

SISTEMA DE ENSINO PRESENCIAL CONECTADO

Tecnologia em Análise e Desenvolvimento de Sistemas

ALEXANDRE BRAGA

ARLEI FABIANO HAEFLIGER

DOUGLAS DE PAULA GONÇALVES

LUIS CARLOS DA SILVA

NEYMAR DANIEL CORRÊA

ROBERTO INACIO MOLON

**PRODUÇÃO TEXTUAL EM GRUPO**

Gestão do Processo de Desenvolvimento I

Caxias do Sul

2016

ALEXANDRE BRAGA

ARLEI FABIANO HAEFLIGER

DOUGLAS DE PAULA GONÇALVES

LUIS CARLOS DA SILVA

NEYMAR DANIEL CORRÊA

ROBERTO INACIO MOLON

**PRODUÇÃO TEXTUAL EM GRUPO**

Gestão do Processo de Desenvolvimento I

Trabalho apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimeno de Sistema da Universidade Norte do Paraná - UNOPAR, para as disciplinas de:

Engenharia e Projeto de Software

Projeto Orientado a Objetos

Programação para Web II

Orientadores:

Prof. Iolanda C. S. Catarino e Marco Ikuro Hisatomi

Disciplina: Engenharia e Projeto de Software

Prof. Paulo Henrique Terra

Disciplina: Projeto Orientado a Objetos

Prof. Anderson E. Macedo Gonçalves

Disciplina: Programação WEB II

Caxias do Sul

2016

SUMÁRIO

[2. OBJETIVO. 6](#_Toc451042219)

[3. DESENVOLVIMENTO. 7](#_Toc451042220)

[3.1. Engenharia e Projeto de Software. 8](#_Toc451042221)

[3.2. Projeto de Arquitetura: 8](#_Toc451042222)

[3.3. Arquitetura de Sistemas Distribuídos: 9](#_Toc451042223)

[3.3.1. Projeto em 3 Camadas: 10](#_Toc451042224)

[3.3.2. Camada de Apresentação: 12](#_Toc451042225)

[3.3.3. Camada de Negócio: 13](#_Toc451042226)

[3.3.4. Camada de Persistência: 13](#_Toc451042227)

[3.4. Arquitetura de Aplicação: 14](#_Toc451042228)

[3.4.1. Modelos de Ciclo da Vida: 14](#_Toc451042229)

[3.4.2. Cascata: 14](#_Toc451042230)

[3.4.3. Prototipação: 14](#_Toc451042231)

[3.4.4. Espiral: 15](#_Toc451042232)

[3.4.5. Codifica-remenda: 16](#_Toc451042233)

[3.4.6. Processos Ágeis: 16](#_Toc451042234)

[3.4.7 Estreme Programming: 16](#_Toc451042235)

[3.4.8 Processo Unificado Rational: 17](#_Toc451042236)

[3.5. Gerenciamento de Configurações: 18](#_Toc451042237)

[3.5.1. Baselines: 18](#_Toc451042238)

[3.5.1.1. Build: 19](#_Toc451042239)

[3.5.1.2. Release: 19](#_Toc451042240)

[3.5.1.3. Versão: 19](#_Toc451042241)

[3.5.2. Versionamento: 19](#_Toc451042242)

[3.6.1. EAP – Estrutura Analítica do Projeto: 20](#_Toc451042243)

[3.6.2. Cronograma das atividades para o desenvolvimento do projeto: 21](#_Toc451042244)

[3.6.3. Relação dos envolvidos, papeis dentro do projeto: 22](#_Toc451042245)

[3.7.1. Programação para Web II: 23](#_Toc451042246)

[3.7.2. Projeto orientado a objeto: 28](#_Toc451042247)

[3.7.3. Diagrama UML: 29](#_Toc451042248)

[3.7.3.1. Diagrama de Classe: 29](#_Toc451042249)

[3.7.3.2. Diagrama de componentes: 30](#_Toc451042250)

[3.7.3.3. Diagrama de caso de uso: 31](#_Toc451042251)

[3.7.3.4. Visão lógica e física do sistema: 32](#_Toc451042252)

[4. CONCLUSÃO 33](#_Toc451042253)

[5. REFERÊNCIA 34](#_Toc451042254)

**1. INTRODUÇÃO.**

Com este trabalho podemos ter uma noção maior dos tipos de modelos de ciclo de vida, percebendo todo o desenvolvimento e preparação, juntamente com o nível de dificuldade e atenção que cada modelo merece, não esquecendo da agilidade que o mercado de hoje exige, podemos conhecer também um pouco dos chamados processos ágeis e a junção de todas os conceitos e metodologias para um resultado esperado dentro do processo unificado rational (RUP).

A gestão de configuração de software é um tema abrangente e especializado, dentro das melhores práticas da Engenharia de Software que é possível o desenvolvimento consolidado dos conhecimentos para desenvolver o software, também podemos preparar os desenvolvedores para encarar os novos desafios nos negócios, e impor a competência destes profissionais.

Teremos o desenvolvimento de um programa para fazer o controle de inventário, um sistema desenvolvido em PHP, com consulta ao Bando de Dados MySQL com todas as Opções de CRUD.

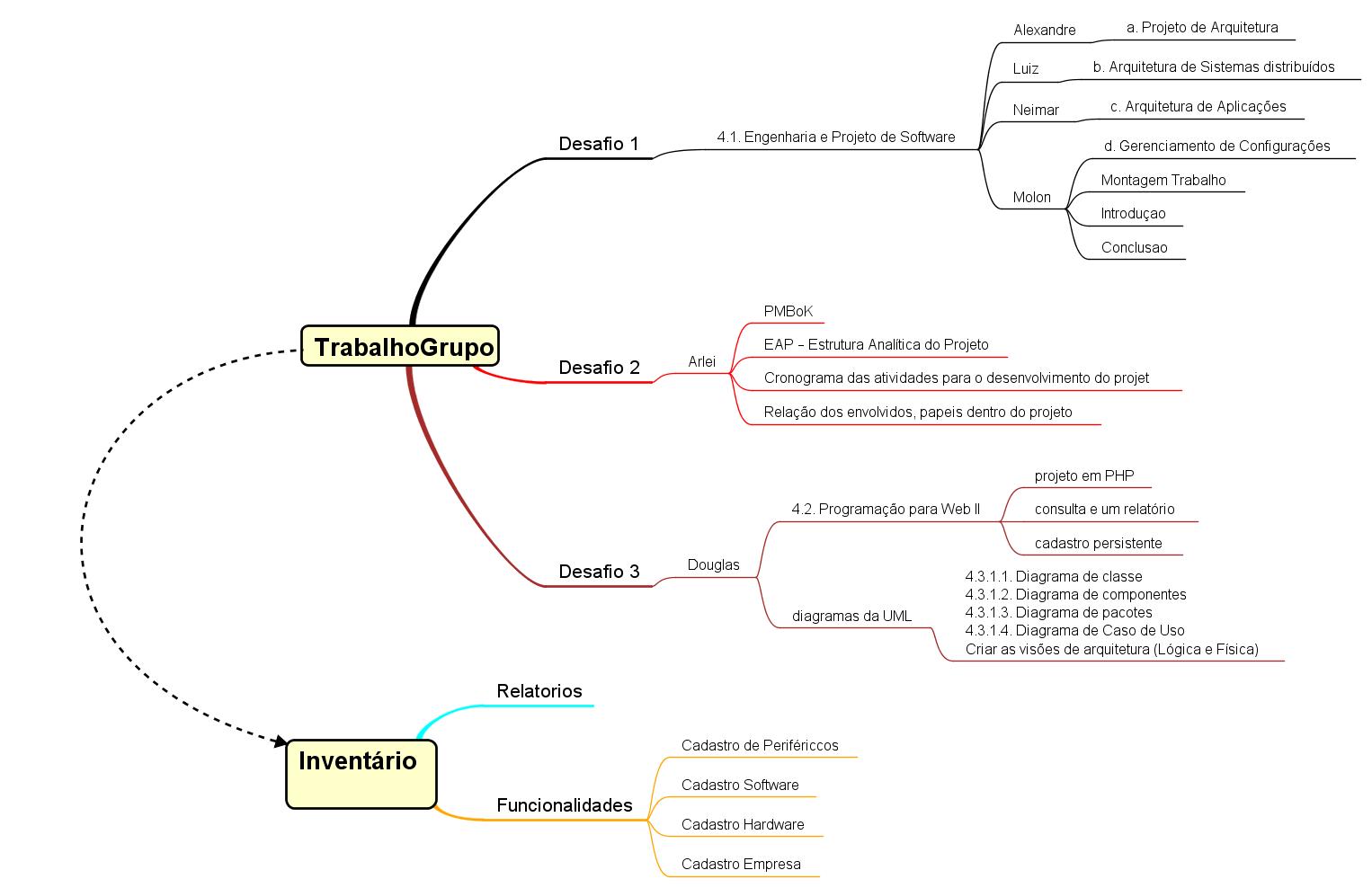
#### 2. OBJETIVO.

