos flexíveis, sustentáveis, redução na complexidade do processo de design e ganho de tempo.

Em programação existem dois escopos de variáveis local e global, variáveis locais tem um mecanismo de escopo local de memória não compartilhada e alocam apenas recursos locais.

Diferentemente variáveis globais que possuem um escopo acessível a todo um programa, seu conteúdo pode ser potencialmente modificado de qualquer local, e qualquer parte de um código pode depender dela trazendo desvantagens em seu uso, consumindo por vez diversos recursos que não estão sendo necessariamente utilizados, da mesma forma pode-se considerar um objeto de conexão de base de dados, instanciando várias conexões para vários objetos.

O padrão de projeto *Singleton* é uma solução para esse tipo de problema tendo como definição garantir que uma classe tenha somente uma instância e forneça um único ponto global de acesso para a mesma, com isso não há o risco de haver mais de uma instância para um único objeto, então a classe gerencia sua própria instancia além de proibir que qualquer outra classe crie uma instância dela, tendo que passar pela classe que gerou o objeto obrigatoriamente. A necessidade de se implementar *singleton* geralmente é quando é preciso globalizar certos aspectos do sistema.

**Vantagem**

A vantagem que o padrão traz ao ser utilizado é quando existe necessidade de criar uma única instância do objeto e ao mesmo tempo compartilhar seus recursos com os demais.

### Delegate

O padrão de projeto *delegate* é um tipo de variável que armazena o endereço de um método. Sua utilidade se baseia no uso de eventos, quando uma classe dispara uma mensagem em tempo de execução, o *delegate* tem o papel de armazenar o endereço de memória onde está o método.

Existe a necessidade de alguns requisitos para uma variável apontar para um método com assinatura declarada no *delegate*.

* Uma instancia desta Classe;
* A mesma assinatura do método, sendo que dever ser do tipo *void* e não receber nenhum parâmetro;
* O método devera ser declarado como *static*.

Contendo uma interface pública, as classes do *delegate* permitem inicializar, adicionar, remover e invocar *delegates*.

Existem duas maneiras para invocarmos um *delegate*.

* () Operador da chamada;
* Método *Invoke*.

### Strategy

Em sistemas de ambientes corporativos existem lógicas condicionais que são estruturas complexas que tender a se expandir e com o tempo se tornar cada vez mais sofisticadas com difícil manuseio. *Strategy* é uma solução que auxiliar a gestão da complexidade que existe na lógica condicional, como é mostrado na figura 15 sugerindo que se desenvolva uma família de classe para cada variação do código e que se forneça para cada classe hospedeira uma instancia de *stategy* para a qual ela delegará em tempo de execução, assim *strategy* nos permite configurar uma classe com um de vários comportamentos, utilizando o conceito de OO chamado de composição.

Exemplo:

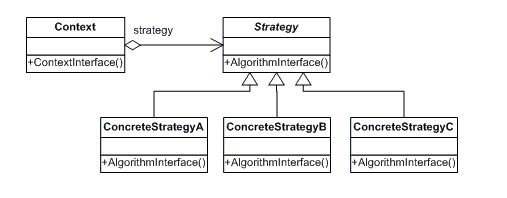


Figura 15: Representação do Conceito de Strategy.

**Strategy** - Interface comum para todas as classes (variações concretas) que definem os diversos comportamentos esperados;

**ConcreteStrategy** - Classes que implementam os algoritmos que devem atender a cada contexto;

**Context** - Classe onde os objetos ConcreteStrategy serão instanciados;

**Vantagens**

Troca de um algoritmo em real time.

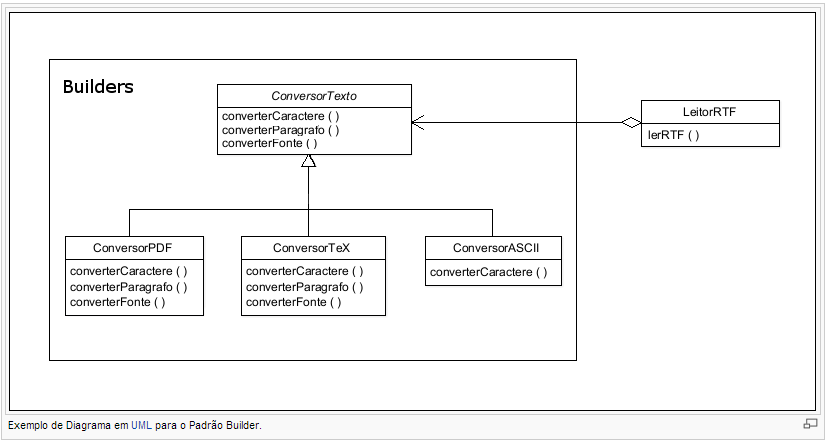
Opção de implementações distintas com a mesma ação;

Classes simples no fluxo de um algoritmo para uma estrutura;

Minimização ou exclusão da lógica de condição limpando ainda mais o código;

### Builder

O Padrão de Projeto Builder é a separação da construção de objetos complexos da sua representação, abstraindo a criação de objetos com particularidades na sua interface, favorecendo a manutenção, uma arquitetura integra e minimizando a complexidade do código. Um objeto se torna complexo quando sua instância necessita de vários métodos para ser feita, não sendo eficiente ter toda lógica de criação em seu construtor, e nem distribuir a lógica em vários métodos adicionais.



**Figura 16 - Representação do Conceito de Builder.**

Fonte: <http://pt.wikipedia.org/wiki/Builder>

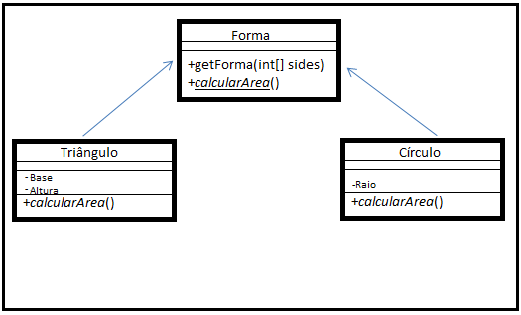
Com builder é possível separar a construção do objeto de sua representação podendo assim construir o objeto por etapas conforme é mostrado na Figura 16.

