|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Autor** | **Descrição** | **Versão** |
| 30/08/2015 | Giuliano Costa | Criação do documento | 0.1 |

[DESENV] – Documento de Arquitetura de Software

**OBJETIVO**

O Documento de Arquitetura de Software provê uma visão geral da arquitetura através de diferentes tipos de visões para descrever os diferentes aspectos do sistema.

**Representação Arquitetural**

A Arquitetura do HMA (Hermes Management Assistant), será representada por 4 visões:

* Visão de Casos de Uso Significativos: Esta seção apresenta alguns casos de uso ou cenários (obtidos dos diagramas de caso de uso), arquiteturalmente mais significativos para o software final e que, portanto, são utilizados para validação da arquitetura proposta;
* Visão Lógica: Esta seção apresenta a visão lógica da arquitetura. Tem por objetivo dar uma visão estática e dinâmica do sistema. Para se conseguir isto esta visão será dividida em:
  + Elementos do Modelo Arquiteturalmente Significativos: Aqui conterão os diagramas que representam graficamente as camadas, subsistemas e pacotes do projeto.
  + Realização dos Casos de Uso: Neste pacote, serão armazenadas as informações das realizações de casos de uso relacionadas aos aspectos dinâmicos e classes específicas (visão estática) utilizados nos Casos de Uso.
* Visão de Componente: Esta seção descreve a estrutura geral do modelo de implementação, a divisão do software em camadas e subsistemas no modelo de implementação e todos os componentes significativos do ponto de vista da arquitetura.
* Visão de Implantação: Esta visão compreende o diagrama de implantação (visão topológica da aplicação) e os elementos que o compõem: processos, dispositivos, conexões, componentes de cada processo, configurações usadas para desenvolvimento, teste e produção.

**Observação**

Este documento pode ser alterado a qualquer momento, sendo necessário a inclusão no cabeçalho o motivo, data e o integrante responsável pela modificação.

**Objetivos e Restrições Arquiteturais**

A arquitetura proposta tem como objetivo disponibilizar um sistema acessível à todos os funcionários do estabelecimento, utilizando camadas separadas logicamente e aderentes à arquitetura especificada expondo funcionalidades de negócios e/ou aplicação.

* **Módulo Gestão**

Módulo responsável pela gestão do estabelecimento e do sistema: Agenda de eventos, Contas a Pagar e Receber, Produtos, Fornecedores, Funcionários, gestão de usuários, abertura e fechamento de caixa, abertura e fechamento de comandas;

* **Módulo Pedidos**

Módulo responsável por controlar os pedidos dos clientes; Emitindo, cancelando e fechando;

**Requisitos Não Funcionais**

Entende-se por requisitos não funcionais termos relacionados ao desempenho, usabilidade, confiabilidade e segurança da aplicação;

* **Portabilidade**
  + O sistema deverá ser hospedado em servidor Windows Server, preferencialmente versão 2012;
* **Implementação**
  + O sistema deverá ser desenvolvido utilizando .NET principalmente C# (Cê Sharp), por ser a linguagem em que a equipe possui maior controle;
* **Interoperabilidade**
  + O sistema deverá comunicar-se com Sql Server, pela facilidade de integração com a linguagem acima definida;
  + O sistema deverá ser multi-thread, visando a performance de requisições;
  + Toda requisição deverá ser assíncrona.
* **Éticos**
  + O sistema não deverá mostrar dados financeiros a usuários não autorizados (garçom, segurança, cozinha);