Consideramos que o objetivo Engenharia de Software é o aprimoramento da qualidade dos produtos de software e o aumento da produtividade dos engenheiros de software, além do atendimento aos requisitos de eficácia e eficiência.

Assim podemos desenvolver um software minimizando os futuros problemas no desenvolvimento, podemos mostrar com este trabalho a possibilidade e a versatilidade que temos para desenvolver sistemas.

#### 3. DESENVOLVIMENTO.

Nada melhor que começar este trabalho apresentando um Mapa mental do nosso projeto:



Com a Analise e Desenvolvimento de Sistemas, que estamos cursando, podemos entender e identificar problemas simples no desenvolvimento de sistema, o primeiro passo é conhecer bem a empresa que vamos atender e fazer o levantamento de todos os requisitos.

A empresa visitada foi a QualityInfo Soluções em Informática, onde a deficiência da empresa era ter um levantamento de Patrimônio, para ser mais especifico um inventário de Equipamentos especificando hardware e software por Empresa e por Usuário.

Vamos utilizar as ferramentas disponíveis para elaborar um sistema onde esta documentação hoje em Papel, fique armazenada e quando necessário será Gerado Relatórios.

#### 3.1. Engenharia e Projeto de Software.

Em 1987, o Project Management Institute - (PMI) elaborou um conjunto de padrões em Gerenciamento de Projetos, chamado The Project Management Body of Knowledge (PMBOK Guide). Tendo atualizações em 1996, 2000, 2004, 2008 e o mais atual é o PMBOK Guide Quinta Edição foi lançado em 2013, também existe em 10 idiomas entre eles o Português Brasileiro.

O Guia PMBOK é reconhecido como um padrão que contém uma linguagem comum no que diz respeito a Gerenciamento de Projetos.

O Guia PMBOK identifica esse subconjunto do conhecimento em gerenciamento de projetos que é amplamente reconhecido como boas práticas aplicáveis à maioria dos projetos na maior parte das vezes, existindo um consenso geral de que sua aplicação pode aumentar as chances de sucesso de muitos projetos.

A gerência de projetos de software é constituída por um conjunto de atividades, que devem ser administradas de acordo com parâmetros de custo, tempo e qualidade, com a utilização de métricas quantitativas e qualitativas, ao longo do processo de desenvolvimento, para garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado.

#### 3.2. Projeto de Arquitetura:

A Arquitetura de Sistemas de Informação foca na análise das necessidades dos usuários no em um sistema a ser desenvolvido onde não e aprofundado nos detalhes tecnológicos, mas sim se concentrar em que o cliente realmente precisa no que se refere as características do negócio, ou seja os analistas deste processo precisão compreender o funcionamento e os objetivos da empresa.

Arquitetura de Sistemas tem por objetivo facilitar a comunicação dos envolvidos no sistema antes da construção, portando temos uma visão genérica do sistema a ser criado onde criamos alternativas em um ponto no qual onde as mudanças são facilitadas não gerando impacto no projeto podendo ser tomada as decisões já que o mesmo estará precedendo a construção do sistema.

O estudo de cenários e a utilização casos de uso relativos ao projeto do cliente são de fundamental importância, onde os analistas possam realmente compreenderam o cliente tentou transmitir. Além de disporem de uma representação gráfica em que constam diferentes funcionalidades e agentes relacionados às mesmas, é comum que casos de uso venham acompanhados por uma descrição textual.

Requisitos como desempenho, segurança, tolerância a falhas e facilidade de manutenção deverão ser colocados em pauta. Artefatos como especificações suplementares podem ser de grande valia neste ponto, uma vez que contemplam estes aspectos que não estão diretamente relacionados a funcionalidades do sistema, mas que podem ainda influenciar de maneira determinante a implementação do projeto.

Se bem conduzida, este tipo de arquitetura pode levar o projeto a alcançar a satisfação do cliente. Assim sendo, um projeto bem definido conseguirá conquistar a confiança do cliente, ao mesmo tempo em que cumprirá os objetivos a que o mesmo se propôs.

A arquitetura é composta por alguns elementos, entenda-se classes, que em conjunto produzem o efeito desejado pela arquitetura como um produto final para o desenvolvimento. Para uma descrição detalhada de como cada classe trabalha, faz parte de toda a metodologia de desenvolvimento do

“Controle de Inventário” cada método e atributo existente é para que o aprendizado na utilização do sistema seja a menor possível.

#### 3.3. Arquitetura de Sistemas Distribuídos:

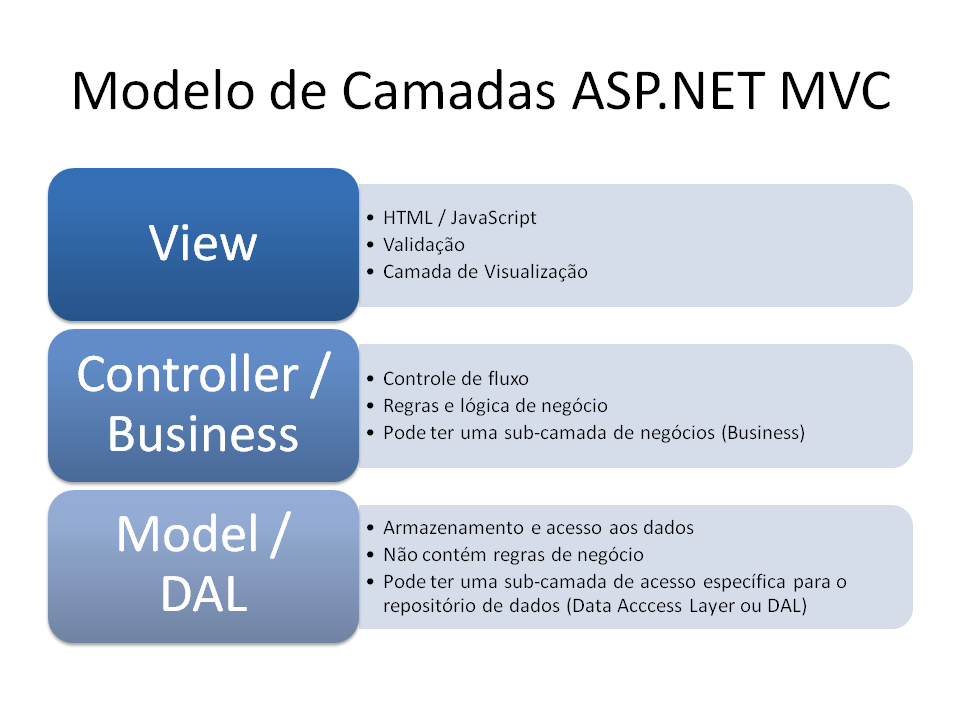
Esta parte do trabalho vamos falar sobre a estrutura de um sistema e sobre os componentes de software.

#### 3.3.1. Projeto em 3 Camadas:

As principais vantagens da arquitetura MVC em camadas tem o objetivo de permitir que aplicações sejam desenvolvidas de maneira produtiva e com facilidade de manutenção, ao meu ver, o desacoplamento entre as necessidades de apresentação e regras de negócio, a facilidade no uso de técnicas de TDD e a praticidade em se introduzir novas representações dos mesmos dados (os três conceitos, aliás, estão intimamente relacionados). Os objetivos principais são:

1. Modularidade - Dividir a aplicação em módulos tão independentes quanto possível.
2. Manutenibilidade - Reduzir o custo de manutenção da aplicação.
3. Extensibilidade - Permitir que novas funcionalidades sejam adicionadas sem grande impacto nas já existentes.
4. Reusabilidade - Permitir que classes e componentes sejam reusados em outros módulos da mesma aplicação ou em outras aplicações.