**Vantagem**

A Construção de objetos complexos em simples passos, tornando o processo de criação genérico para poder criar diferentes representações.

### Abstract Factory

O conceito do padrão de projeto Abstract factory é o de utilizar uma interface genérica de determinado objeto, sem especificar a classe concreta do mesmo, assim sua criação se torna dinâmica, sendo desconhecida a classe de implementação, apenas sua interface conforme é ilustrado na figura 17.



**Figura 17 - Representação do Conceito de Abstract Factory.**

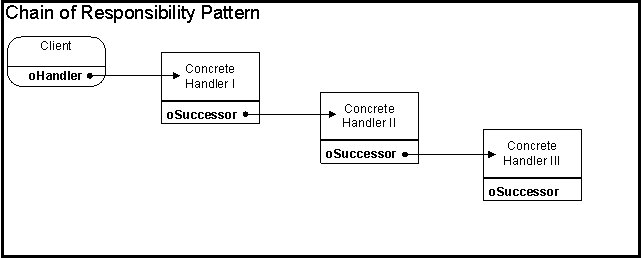
Fonte: Elaborado pelo autor

Segundo GAMMA (1995), Definir uma interface para criar um objeto, mas deixar as subclasses decidirem que classe instanciar. O Factory Method permite adiar a instanciação para subclasses.

Seu objetivo está em várias classes executarem o mesmo tipo de operação ou método, retornarem o mesmo tipo abstrato, mas, internamente instanciar classes diferentes das que o implementam.

### Chain of Responsibility

Com a ideia de evitar o acoplamento entre remetente de uma requisição e o receptor, o padrão de projeto *Chain of Responsability* constrói uma cadeia de objetos receptores e passa a requisição entre eles até que algum receptor tenha uma lógica compatível para trata-lo, assim ele elimina as estruturas de decisão do código. Uma vez que se tem essa estrutura, minimiza-se a dependência do remetente a um só receptor, dando oportunidade para mais de um objeto receptor tratar a requisição do remetente.



**Figura 18 - Representação de Conceito Chain of Responsability.**

Fonte: <http://weblogs.foxite.com/andykramek/2006/12/24/design-patterns-the-chain-of-responsibility/>

# Especificação do Projeto

## Levantamento de Requisitos

Entender o que o usuário necessita ou os envolvidos no negócio, antes do desenvolvimento é fundamental para as atividades de produção do software. O objetivo dos processos é garantir a qualidade do desenvolvimento do software, atendendo o que o usuário precisa, dentro do orçamento e prazos acordados. O IEEE2 definiu o levantamento de requisito de software como parte integra da Engenharia de Requisitos, através desse processo que os desenvolvedores e usuários desvendam, revisam e esclarecem as necessidades dos usuários e identifiquem as restrições nas etapas desenvolvimento e sistema. (THAYER, 1997).

## Descrição geral do software

Para o desenvolvimento de um software é necessário que seja mapeado os requisitos, levando em consideração os pontos chaves em uma aplicação como um Portal. Nesse documento abordaremos detalhadamente os principais pontos de especificação do software tendo como diretrizes os seguintes tópicos:

* Descrição geral do Portal;
* Requisitos específicos;
* Manual de orientação de uso;
* Políticas do Portal.

**Nome do Projeto:**

Portal da vida Política

**Nome do Produto:**

Pvp 1.0

### Descrição Resumida do Serviço

Uma solução voltada a organizar e centralizar informações sobre a vida política, dando ao eleitor a oportunidade de publicar noticias que são de interesse de todos os demais eleitores, tendo como objetivo, maior transparência na vida publica do politico, dando a possibilidade do eleitor, ter uma escolha consciente do voto.

### Objetivos deste documento

Documentar os requisitos dentro do escopo de planejamento detalhando as Definições de papeis dos atores, os casos de uso, requisitos funcionais e não funcionais e as restrições do sistema. Auxiliando-nos na base para todo o ciclo de desenvolvimento do projeto.

## Definição de Papéis

### Usuário

O usuário é o membro (colaborador) do portal, é ele que publicará a maioria das notícias, poderá debater sobre as matérias tendo como regra o respeito mutuo e o linguajar apropriado, respeitando a opinião dos demais membros. O conteúdo antes de publicado será avaliado pelos moderadores para garantir que a informação possui uma fonte confiável. O usuário terá direito a disponibilizar noticias, poderá sofrer bloqueios temporários de visualização e até ser banido caso o moderador julgar necessário.

### Moderador

Um moderador por definição é alguém que gerencia algo, se tratando de um sistema (portal) o moderador irá controlar o comportamento dos usuários, para isso terá funcionalidades diferenciadas contendo responsabilidades e privilégios, é ele que irá gerir o comportamento dos membros, tendo um papel fundamental na organização e cumprimento das normas do portal.

O moderador poderá a qualquer momento punir os usuários que não tenham comportamento adequado ou descumpram as regras, tendo como medida penalidades como suspensão por um determinado tempo ou exclusão do portal, permissões essas que o moderador deverá ter para que as políticas de conduta sejam cumpridas pelos membros do portal.

O moderador terá que gerenciar os membros e os possíveis conflitos entre eles, supervisionar o que é postado para identificar se existe algo que está sendo descumprido conforme as regras impostas, identificar publicação de matérias que tenham como fim propaganda eleitoral, editar, mover e apagar matérias descabidas ou fora da ideologia do portal. O moderador terá como perfil uma pessoa equilibrada capaz de manter a ordem apaziguando brigas, procurando manter um bom relacionamento com os usuários, ser ético e ter bom censo em suas ações. É necessário que o moderador tenha disponibilidade (estar on-line) para esclarecer as dúvidas que surjam por parte dos membros, não sendo necessário saber todas as respostas, mas sim as regras e normas do portal e nem estar disponível vinte quatro horas por dia.