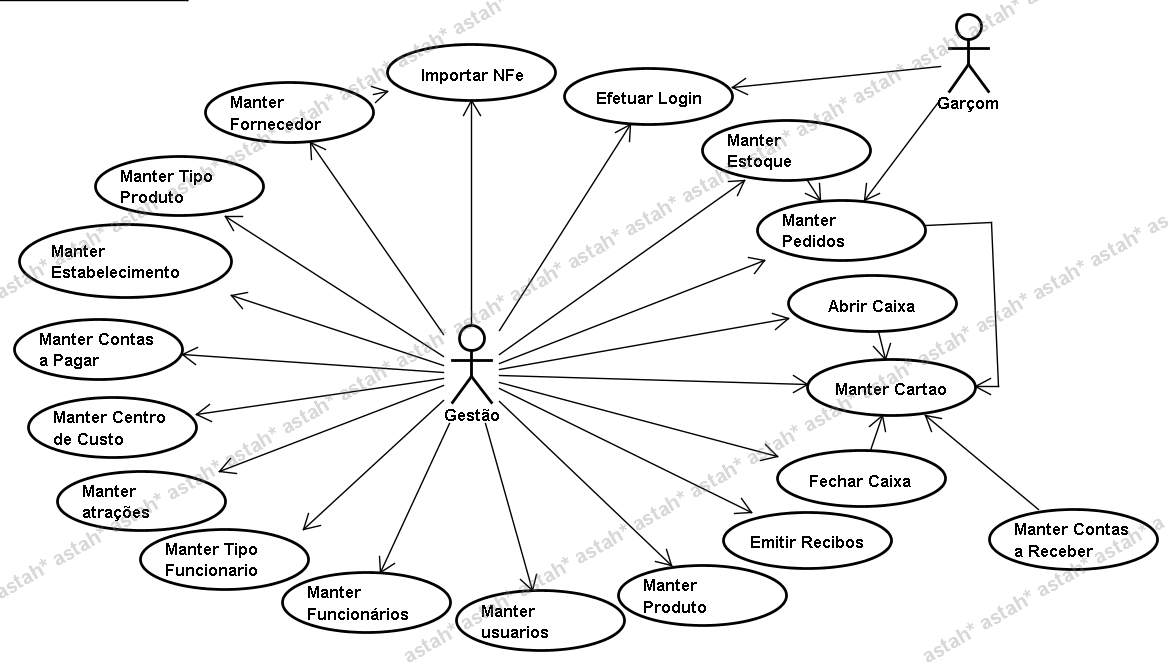
**Requisitos Funcionais**

Entende-se por requisitos funcionais funções que definem um software ou parte dele;

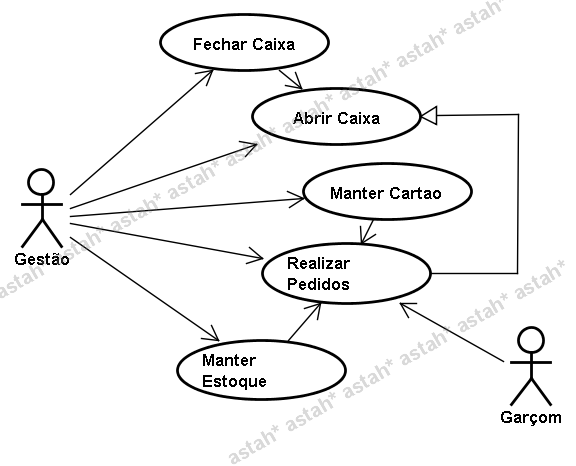
* O sistema deverá emitir relatórios e recibos em formato .PDF;
* O sistema deverá importar NFe (Nota Fiscal Eletrônica) e salvar no caminho estruturado a seguir: *../HermesManagementAssistant/XML/NomeFornecedor/DataNota.pdf*;
* Os recibos emitidos a funcionários deverão estar salvos no caminho estruturado a seguir: *../HermesManagementAssistant/Recibos/NomeFuncionario/DataPagamento.pdf*;
* O sistema deverá ser capaz de trabalhar com qualquer impressora não fiscal disponíveis atualmente no mercado (Somente impressoras com produção e suporte técnico atual);

As restrições arquiteturais ficam por conta do conhecimento da equipe sobre implementação e o prazo estipulado para o desenvolvimento. O sistema não efetuará backup automaticamente;

**Visão de Casos de Uso**



**Visão de Casos de Uso Significativos**



Os casos de uso significativos foram escolhidos utilizando por base os processos de maior impacto para o estabelecimento, não foi efetuado um levantamento de risco dos casos de uso pela não necessidade, visto que a implementação dos mesmos segue o mesmo padrão de complexidade dos demais uc´s.

Breve descrição das funcionalidades dos Casos de Uso Significativos:

* **Abrir Caixa:**

Neste caso de uso é realizado a abertura do caixa nos dias de abertura do estabelecimento, somente após a conclusão do fluxo principal deste caso de uso, é possível a realização do UC – Realizar Pedidos;

* **Fechar Caixa:**

Neste caso de uso é realizado o fechamento diário do estabelecimento, com demonstração de lucro e efetivo financeiro.