Entre outros benefícios, a divisão em camadas independentes permite a substituição da interface gráfica ou do meio de armazenamento dos dados (trocar arquivos por um SGBD, por exemplo) sem afetar as regras de negócio da aplicação. Isso facilita a reusabilidade das classes do negócio em outras aplicações e permite maior flexibilidade na escolha de tecnologias para implementar a aplicação.



As três camadas da arquitetura podem ser vistas na Figura 1, e têm os seguintes papéis:

1. Camada de Apresentação ou Interface do usuário (UI) – Esta camada tem a função de implementar uma interface de entrada e saída para a interação da aplicação com usuário. Seu papel é de validar as informações fornecidas pelo usuário e de conectá-lo aos serviços oferecidos pela camada de Negócio.
2. Camada de Negócio Camada de negócios - BLL– Esta camada representa o núcleo da aplicação e é responsável por implementar a lógica de negócio da aplicação. Nela estão todas as classes inerentes ao domínio da aplicação.
3. Camada de Persistência ou acesso a dados - DAL – Esta camada é responsável pela persistência e acesso aos dados da aplicação. Ela isola o resto da aplicação do meio de armazenamento usado (memória, arquivos, SGBD, aplicações legadas, etc.) de maneira que, se o meio de armazenamento for trocado, apenas as classes desta camada precisarão ser modificadas ou substituídas.



Visão geral da arquitetura em camadas

#### 3.3.2. Camada de Apresentação:

Esta camada é geralmente chamada de GUI (Graphical User Interface) e, no caso de aplicações .NET, oferece conteúdo estático e conteúdo dinâmico personalizado, que pode ser apresentado nos mais variados formatos disponíveis, como HTML, Windows Forms ou XML, para atender aos diferentes tipos de dispositivos cliente, como Desktop PC, celulares e PDAs.

A camada de apresentação é implementada com uso dos componentes visuais da plataforma .NET, tanto para ambientes Web quanto para ambiente desktop. O objetivo é permitir ao Desenvolvedor obter produtividade através da facilidade do desenvolvimento da interface, usando tecnologias como a CodeBehind, que CodeBehind permite separar em arquivos diferentes o código HTML do código de uma linguagem da plataforma .NET, como C# e VB.NET.

As classes dessas camadas utilizam os serviços oferecidos pela camada de negócios.

#### 3.3.3. Camada de Negócio:

Esta camada é responsável por implementar a lógica de negócio da aplicação. Nela estão todas as classes inerentes ao domínio da aplicação, como as classes de entidade e UI.

São as regras do negócio, as quais determinam de que maneira os dados serão utilizados. Esta camada foi deslocada para o Servidor de aplicações. Desta maneira, quando uma regra do negócio for alterada, basta atualizá-la no Servidor de aplicações. Após a atualização, todos os usuários passarão a ter acesso a nova versão, sem que seja necessário reinstalar o programa em cada um dos computadores da rede. Vejam que ao centralizar as regras do negócio em um Servidor de aplicações, estamos facilitando a tarefa de manter a aplicação atualizada.

#### 3.3.4. Camada de Persistência:

Nesta camada temos o servidor de Banco de dados, no qual reside toda a informação necessária para o funcionamento da aplicação. Cabe ressaltar, novamente, que os dados somente são acessados através do Servidor de aplicação, e não diretamente pela aplicação Cliente.

A camada de persistência é implementada através do mapeamento entre TypedDataSets e as tabelas da base de dados. Também são utilizados DataSets comuns para consultas que retornem mais de um registro de uma ou mais tabelas e que não possuem uma entidade de negócio associada, como é o caso de retorno de métodos de relatórios.

Banco de dados orientados a objetos não causa nenhum impacto na coleção de negócio, pois a mesma referência uma interface negócio-dados imutável. Deverá existir um conjunto distinto de classes para tratar cada tipo de meio de armazenamento possível da camada de persistência. Deve-se utilizar o padrão de projeto Factory (ver seção 8.3), para que sejam criadas as classes do meio de armazenamento desejado.

#### 3.4. Arquitetura de Aplicação:

Neste passo da Analise esta definido as decisões de projeto de arquitetura, onde é mostrado a organização do sistema e os modelos de controle.

#### 3.4.1. Modelos de Ciclo da Vida:

Os modelos de ciclo de vida são divididos em quatro tipos, são eles: cascata, prototipação, codifica-remenda e espiral.

#### 3.4.2. Cascata:

Este é o modelo considerado mais antigo, criado em aproximadamente 1970, tem como marco as fases sequenciais e sistemáticas, ou seja, como um vídeo game, só pode-se passar para uma fase posterior tendo completado a fase anterior, caso contrário, “*game over”*, deve-se começar todas as fases novamente.

#### 3.4.3. Prototipação:

É um processo que visa entender os objetivos e necessidades do usuário antes de desenvolver um projeto, sendo assim a prototipação divide-se em algumas partes:

1. Obtenção de requisitos:

Esta etapa visa conseguir o maior número de informações possíveis sobre o que o cliente deseja e a sua necessidade, analisando também algumas possíveis definições adicionais que podem melhorar o projeto.

1. Projeto Rápido:

É uma espécie de projeto onde somente é feito as entradas e saídas do programa, as quais são mais visíveis ao usuário.

1. Construção do protótipo:

Nesta fase a realização do projeto é concretizada.

1. Avaliação do protótipo:

Aqui o cliente juntamente com o desenvolvedor avalia o protótipo.

1. Refinamento do Protótipo:

Esta fase funciona como uma peneira onde o cliente e o desenvolvedor escolhem o que continuará ou não no software.

1. Construção do produto:

Nesta fase é a essencial, onde o protótipo (fase de testes) termina e o projeto é construído baseando-se apenas no que o desenvolvedor e o cliente acharam conveniente.

#### 3.4.4. Espiral:

Espiral é um tipo de ciclo de vida onde usa-se pequenos ciclos, de uma certa maneira a forma sequencial é usada, mas ao mesmo tempo, para cada etapa existe um teste em que a fase é analisada, caso não haja sucesso no protótipo dessa fase pode-se deixá-la inacabada e seguir o projeto adiante e mais tarde voltar a fase inacabada, pulando assim de um ciclo para outro sem alterar o produto do projeto.

#### 3.4.5. Codifica-remenda:

É quando o desenvolvedor cria seu protótipo, faz seus testes, cria seu projeto e apresenta ao cliente, se o mesmo analisar sugerir ou exigir alguma coisa o desenvolvedor somente acrescenta as exigências no programa e assim vai fazendo até seu cliente mostrar-se satisfeito.

#### 3.4.6. Processos Ágeis:

Cada vez mais o mercado procura o melhor trabalho desenvolvido no menor espaço de tempo e com recursos limitados. Assim surgiram os processos ágeis que são um conjunto de metodologias que geram uma base de conceitos para a criação de um projeto de software. Exemplo de processo ágil é Extreme programming (XP).

#### 3.4.7 Estreme Programming:

Uma das metodologias ágeis usadas por pequenas e medias empresas, onde existem constantemente mudanças, alterações e ajustes no seu desenvolvimento.

Possui 04 valores fundamentais: comunicação, simplicidade, feedback e coragem.

Por meio destes quatro valores é possível realizar constantemente vários feedbacks, assim analisando e gerando resultados simples e implantando mudanças sempre a procura do melhor e mais satisfatório resultado.

#### 3.4.8 Processo Unificado Rational:

RUP - O processo unificado visa distribuir em etapas e auxiliar no desenvolvimento de sistemas orientados a objetos.