O cargo de moderador trata-se apenas de um membro do portal com boa vontade que se dispôs a dar um pouco de seu tempo ao portal, sendo um trabalho voluntário onde não se recebe nenhum tipo de compensação e sim responsabilidades tendo ele que cumprir determinadas tarefas da função. Para ser qualificado como um moderador o usuário deverá ser membro ativo no portal e como requisito ter um histórico sem punições, procurar ajudar os membros mais novos esclarecendo dúvidas e postar novidades. Deverá o moderador conhecer bem as normas do portal para saber analisar as publicações dos membros, para não cometer erros e se manter como moderador.

### Administrador

O administrador (membro da equipe de desenvolvimento) é quem vai controlar o fluxo das ações dos usuários no portal, interagindo apenas com os moderadores, terá as permissões completa do portal, podendo criar e excluir um moderador como tornar um membro um moderador e vice-versa. Será disponibilizado pelo administrador os termos de uso, onde será detalhado o que o usuário tem direito, infrações e suas penalidades, o mesmo servindo para moderador que está sujeito as mesmas penalidades e perda da função.

### Anônimo

O usuário anônimo como próprio nome se descreve será o usuário que não terá identificação e nenhum vinculo com o portal, tendo a permissão apenas de visualização das noticias, não podendo publicar nem comentar. Sendo um visitante a conhecer o portal, tendo a possibilidade de se cadastrar e se tornar um membro a qualquer momento.

## Escopo do Software

**Missão do software**

Fornecer ao eleitor um espaço no qual possa publicar informações sobre a vida publica do político, podendo trocar informações e ideias com outros eleitores sobre as notícias publicadas.

**Benefícios do Software**

|  |  |
| --- | --- |
| Benefícios | Valor para o eleitor |
| Maiores informações sobre o candidato. | Essencial |
| Alternativa para os meios tradicionais midiáticos. | Fontes diversificadas de informação |
| Informação trazida de eleitores para eleitores | Melhor escolha no voto |

## Requisitos Específicos

### Requisitos Funcionais

**Descrição do caso de uso**

|  |  |  |
| --- | --- | --- |
| **N** | **Nome do Caso de uso** | **Descrição detalhada** |
| 1 | Gerenciamento de usuários | Realiza o gerenciamento de usuários, ou seja, bloqueios, penalidades, exclusões, permissões. |
| 2 | Gerenciamento de Publicações | Realiza o gerenciamento de publicações, ou seja, cadastrar liberar, excluir. |
| 3 | Gerenciamento de Penalidades | Realiza o gerenciamento de penalidades impostas aos usuários, ou seja, tempo de bloqueio, exclusão de usuário, perda de permissões, perda de função. |
| 4 | Identificação de usuário | Valida o usuário no sistema. |
| 5 | Consultas | Consulta as publicações e Comentários feitos. |

|  |  |  |
| --- | --- | --- |
| **N** | **Caso de Uso** | **Descrição** |
| 1 | Cadastro de Usuário | O cadastro inicial é um pré-requisito para o usuário acessar ao portal. |
| 2 | Cadastro de Partido | Antes de publicar a noticia se o partido ainda tiver cadastro deverá ser cadastrado pelo usuário |
| 3 | Cadastro de Politico | Antes de publicar a noticia se o politico ainda não tiver um cadastro deverá ser cadastrado pelo usuário |
| 4 | Cadastro de Estado | Cadastro prévio dos Estados |
| 5 | Cadastro de Cidade | Cadastro prévio de Cidades |
| 6 | Alteração de senha | Usuário pode alterar a senha quando lhe convier |
| 7 | Alteração de foto | Usuário pode alterar foto de perfil quando lhe convier, não podendo ter mais de uma. |
| 8 | Publicar | Cadastro de noticia sobre o Candidato |
| 9 | Visualizar publicações | Visualizar as publicações feitas |
| 10 | Excluir publicações | Excluir publicações que não se enquadram na políticas do portal |
| 11 | Comentar publicação | Insere comentário na publicação |
| 12 | Excluir publicação | Excluir publicação que não se enquadram nas políticas |
| 13 | Alteração de Perfil | Altera o perfil de Usuário para um perfil de Moderador com mais permissões ou vice-versa |
| 14 | Bloqueio Temporário | Bloqueio temporário de usuário por infringir alguma política |
| 15 | Excluir Membro | Exclusão de membro por atingir o numero máximo de penalidades. |

### Requisitos não Funcionais

|  |  |
| --- | --- |
| N | **Descrição** |
| 1 | Não haverá salas de bate-papo. |
| 2 | Não haverá uma indicação se o membro do portal está online ou off-line. |

### Descrição dos tipos de usuário

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Ator** | **Frequência de uso** | **Papel no portal** |
| 1 | Membro | Livre | Colaborador |
| 2 | Moderador | Diário | Gestor dos Colaboradores |
| 3 | Administrador | Quando requisitado | Gestor do Portal |
| 4 | Anônimo | Livre | Interessado |

# Modelo de analise de software

## Regras adotadas para o Portal

# RN01

|  |  |
| --- | --- |
| Nome | Operadores prévios |
| Descrição | Pelo menos 1 usuário administrador e 1 moderador, deverão estar previamente cadastrados no portal. |

# RN02

|  |  |
| --- | --- |
| Nome | Funcionalidades de exclusão, alteração e cadastros do sistema. |
| Descrição | Somente Administradores e Moderadores poderão excluir ou modificar dados referentes aos efetuados no Portal. |

# RN03

|  |  |
| --- | --- |
| Nome | Requisito de senhas do Portal. |
| **Descrição** | A senha deve ter no mínimo 8 dígitos alfanuméricos. |