* **Manter Cartão:**

UC responsável pela entrada e saída do cliente, somente será possível a emissão de pedidos após a realização da entrada do cliente, este caso de uso está intimamente ligado com a realização de pedidos, sendo que este último só poderá ocorrer após o registro do cartão de entrada no sistema;

* **Realizar Pedidos:**

Este caso de uso poderá ser realizado tanto pelo usuário(s) gestão como o usuário garçom. UC responsável por armazenar e gerir os pedidos realizados pelos clientes do estabelecimento;

* **Manter Estoque:**

Caso de uso responsável por manter a instabilidade do estoque do estabelecimento, sendo possível a inclusão e edição das quantidades de produtos disponíveis. A cada realização de pedidos efetuadas pelo caso de uso Realizar Pedidos, é necessário a baixa dos produtos, pela quantidade referente;

**Visão Lógica**

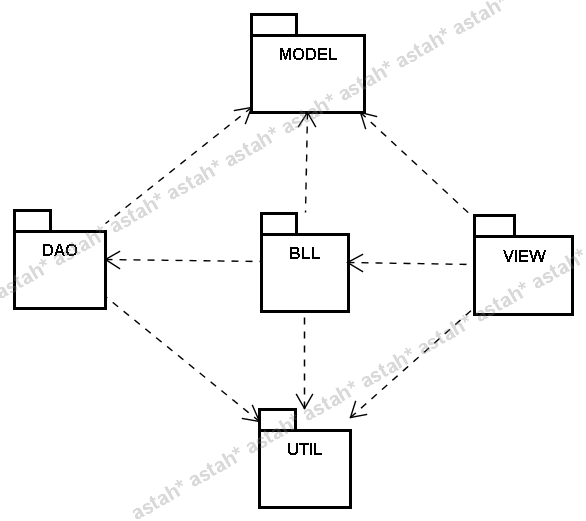
* **Descrição**

O sistema será desenvolvido utilizando a plataforma .NET, mais especificamente C# para controlar regras de negócio (Core) da aplicação e ASP MVC 4, HTML 5, JavaScript e CSS 3 responsáveis pela interface (FrontEnd). Toda comunicação com o CORE da aplicação dará através de uma camada externa (WCF), que terá como principal função o encapsulamento dos métodos.

O armazenamento das informações será através de banco de dados Microsoft Sql Server 2012 Express.

**Pacotes e Subsistema**

Visão responsável por demonstrar de forma macro todos os projetos de sistema;



* **Model:** Responsável por conter todos os objetos (modelos) que irão percorrer o sistema como um todo (exceto a DAO);
* **Entity:** Responsável por conter todos os objetos (modelos) de banco, os objetos aqui presentes deverão ser cópias exatas da estrutura do banco de dados;
* **DAO (Data Access Object):** Responsável pelo acesso a banco de dados (consultas, inserções, edições e exclusões).
* **BLL (Business Layer Logic):** Camada responsável por conter todas as regras de negócio;
* **WCF :** Camada responsável pela comunicação das aplicações de comunicação com usuário e a camada de regra de negócio.
* **VIEW:** Camada responsável pela interface com usuário, e transporte das informações a camada de abstração (WCF);
* **UTIL:** Camada responsável por conter objetos e funções úteis a todas as camadas do sistema, validações, verificações, conversores e etc;