O RUP acaba se tornando semelhante, contudo para cada etapa tem que se alcançar um resultado. Podemos chamar de uma “prova real”, como se fosse em uma conta de matemática, sempre achamos um resultado e para sabermos se está certo, fazemos então a prova real, gerando assim um projeto (“uma conta”) sem falhas, com o resultado esperado. Para se exata o RUP divide-se em 4 fases e 9 disciplinas. São elas:

* - Concepção;
* - Elaboração;
* - Construção;
* - Transição;
* - Modelagem de negócios;
* - Requisitos;
* - Análise;
* - Implementação;
* - Teste;
* - Implantação;
* - Gerência de configuração e mudança;
* - Gerência de projetos;
* - Ambiente.

#### 3.5. Gerenciamento de Configurações:

Os programas de computadores ou sistemas de software estão em sempre evoluindo. Portando podemos definir que o desenvolvimento de um produto consome 75% do custo total do seu ciclo de vida e 20% em manutenção de erros de implementação na adaptação do software devido a modificações, regras de negócios entre outros.

A Gerência de Configuração de Software existe para controlar estas modificações onde usamos métodos e ferramentas visando aumentar a produtividade e diminuir erros que ocorrem durante o desenvolvimento e implantação do Software.

No gerenciamento de configurações está vinculada as principais regras de controle do software como as correções, extensões e adaptações que deveram ser aplicadas durante o ciclo de vida de um programa e também temos a evolução sistemático e a rastreabilidade indispensável quando o programa e manipulado por muitas pessoas, ou seja, produzir artefatos e modificá-los por diferentes recursos desde o planejamento e levantamento de requisitos até a construção e entrega do produto.

Todas as alterações são registradas visando a garantia de recuperação dos dados alterados tanto pelos desenvolvedores ou por solicitação do cliente.

Portanto a Gestão de configuração de software é um conjunto de atividades de acompanhamento e controle que começa quando o projeto de engenharia de software tem início e só termina quando o software é retirado de operação, palavras de PRESSMAN, 2006, p. 599.

#### 3.5.1. Baselines:

Baselines que é a configuração do software em um ponto do tempo, serve como base para os passos posteriores do desenvolvimento; são marcos no processo de desenvolvimento e um artefato ou vários artefatos somente chega a uma configuração depois de um baseline definindo.

Os Baselines permitem a possibilidade de reproduzir uma versão anterior do sistema, estabelece uma ligação entre o predecessor e o sucessor dos artefatos do projeto e o controle de mudanças com referencial para comparações, negociações e discussões.

#### 3.5.1.1. Build:

Build: é uma versão do meu sistema antes de ser entregue, ou seja, ela é gerada passo a passo sendo definido como build, lembrando que nesta parte do desenvolvimento que ela ser perfeita. Portando dependendo do que foi definido no projeto a geração de Build deve automatizada e realizada com frequência.

#### 3.5.1.2. Release:

Release: é a depuração dos Build, nesta fase do projeto e feita a entregue do sistema e, portanto, tem que estar perfeita, aqui já estamos falando de um produto de software, sem erros entregue e implantado no cliente.

#### 3.5.1.3. Versão:

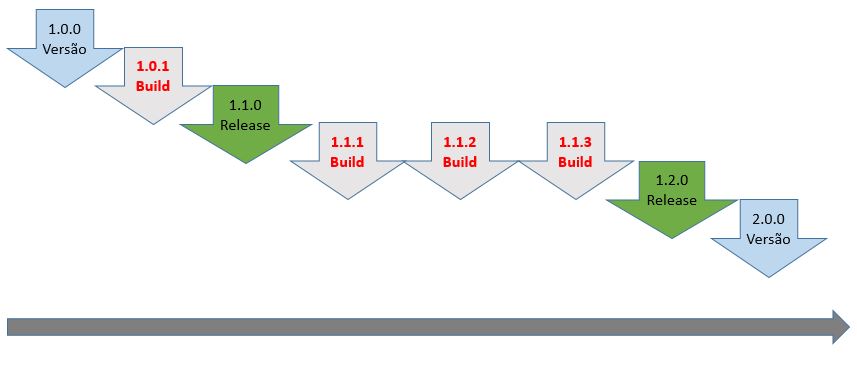
Versão: Como foi definido dentro de uma gerencia de configuração podemos definia as versões de forma que o desenvolvimento evolui muito até chegar a mudar o programa.

#### 3.5.2. Versionamento:

Quando definimos no plano de Gerenciamento de configurações existem regras especificas para seguir com o versionamento em todas as fases do programa, então ela pode ser:

1. Temporal: definidas de tempo em tempo a geração de um Build ou uma versão.
2. Arbitrária: são definidas as Build, release ou versão em qualquer momento.
3. Quantitativo: definida na política do desenvolvimento como exemplo após 3 build definir a release, ou a cada release uma versão.

Veja a Imagem abaixo.

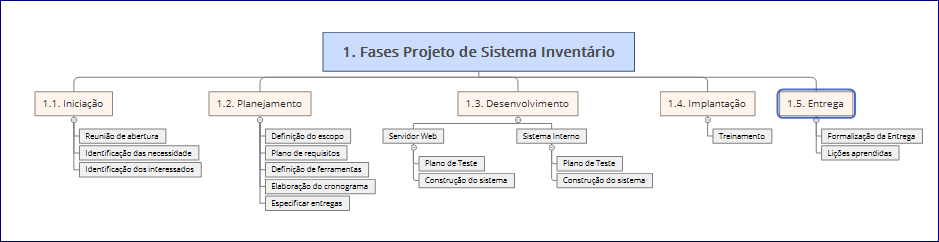


#### 3.6.1. EAP – Estrutura Analítica do Projeto:

A EAP é uma decomposição hierárquica orientada à entrega do trabalho a ser executado pela equipe para atingir os objetivos do projeto e criar as entregas requisitadas. Cada nível descendente da EAP representando uma definição gradualmente mais detalhada do trabalho do projeto. É finalizada pela determinação de contas de controle. Esses identificadores produzem uma estrutura para sumarização hierárquica de custos, cronograma e informações sobre recursos.

Para o projeto do sistema de inventário foram definidas várias etapas do projeto a serem realizadas, começando pela reunião de abertura, com identificação das necessidades e dos interessados.

Tendo definido o início do projeto, deu-se o planejamento do projeto, que permitiu dar início aos trabalhos de desenvolvimento do sistema. Com a conclusão do desenvolvimento foi feita a implantação do sistema propriamente dita, com treinamento aos usuários e após isso a entrega do resultado.

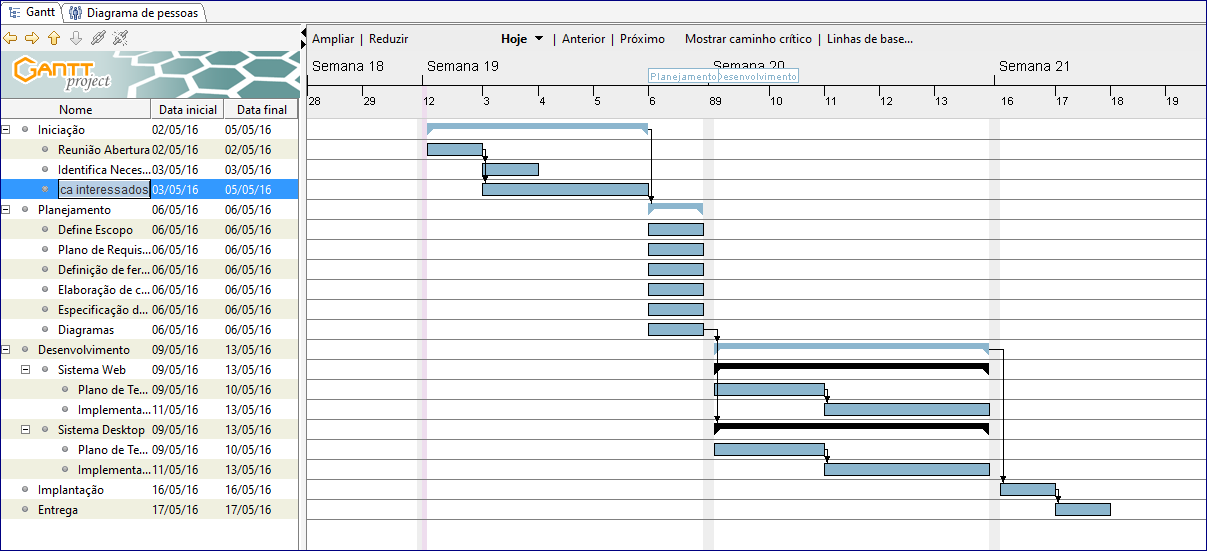


#### 3.6.2. Cronograma das atividades para o desenvolvimento do projeto:

Desenvolver o Cronograma é um processo iterativo que envolve analisar a sequência das atividades, sua duração, seus requerimentos de recursos e suas restrições para criar o cronograma do projeto e determinar as datas de início e término de cada atividade.