# RN04

|  |  |
| --- | --- |
| Nome | Privilégios de acesso a áreas do portal. |
| Descrição | O portal deverá existir privilégios, de acordo com a função de cada usuário. |

## Restrições

|  |  |  |  |
| --- | --- | --- | --- |
| N | **Restrição** | | **Descrição** |
| **1** | Segurança | | O portal deverá restringir o acesso através de senhas individuais, para cada usuário. |
| **2** | Funcionalidade | O portal deve compor as funcionalidades de acordo com as especificações descritas. | |
| **3** | Permissões | As permissões devem se enquadrar de acordo com o perfil do usuário, sem exceções. | |

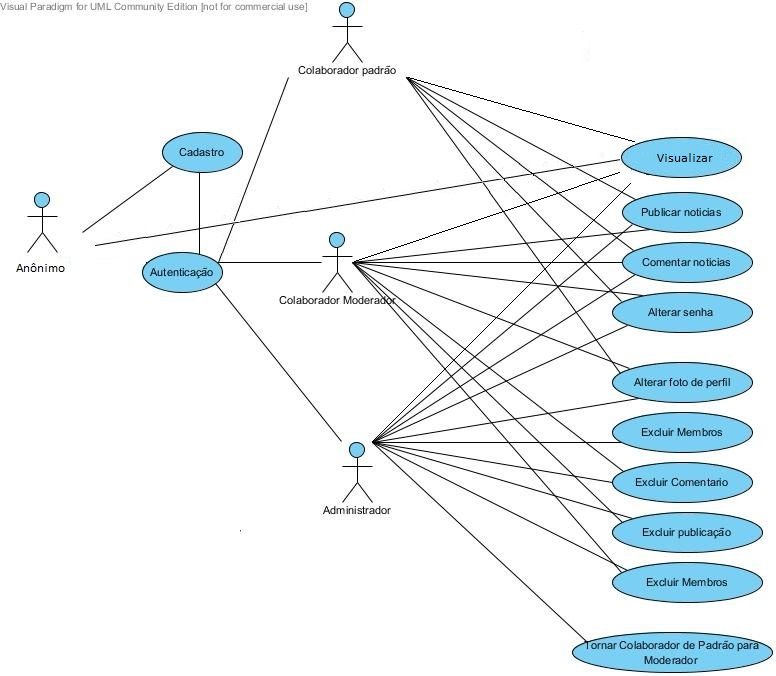
## Descrição geral de funcionamento

### Diagrama de contexto

Segundo Furlan, o diagrama de caso de uso é um conjunto de casos incluídos por limitações de domínio, participação, interação e associações entre atores assim como generalizações entre casos de uso (FURLAN, 1998).

A modelagem de casos de foi utilizada para ilustrar as necessidades de desenvolvimento do sistema e trazer ao leitor um entendimento do que foi feito.

A Figura 25 exibe uma visão geral do Portal, com suas conexões e interações com o usuário. Conexões são produzidas por eventos, que são trocas de mensagens entre o sistema e o usuário, através das conexões podemos estudar os eventos que estão integrados ao sistema.



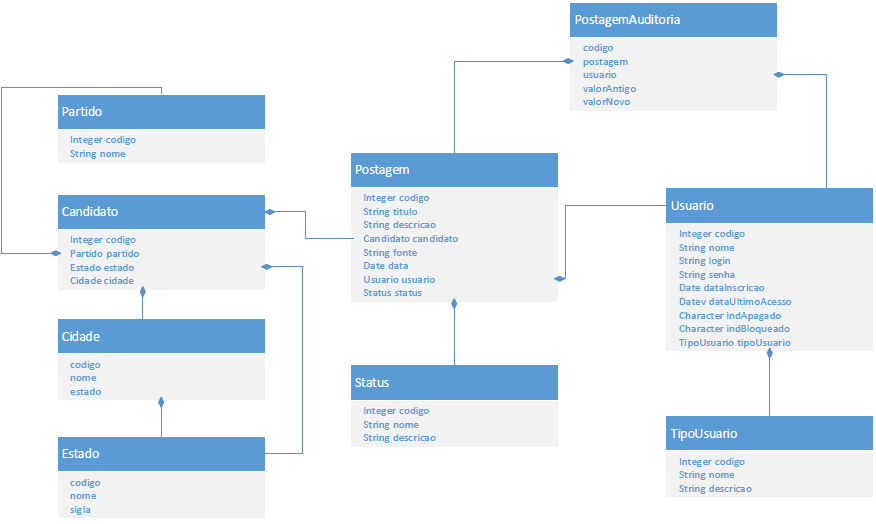
**Figura 27 - Ilustração global de atores e casos de uso**

Fonte: Elaborado pelo autor

### Diagrama de Classe

A figura 2 mostra a estrutura do portal, exibe um conjunto de classes junto a seus atributos e seus respectivos relacionamentos, ilustrando a idéia inicial do portal.

Contendo a classe Usuário, que está relacionada à classe que define suas permissões (TipoUsuario), a Classe Candidato que se relaciona a classe Partido, as classes Cidade e Estado que se relaciona a Candidato, tendo como ultimo relacionamento a classe Postagem que se relaciona com a classe Usuário e a classe Status que determina se a noticia está conforme as políticas do portal.

******

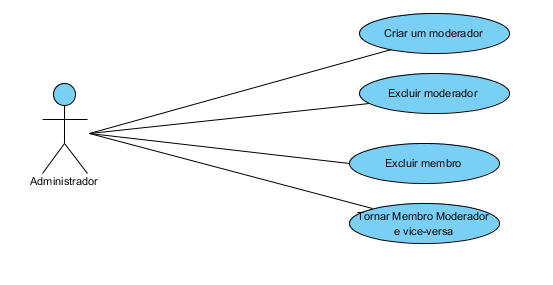
# Figura 2 - Diagrama de Classes representando as principais entidades do Portal.

Fonte: Elaborado pelo autor.