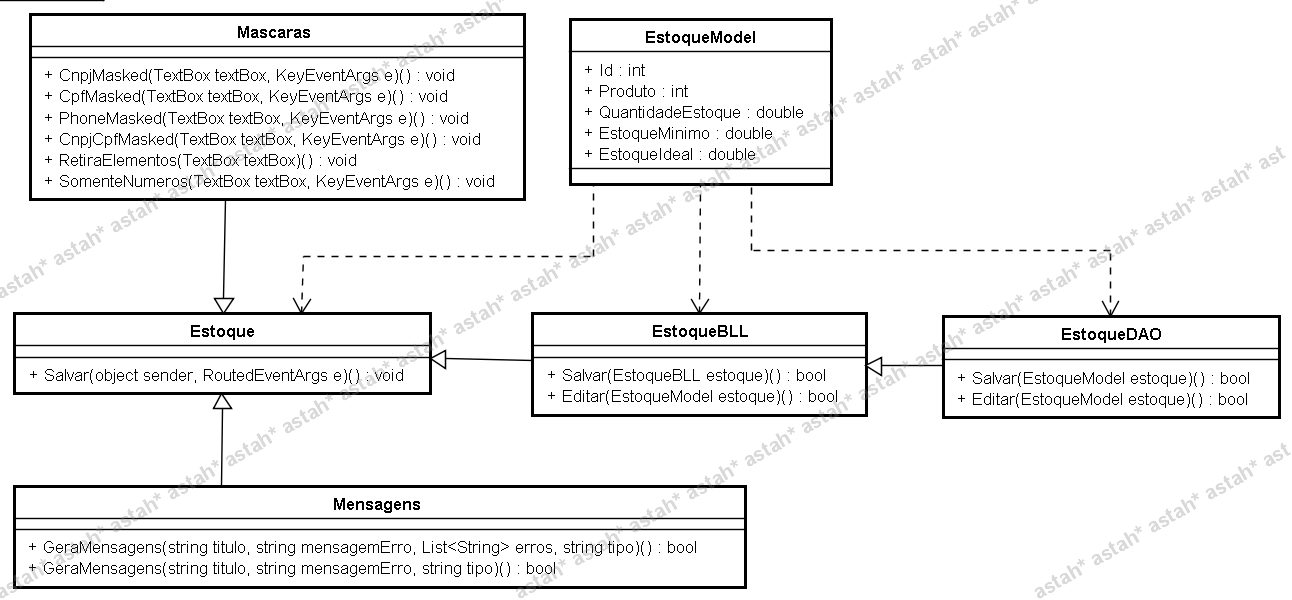
**Realização dos Casos de Uso Significativos**

As realizações de caso de uso aqui descritos somente serão demonstrados os fluxos principais, sendo responsabilidade dos documentos de especificação de caso de uso a demonstração dos demais fluxos (alternativos e exceção).

Todos os casos de uso deverão ser estruturados conforme pastas referenciadas no arquivo *HMA\_Project.juth (em anexo),* sendo desnecessário a diagramação de pacotes nesta fase documento.