Uma boa prática para montar o cronograma é descrever as fases, as entregas, os pacotes de trabalho, as atividades e os marcos conforme tabela abaixo.

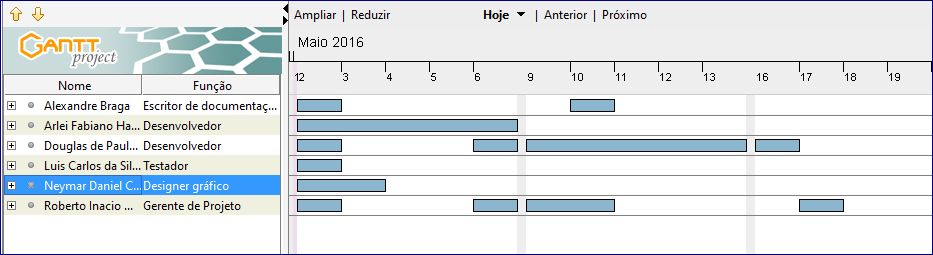
Durante o planejamento do projeto, foi desenvolvido o cronograma das atividades a serem realizadas durante o projeto. Para a construção desse documento foram analisadas as sequências das atividades, sua duração, seus requerimentos de recursos e suas restrições, determinando assim as datas de início e término de cada atividade.



#### 3.6.3. Relação dos envolvidos, papeis dentro do projeto:

Segundo o Guia PMBOK®, o pessoal do projeto estará pronto quando pessoas apropriadas tiverem sido designadas. A documentação dessas designações pode incluir um diretório da equipe do projeto, memorandos para membros da equipe, e inclusão de nomes em outras partes do plano de gerenciamento do projeto, como organogramas e cronogramas.

Uma vez definidas as atividades e necessidades do projeto, foi feita a definição das atividades e responsabilidades de cada integrante da equipe, conforme suas aptidões.



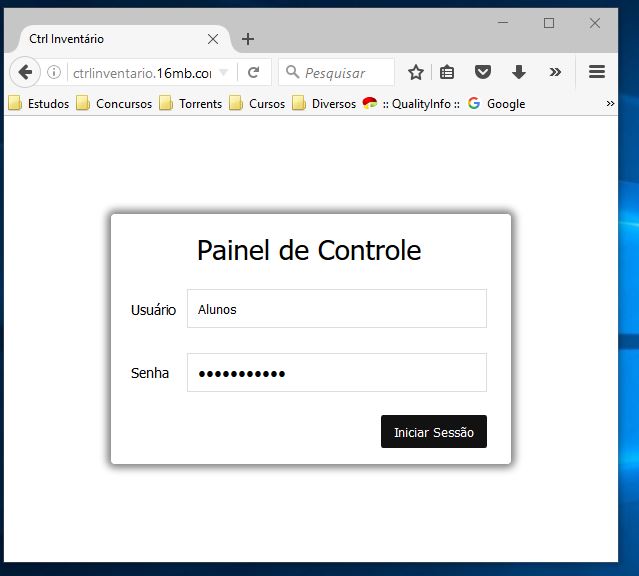
#### 3.7.1. Programação para Web II:

Agora apresentaremos um projeto em PHP de um sistema de controle de Inventário, com um cadastro persistente e estruturado.

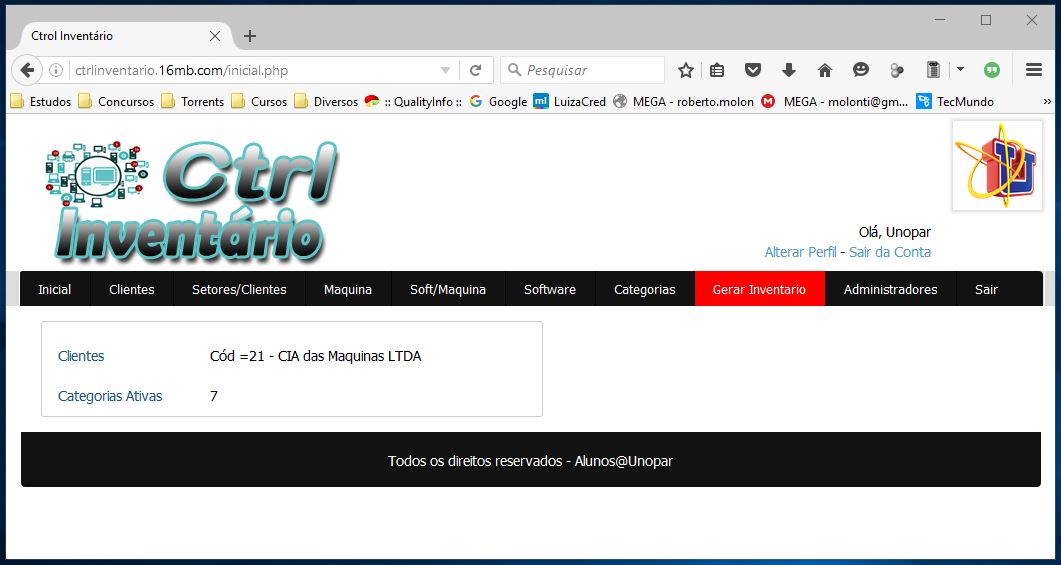
Será demostrado neste projeto, um cadastro de clientes e produtos, teremos as telas para fornecer consultas e cadastro do projeto e assim podemos demostrar os relatórios.

Segue também o link para acesso ao sistema.

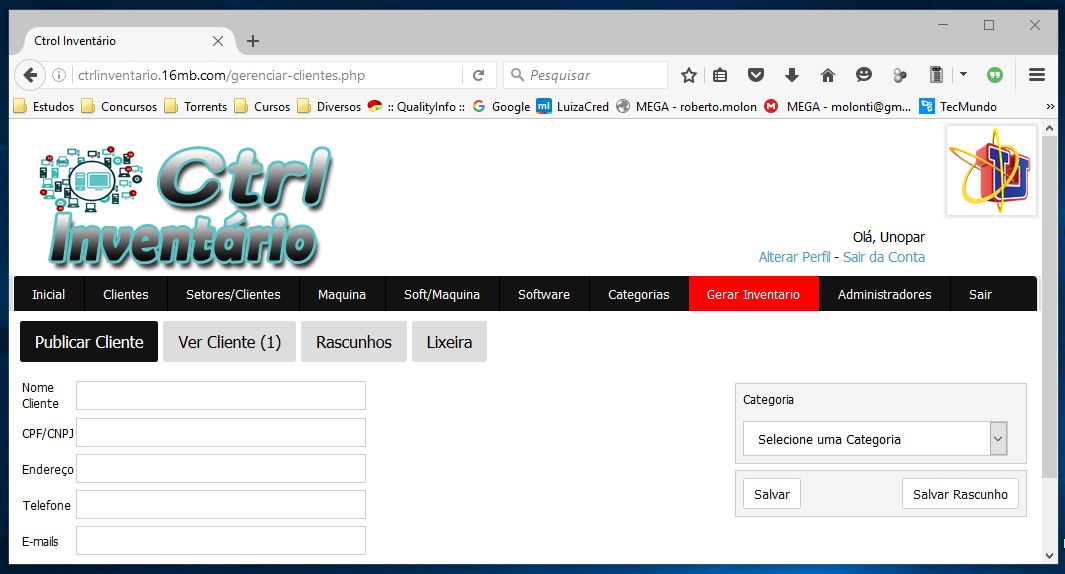
<http://douglasgoncalves.info/trabalhos/CtrolInventario/>