# 

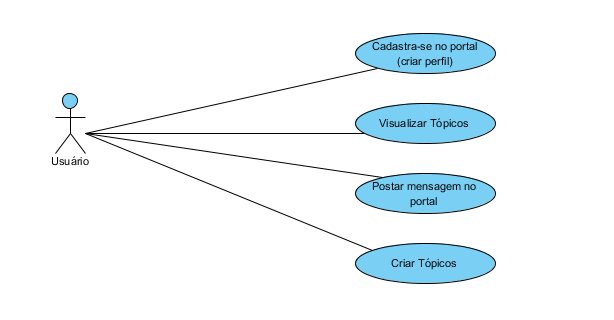
### Diagrama de Caso de Uso

Abaixo os casos de uso que se seguem representam as permissões e papeis que os Usuários têm dentro do portal.



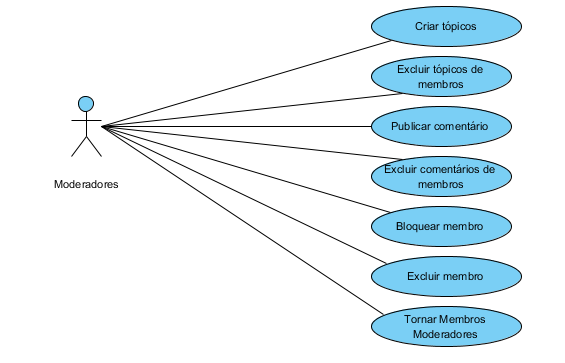
**Figura 3 - Casos de uso do usuário Administrador**

Fonte: Elaborado pelo Autor



**Figura 4 – Caso de uso do usuario Colaborador**

Fonte: Elaborado pelo Autor



# Figura 5 - Caso de uso do usuário Moderador

Fonte: Elaborado pelo Autor

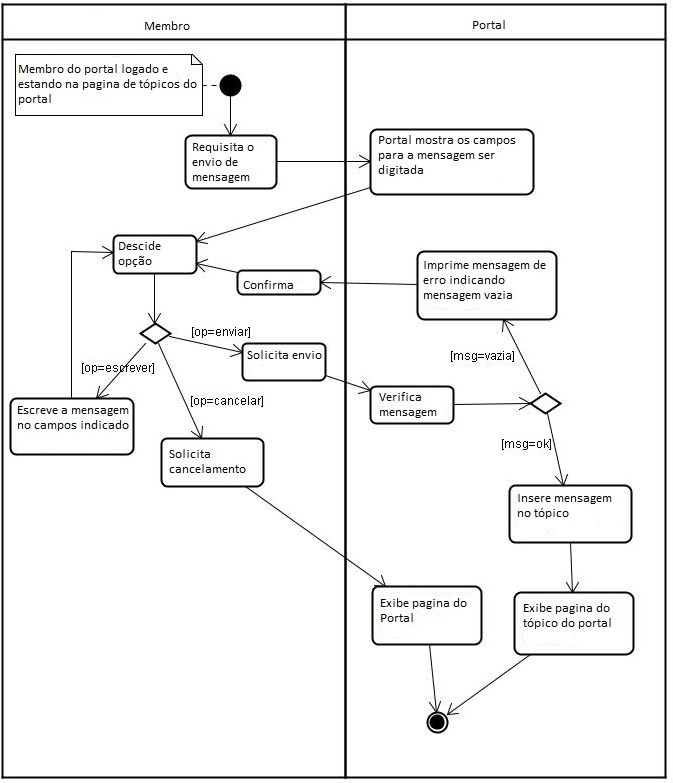


# Figura 5 - Caso de uso do usuário Anônimo

Fonte: Elaborado pelo Autor

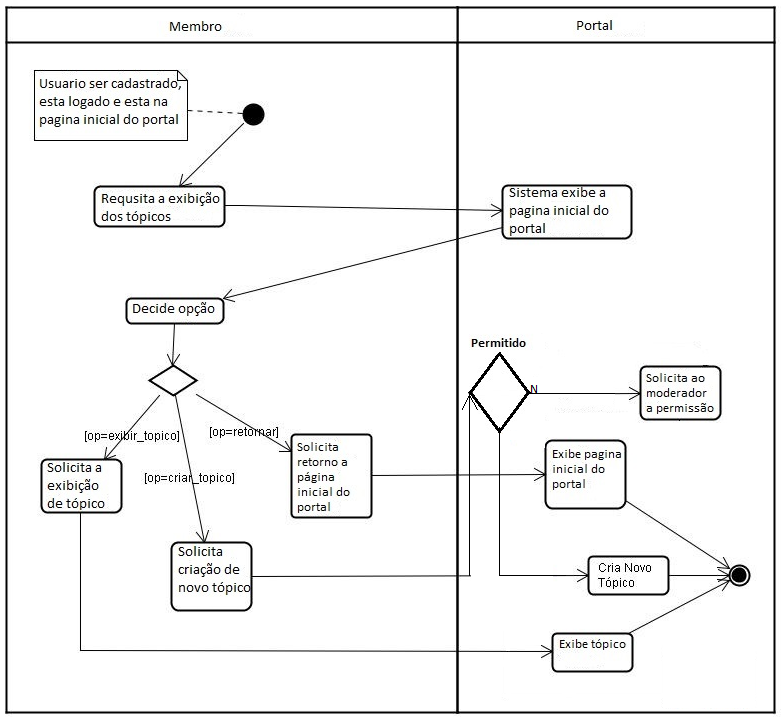
### Diagrama de Atividade

Abaixo segue as trocas de mensagens que o usuário e o portal terão ao longo de uma ação