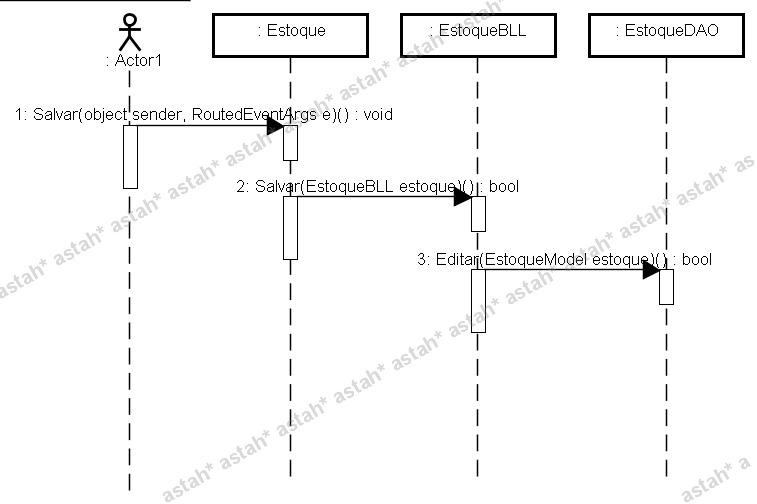
**[Manter Estoque]**

**Visão Estática**

****A visão estática é dada através dos diagramas de classe a seguir

**Visão Dinâmica**

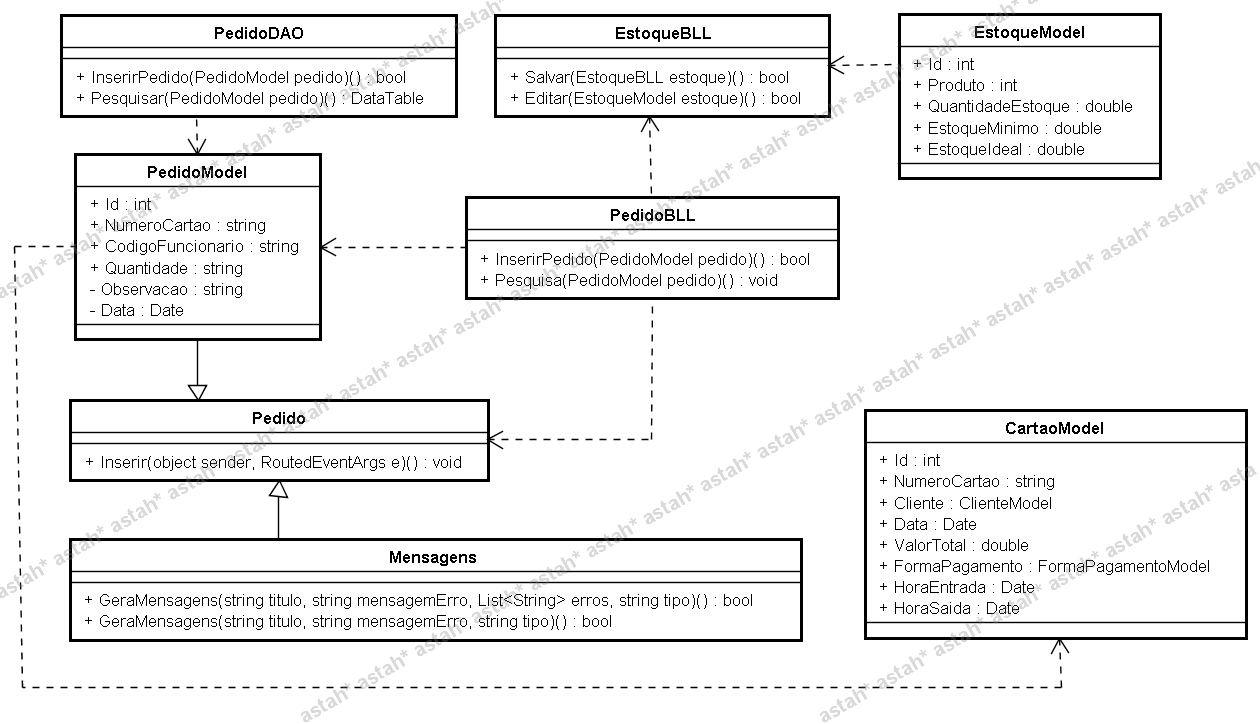
A visão dinâmica é dada através do diagrama de sequência a seguir (Fluxo Principal)