Usuário: unopar Senha: unopar

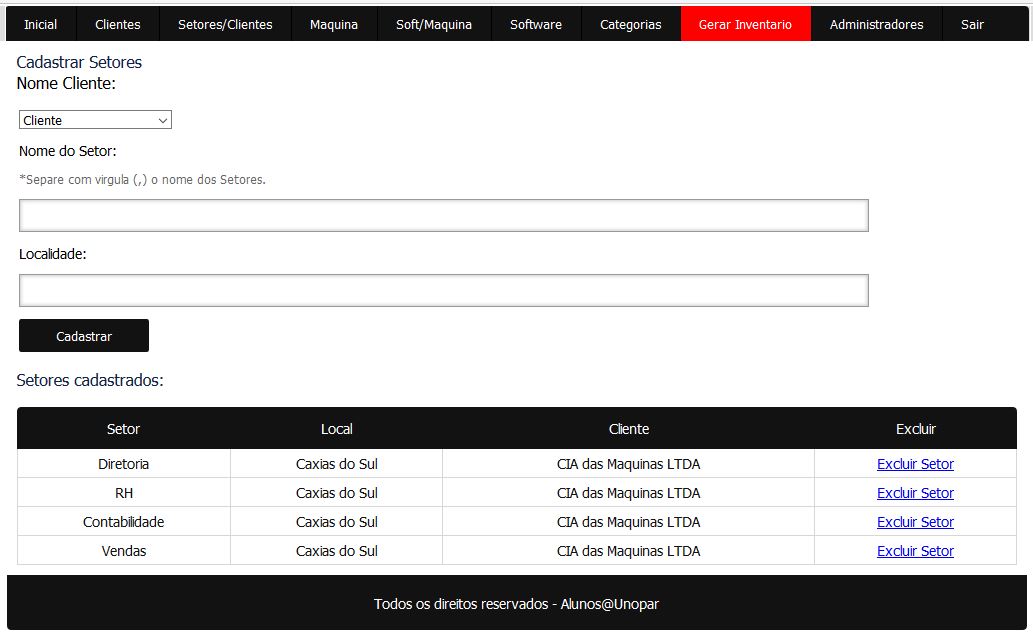
Tela de Login.



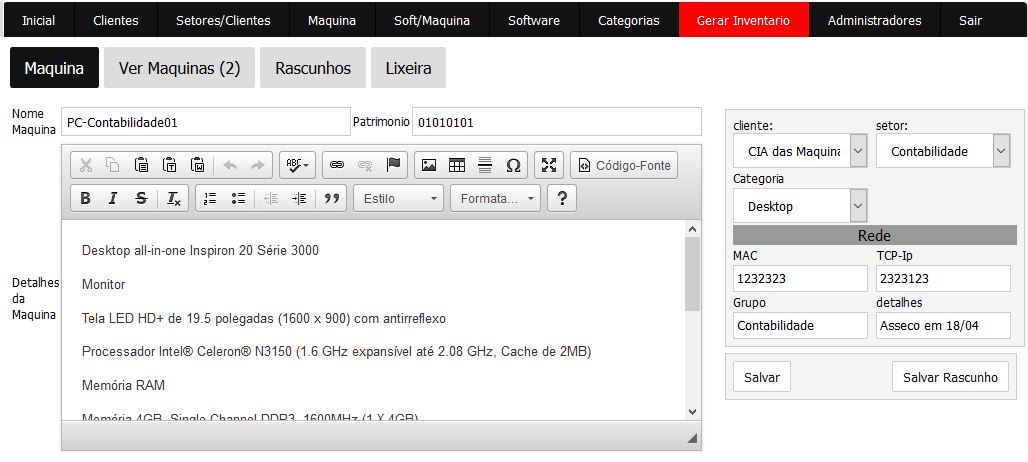
Tela dos Menus



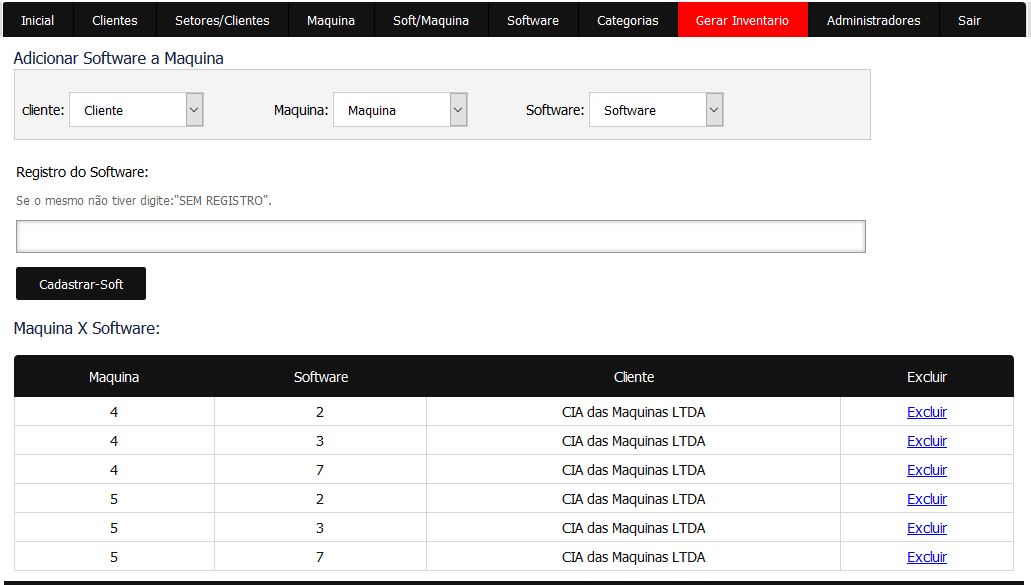
Tela de Cadastro do Cliente.



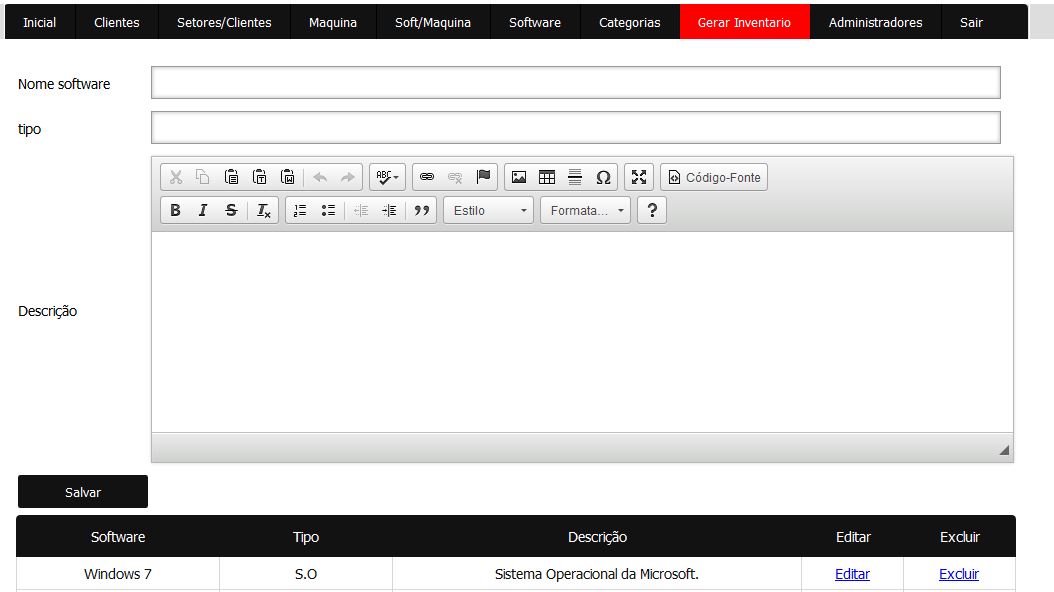
Cadastro dos Setores.