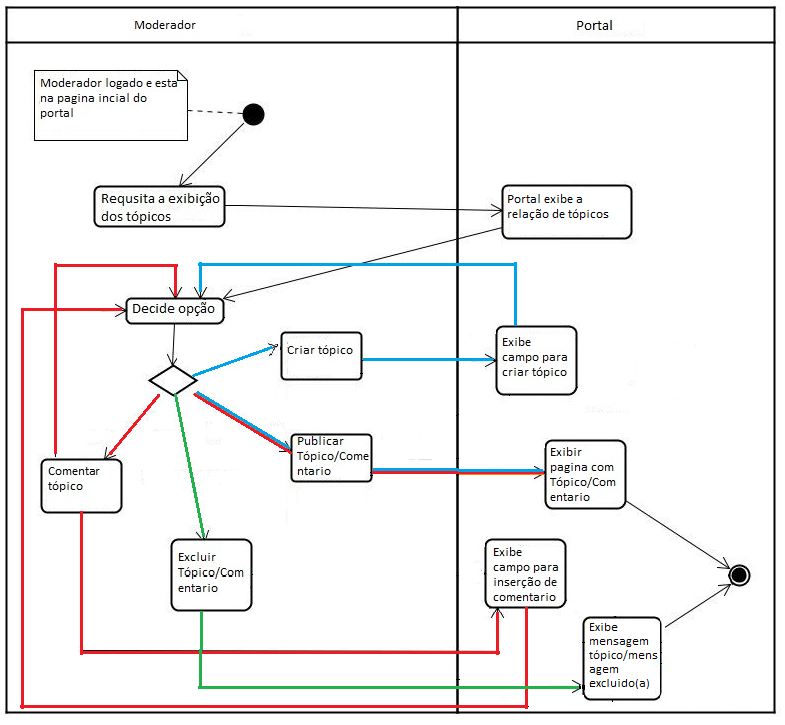
**Figura 6 - Envio de mensagem**

Fonte: Elaborado pelo Autor



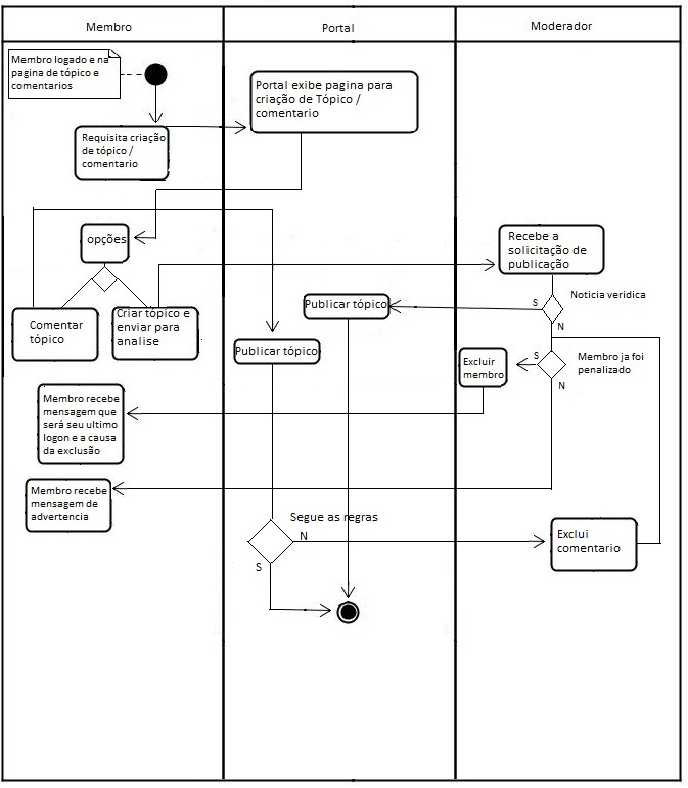
**Figura 7 - Inserção de novo tópico.**

Fonte: Elaborado pelo Autor



**Figura 8 - Interação de Moderador com tópico e comentário.**

Fonte: Elaborado pelo Autor



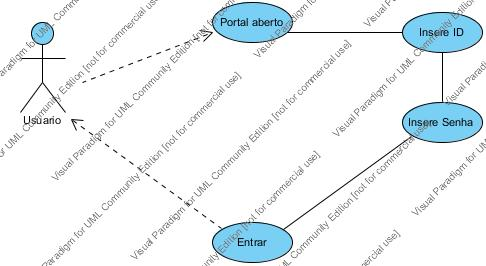
## Figura 9 - Exclusão de usuários.

Fonte: Elaborado pelo Autor

### Telas do Portal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Nome da Tela | Ator | Caso de uso | Descrição |
| 1 | Tela de autenticação | Usuário, Moderador e administrador | Identificação | Tela de login para a autenticação do usuário. |
| 2 | Cadastro de usuário | Usuário | Cadastro | Cadastra os usuários no portal. |
| 3 | Perfil do Usuário | Moderador | Cadastro | Cadastra e alterar dados dos usuários. |
| 4 | Alteração  de senha | Usuário e Moderador | Cadastro | Alteração de senha. |
| 5 | Alteração  de foto | Usuário e Moderador | Cadastro | Alteração de foto de perfil. |
| 6 | Publicar | Usuário e Moderado | Upload | Publicar informações no portal. |
| 7 | Visualizar publicações | Usuário e Moderador | Consulta | Visualizar as publicações realizadas. |
| 8 | Pesquisa | Usuário e Moderador | Consulta | Pesquisar por publicações realizadas. |
| 9 | Gerenciamento Penal | Moderador e Administrador | Cadastro | Aplicar, alterar, e liberar penalidades aos usuários. |
| 10 | Tela Principal | Usuário Moderador e Administrador | Consulta | Tela mostra os itens do portal |

### Especificação do Cenário

****

**Figura 28 – Caso de uso logon do usuário no portal**

Fonte: Elaborado pelo autor

Ator primário: Usuário, Moderador, Administrador.