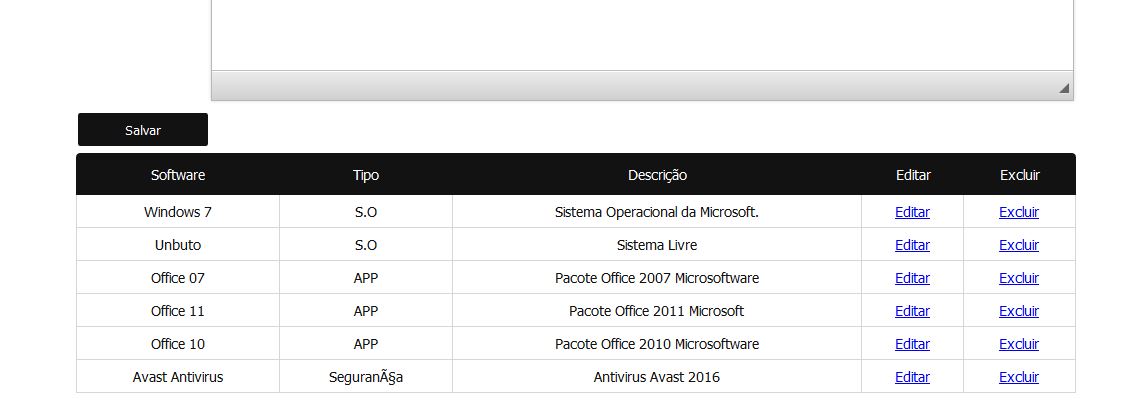
Tela de Cadastro do Equipamento.



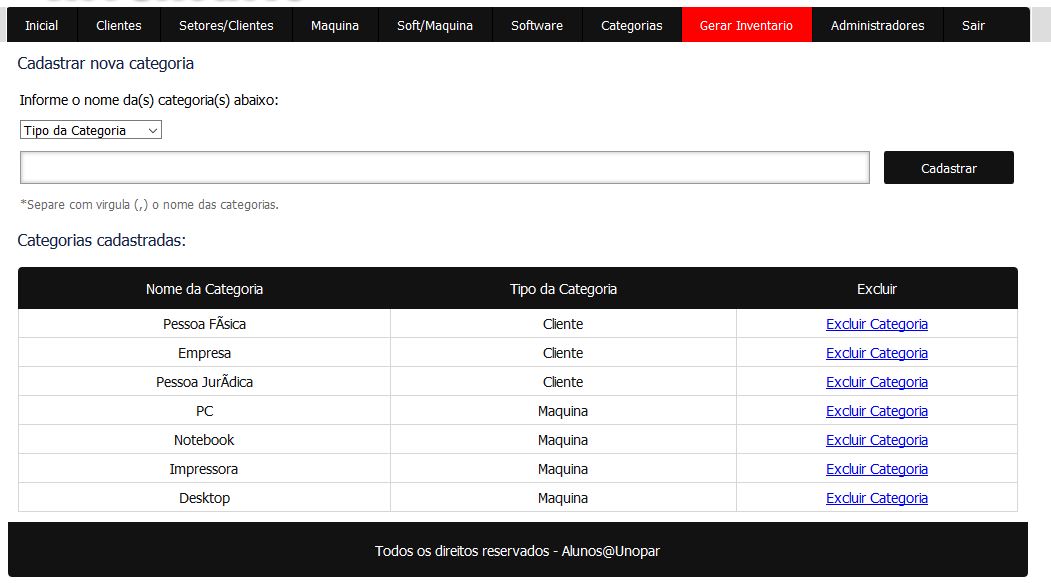
Tela de Cadastro do Software.



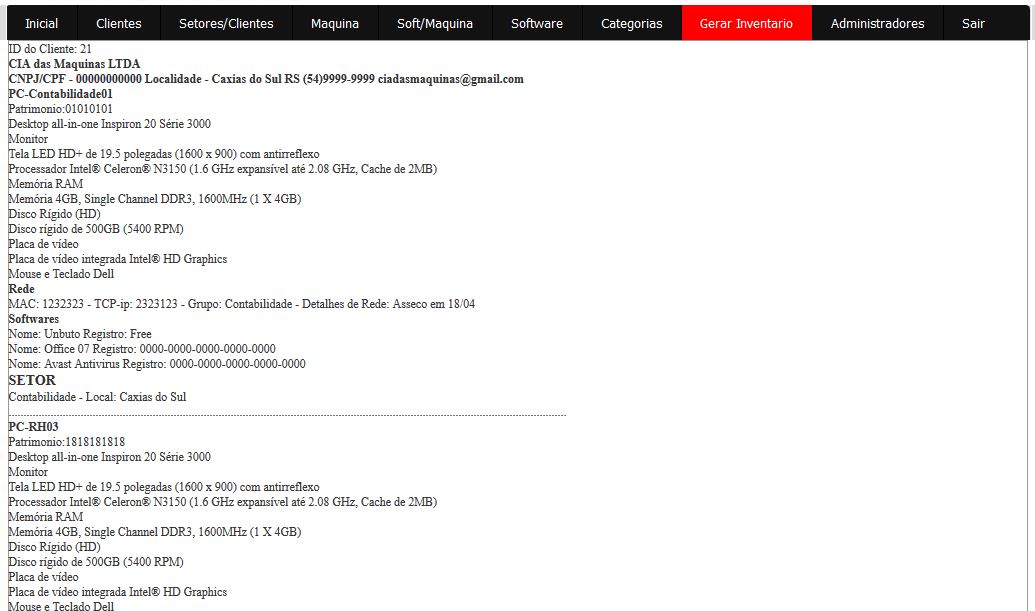
Tela de Cadastro de Software Detalhada.



Tela de Software.



Tela de Cadastro de Categorias.



Tela de Relatório.

#### 3.7.2. Projeto orientado a objeto:

O projeto orientado a objetos está baseado na ideia de definir o sistema usando objetos autocontido e classes de objetos. Projeto orientado a objetos é uma estratégia de projeto em que os projetistas pensam em termos de coisas, em vez de funções. A funcionalidade do sistema é expressa em termos de serviços oferecidos pelos objetos.

O Projeto Orientado a Objeto é a fase em que se procura organizar melhor os objetos, de formar que se consiga solucionar seus problemas, para que se possa fazer a interligações entre os objetos que estão dispostos ao longo do projeto.

Projeto orientado a objeto é parte do desenvolvimento orientado a objeto e é baseado na Análise Orientada a Objetos e cria a base para a programação do sistema Orientado a Objetos.

O projeto Orientado a Objetos traz inúmeras vantagens entre as quais: Facilidade de manutenção, pois objetos podem ser entendidos como entidades independentes e com isso facilita a manutenção no sistema, uma vez que alterações internas em um objeto não interferem necessariamente nos outros objetos. Os objetos são componentes potencialmente reutilizáveis, uma vez que são independentes e se definidos de forma genérica permitirão a instanciação em qualquer lugar que o necessite. Para vários sistemas, existe um nítido mapeamento entre as entidades do mundo real para objetos no sistema.

#### 3.7.3. Diagrama UML:

Na definição do Projeto Orientado a Objetos são desenvolvidos os diagramas, que definem o que compõem o sistema e como tudo isso irá funcionar. Na UML existem inúmeros diagramas. O diagrama de Casos de Uso, onde são apresentadas as principais funcionalidades do sistema do ponto de vista do usuário. O diagrama de Classes que é a representação das classes e suas relações, sendo uma estrutura abstrata dos dados que compõem o sistema. O Diagrama de Componentes que captura a estrutura física da implementação, sendo construído como parte da especificação da arquitetura. O diagrama de componentes tem como objetivo: Organizar o código fonte (ambiente de desenvolvimento), construir um release executável (ambiente de produção) e especificar uma base de dados física.

#### 3.7.3.1. Diagrama de Classe:

No levantamento de requisitos para o sistema de inventário, foram identificadas algumas classes que precisam fazer parte do sistema. Abaixo a definição já desenhada do diagrama de classes para o sistema.

A classe cliente que identifica o dono das máquinas propriamente dito. Cada cliente poderá ter diversos locais onde ficam as máquinas, sendo identificado como Setores. Nos setores estarão as máquinas que farão parte do inventário. Cada máquina poderá ter uma lista de softwares instalados, e poderá ter um ou mais acessos configurados, sendo que todas essas informações compõem o cadastro.