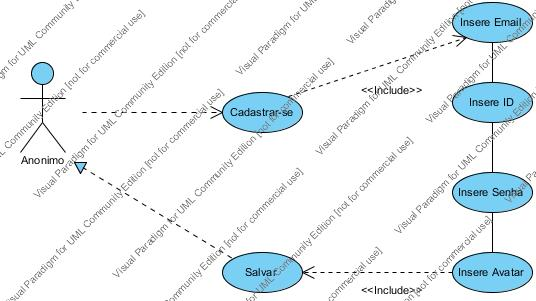
Pré-condição: O site do Portal deve estar aberto

Fluxo Principal

1 - O usuário insere o ID no campo login;

2- O usuário insere a senha no campo senha;

3- O usuário clica no botão Entrar.

****

**Figura 28 - Caso de Uso Cadastro de Usuário Colaborador**

Fonte: Elaborado pelo Autor

Ator primário: Usuário

Pré-condição: O site do Portal deve estar aberto

Fluxo Principal

1 - O usuário clica no botão ‘Cadastrar-se’;

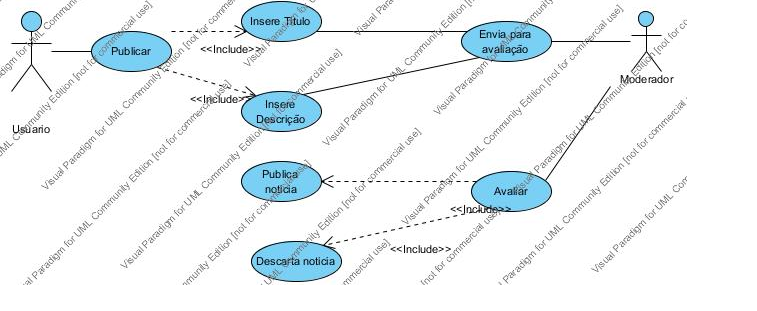
2- É carregada a tela de cadastro de usuário com os campos E-mail, Id e Senha;

3- O usuário insere e-mail;

4- O usuário insere o id;

5- O usuário insere uma senha com no mínimo 8 caracteres;

6- O usuário clica no botão ‘Cadastrar’.



**Figura 29 - Caso de Uso Publicar Noticia**

Fonte: Elaborado pelo Autor

Ator primário: Usuário Moderador

Ator Secundário: Usuário Colaborador

Pré-condição: Deve-se estar logado no Portal

Fluxo Principal:

1 - O colaborador clica no botão ‘Publicar’;

2- É carregada uma tela com os campos de Titulo e Descrição;

3- O colaborador insere titulo (Noticia) e a descrição (Conteúdo);

4- O colaborador clica no botão ‘Submeter Analise’;

5- O moderador verifica o conteúdo e a fonte;

6- O moderador clica no botão ‘Permitir’.

Fluxo Alternativo:

1 - O colaborador clica no botão ‘publicar’;

2- É carregada uma tela com os campos de Titulo e Descrição;

3- O colaborador insere titulo (Noticia) e a descrição (Conteúdo);

4- O colaborador clica no botão ‘Submeter Analise’;

5- O moderador verifica o conteúdo e a fonte;

6- O moderador clica no botão ‘Descartar’.

Realização do Caso de Uso Cadastro de Partido

Desenho caso de uso

Ator primário: Usuário Administrador

Pré-condição: Deve-se estar logado no Portal como Administrador

1. Clicar no botão ‘Partidos’;
2. É exibido um menu com a opção de Cadastro de Partido;
3. Clicar na opção Cadastro de Partido;
4. É carregada uma tela com os campos Nome, Sigla, Presidente;
5. O administrador insere o nome do partido, sua sigla e o Presidente em vigor;
6. O administrador clica no botão Incluir.

Realização do Caso de Uso Listar Partidos

Desenho caso de uso

Ator primário: Usuário, Moderador, Administrador

Pré-condição: Deve-se estar logado no Portal e o Partido não deve contem na relação de partidos.

1. Clicar no botão ‘Partidos’;
2. É exibido um menu com a opção de Listar Partido;
3. Clicar na opção Listar Partidos;
4. É carregada uma tela com uma lista de partidos cadastrados;
5. O administrador insere o nome do partido, sua sigla e o Presidente em vigor;
6. O administrador clica no botão Incluir.

Realização do Caso de Uso Cadastro de Politico

Desenho caso de uso

Ator primário: Usuário, Moderador, Administrador

Pré-condição: Deve-se estar logado no Portal e o Partido não deve contem na relação de partidos.

1. Clicar no botão Políticos;
2. É exibido um menu com a opção de Cadastro de Politico;
3. Clicar na opção Cadastro de Politico;
4. É carregada uma tela com os campos Nome, Partido;
5. O administrador insere o nome do politico e o partido que ele concorre;
6. O administrador clica no botão Incluir.

# Conclusão e etapa futura

## Conclusão

Nesse trabalho de conclusão de curso foram expostos diversos conceitos que foram utilizados ao longo do projeto, desde sua modelagem até a sua conclusão, abordaremos as conclusões tiradas de cada capitulo que foi mostrado ao longo do desenvolvimento do projeto.

No capitulo 2 vimos que a utilização dos processos de desenvolvimento trouxe a organização e controle na criação do software e um escopo amplo nas etapas de desenvolvimento, dando ao desenvolvedor e ao cliente uma visão de custo e viabilidade na construção de um projeto.

No capitulo 3 vimos que a arquitetura de software objetiva particionar o software em componentes menores, através disso concluímos que existe uma simplificação em sua analise e manutenção, mostrando aos desenvolvedores os relacionamento e interações entre os componentes, facilitando seu entendimento, independente da linguagem de programação que foi o software foi construído. Através do SpringFramework, concluímos que sua utilização foi de fundamental importância no desenvolvimento do nosso projeto, pois atendeu com planejado nos pontos de minimizar o acoplamento, reutilização do código e na separação das camadas da aplicação.

No capitulo 4 conhecemos sobre os conceitos de banco de dados relacional e suas propriedades, concluímos que seu funcionamento e sua participação consolidada no mercado de softwares corporativos, nos da garantias de ser apropriado para uso em nosso projeto acadêmico.