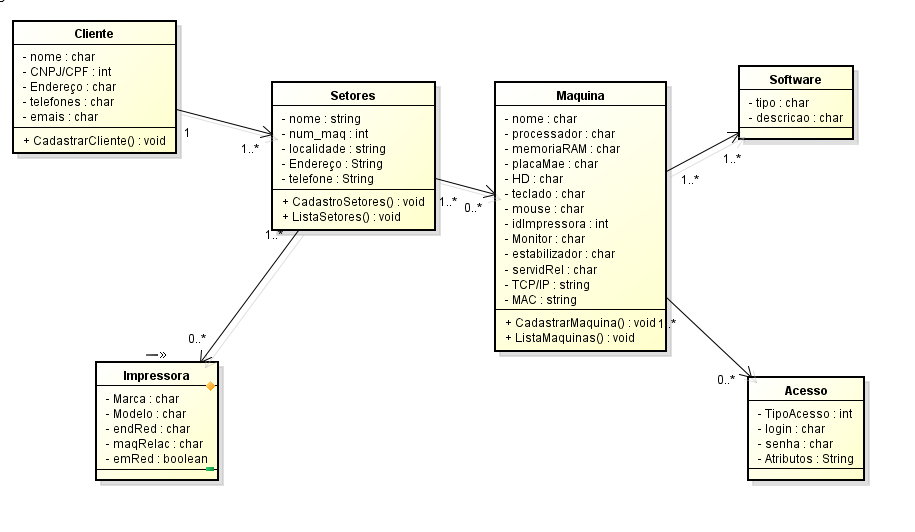
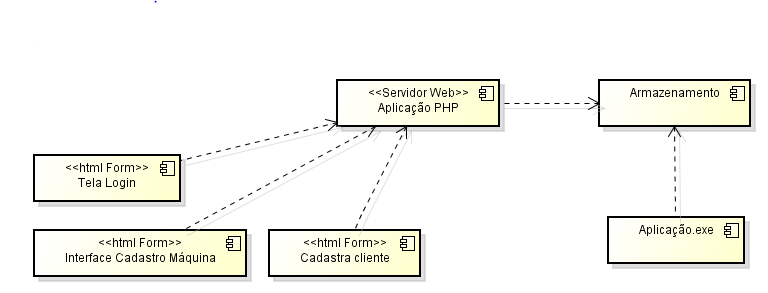


Figura: Diagrama de classes Fonte: Do autor.

#### 3.7.3.2. Diagrama de componentes:

O sistema terá por componente um servidor Web que hospedará as páginas que compõem o cadastro que ficará disponível de qualquer lugar para as pessoas autorizadas a realizarem o cadastro das informações. O armazenamento, que é a parte do banco de dados e uma aplicação executável, acessível somente ao administrador do sistema. No sistema terá as páginas Web, com formulários onde os cadastros serão acessados e feitos.



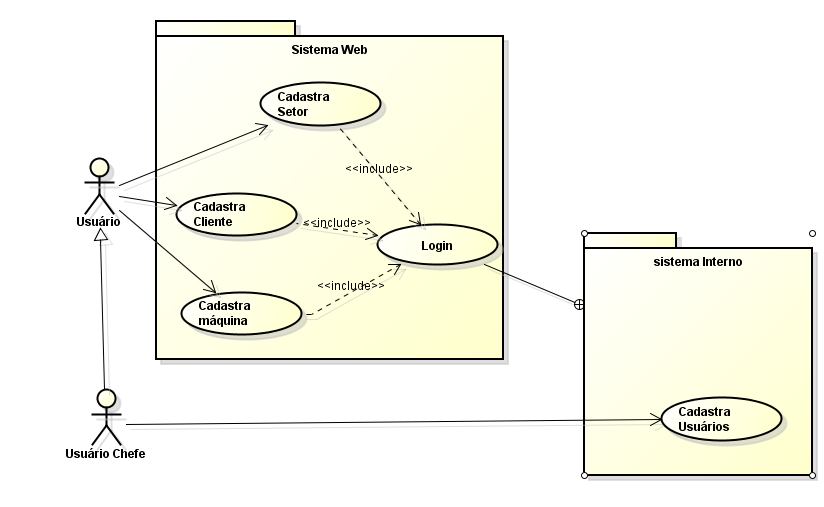
#### 3.7.3.3. Diagrama de caso de uso:

Casos de uso são narrativas em texto, descrevendo a unidade funcional, e são amplamente utilizados para descobrir e registrar requisitos funcionais nos sistemas. Os diagramas de Casos de Uso são representações dos Casos de Uso e podem ser representados por uma elipse contendo, internamente, o nome do caso de uso.

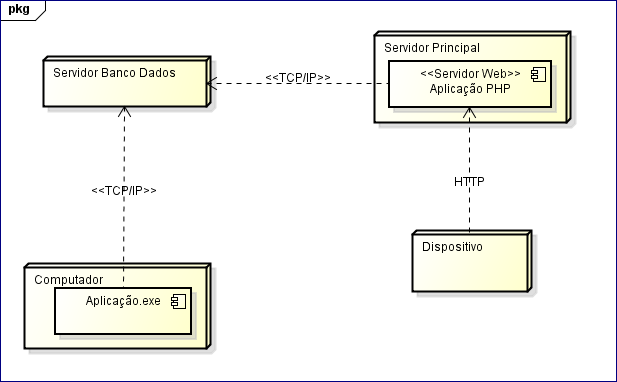
Um caso de uso é uma sequência de interações entre o ator (alguém ou algo que interage com o sistema) e o sistema, que acontece de forma atômica, na perspectiva do ator. Isso significa que quando criamos um caso de uso apenas nos preocupamos com “o que” o sistema deve fazer e não com “o como” deve fazer.

Essa documentação desenvolvida além de dar ideia do escopo do sistema, consegue descrever o que será desenvolvido em uma linguagem adequada também para comunicação com os stakeholders.

Para o sistema foram identificadas a necessidade de um sistema web, onde os cadastros seriam feitos através de páginas web em computadores ou dispositivos. Para o controle dos usuários verificou-se a necessidade de um cadastro de usuários, acessível somente a um usuário chefe.



#### 3.7.3.4. Visão lógica e física do sistema:



#### 4. CONCLUSÃO

Por meio deste trabalho podemos analisar e presenciar todas as etapas que a geração de um sistema exige, podemos vivenciar as etapas fazer teste e desenvolver um software para controlar o inventario de maquinas das empresas, visando a usabilidade e a utilidade que a empresa precisa e gerenciar através de captação de requisitos o cliente solicitou.

#### 5. REFERÊNCIA

**PERINI**, Luis Cláudio, **Hisatomi**, Marco Ikuro e **Berto**, Wagner Luiz, **Engenharia de Software.** São Paulo: Editora Pearson Education do Brasil, 2009.

**FREITAS**, Veronice, **Programação Web II.** São Paulo: Editora Pearson Education do Brasil, 2013.

**SILVA**, Flavio de Almeida e, **Desenvolvimento Orientado a Objetos II.** São Paulo: Editora Pearson Education do Brasil, 2013.

SOFTWARE, Ciclo de Vida. Disponível em: **<http://www.mandrado.com/gerenciamento-do-ciclo-de-vida-de-software/1-gerenciamento-de-configuracao-de-software>**. Acessado em mar/2015

TUTORIAIS, Gerencia de Configurações. Disponível em: **<http://www.pronus.eng.br/artigos\_tutoriais/gerencia\_configuracao/gerencia\_configuracao.php?pagNum=0>**. Acessado em mar/2015

DEVMEDIA, Software. Disponível em: **<http://www.devmedia.com.br/gerencia-de-configuracao-de-software/9145>.** Acessado em mar/2015

UOL, Mapa Mental. Disponível em: **<http://vestibular.uol.com.br/noticias/redacao/2013/06/03/mapa-mental-possibilita-memorizar-em-longo-prazo-conteudos-para-o-vestibular.htm>.** Acessado em mar/2015

Vídeos:

**< https://www.youtube.com/watch?v=m74Pmlo0ZMI >**

**< https://www.youtube.com/watch?v=KX2N5k-OcFs >**

**< https://www.youtube.com/watch?v=R3dHpV5Mjho>**

**<** **https://www.youtube.com/watch?v=-ZoMuf5Xt5Q>**