Através dos conceitos dos padrões de projetos concluímos que sua utilização evitou uma série de problemas que iriamos enfrentar ao longo do desenvolvimento do portal e também nos ajudou na reutilização de código.

No capitulo 5 Concluímos que através da documentação formal que é a especificação do projeto, podemos expor de forma detalhada e conclusiva a ideia do software, contendo vários fatores como, funcionalidades, restrições gerais, escopo do portal, que ajudarão ao leitor entender o que se deseja ser desenvolvido.

No capitulo 6 Mostramos os modelos de analise de software detalhando os fluxos funcionais e as restrições, o que foi fundamental para a validação dos requisitos do portal, exibindo através de descrições ilustrativas de caso de uso em alto nível para o leitor entender de forma clara o que se planeja desenvolver.

## Etapa futura

A próxima etapa, planejamos publicar o portal em um host na internet que manteremos por um tempo determinado, assim poderemos divulgar nossa solução em redes de amigos e network. Conforme o retorno que o portal terá de popularidade, acessos e membros manteremos ele por mais tempo fazendo atualizações necessárias do software.

# 

# REFERÊNCIAS BIBLIOGRÁFICAS

AÉCE, I. Analisando o Microsoft PetShop 3.0. São Paulo, 2005. Disponível em: <http://www.projetando.net/Sections/ViewArticle.aspx?ArticleID=14>. Acesso em: 07 abr. 2014.

Alexander, Christopher. A Pattern Language. New York: Oxford University Press, 1977.

Arun, Gupta. Java EE 7 Essentials. O'Reilly, 2013.

BORBA, Felipe Moraes. (2005), *Razões Para a Escolha Eleitoral*: Influência da Campanha Política na Decisão do Voto em Lula

CAELUM.Apostila(2005). JAVA E ORIENTAÇÃO A OBJETOS. Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/>. Acesso em: Novembro de 2013

Cristiano, Matheus. SQL Injection: o que é, por que funciona e como prevenir. fonte. Online. Disponível em <http://www.devmedia.com.br/artigo-sql-m> Acesso em: 28 abril. 2014

Csillag, João Mario. Análise do Valor. Editora Atlas AS 4ª Edição São Paulo 1995.

Date, C.J.; Int. a Sistemas de Bancos de Dados, tradução da 4a.edição norte-americana, Editora Campus, 1991.

ELMASRI, R. & NAVATHE, S. B. Sistemas de Banco de Dados 4™ Edição, São Paulo: Addison Wesley, 2005.

Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra. Head First Design Patterns. O'Reilly Media, 2004.

FIGUEIREDO, Marcus Faria. (2007), “Intenção de Voto e Propaganda Política: Efeitos da Propaganda Eleitoral”,*Logos*, no. 27: 9-20.

FOWLER, Martin. Padrões de Arquitetura de Aplicações Corporativas / Martin Fowler; tradução Acauan Fernandes. – Porto Alegre : Bookman, 2006.

GAMMA, Erich, HELM, Richard; JOHNSON, Ralph, VLISSIDES, John. “Padrões de Projeto: soluções reutilizáveis de software orientado a objetos”. 1. ed. Porto Alegre: Bookman, 2000

Introdução à arquitetura e design de software: Uma visão sobre a plataforma Java, Paulo Silveira, Guilherme Silveira, Sérgio Lopes, Guilherme Moreira, Nico Steppat, Fabio Kung.

JOSHUA KERIEVSKY. Refatoração para padrões. Bookman, ISBN: 9788577802449, 2008.

KAUFELD, J. Access 95 para Windows para leigos: Um manual para novos usuários. LUDEMIR, J. São Paulo: Berkeley Brasil, 1996. 352 p.

Korth, H.F. e Silberschatz, A.; Sistemas de Bancos de Dados, Makron Books, 2a. edição revisada, 1994.

Macoratti, José Carlos. O modelo MVC - Model View Controller. fonte. Online. Disponível em <http://www.macoratti.net/vbn\_mvc.htm>. Acesso em: 25 Março.2014

MAGELA, R. Engenharia de software aplicada: fundamentos. Rio de Janeiro: Alta Books, 2006.

MYSQLBRASIL. Homepage Online. Disponível em: < http://www.mysqlbrasil.com.br >

Acesso em 20 fev. 2014

SUN. Core J2EE patterns: data access object. Santa Clara, [2007]. Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>. Acesso em: 15 abr. 2014

THAYER R. H. e DORFMAN M. Tutorial: Software Requirements Engineering. IEEE Computer Society Press, Los Alamitos, Calif., 1997

Test-Driven Develpment + Design Patterns, Mauricio Aniche e Eduardo Guerra, Casa do código

Vire o jogo com Spring Framework Autor, Henrique Lobo Weissmann, Casa do código

<http://www.caelum.com.br>

<http://www.eclipse.org>

<http://www.spring.io> <http://noticias.terra.com.br/brasil/politica/eleicoes/pesquisa-horario-politico-nao-influencia-60-dos-estudantes,0baedf0a2566b310VgnCLD200000bbcceb0aRCRD.html>

<http://www.jornaljovem.com.br/edicao4/tema14.php>.

<http://www1.folha.uol.com.br/poder/1159037-horario-eleitoral-nao-influencia-voto-dos-jovens-aponta-enquete.shtml>

<http://www.academia.edu/392273/Cientistas_Politicos_Comunicologos_e_o_Papel_da_Midia_nas_Teorias_da_Decisao_do_Voto>

<http://www.macoratti.net/vbn_mvc.htm>

<http://acens.com.br/blog/linguagens-do-mercado-julho-de-2013/>

<http://www.slideshare.net/fernandoabcampos/estudo-de-mapeamento-objetorelacional-com-framework-hibernate>

www.scrum.org

<http://www.caelum.com.br/apostila-java-web/>

<http://www.springframework.org>