**[Efetuar Pedido]**

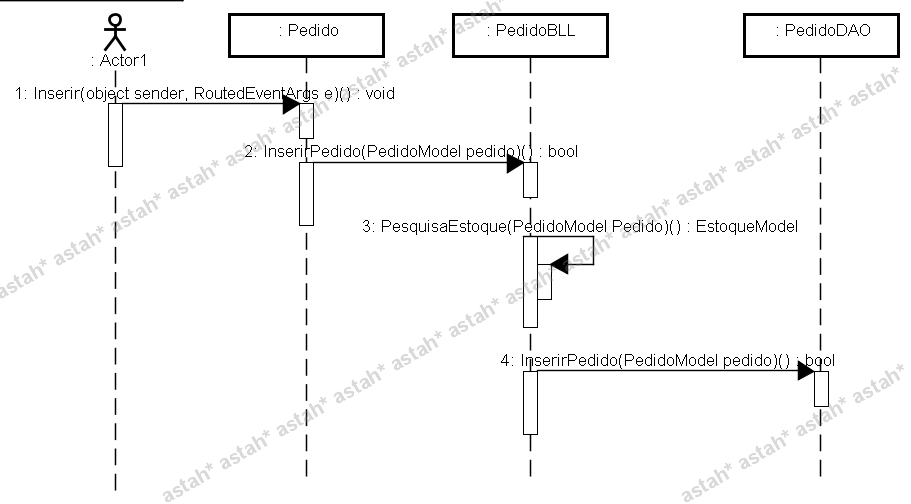
**Visão Estática**

A visão estática é dada através dos diagramas de classe a seguir

****

**Visão Dinâmica**

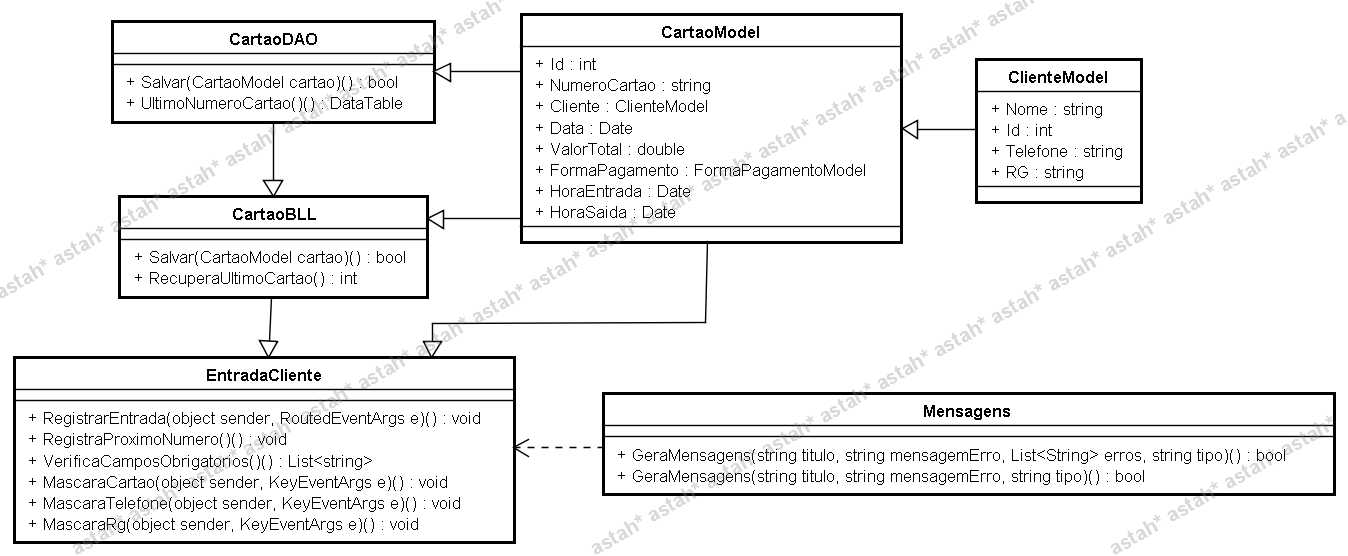
A visão dinâmica é dada através do diagrama de sequência a seguir (Fluxo Principal)



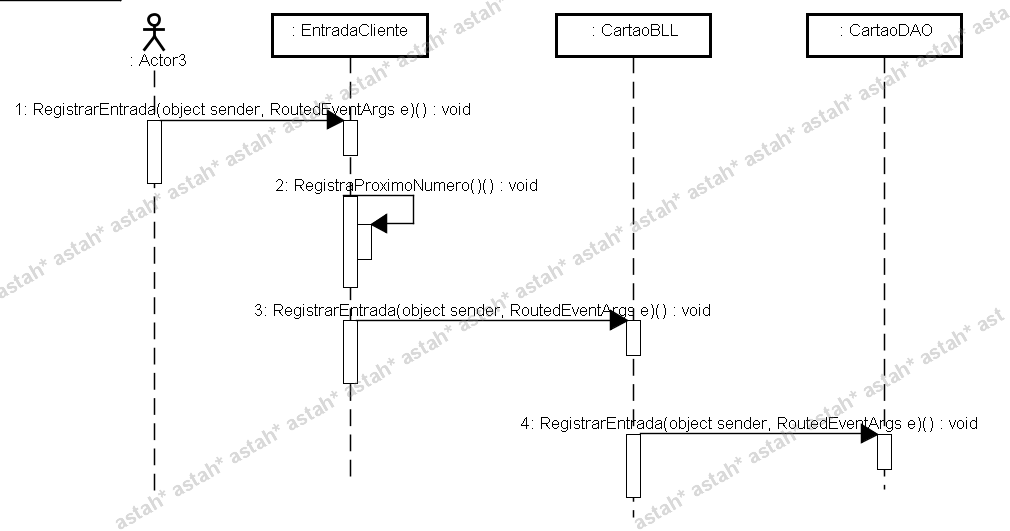
**[Manter Cartão]**

**Visão Estática**

A visão estática é dada através dos diagramas de classe a seguir

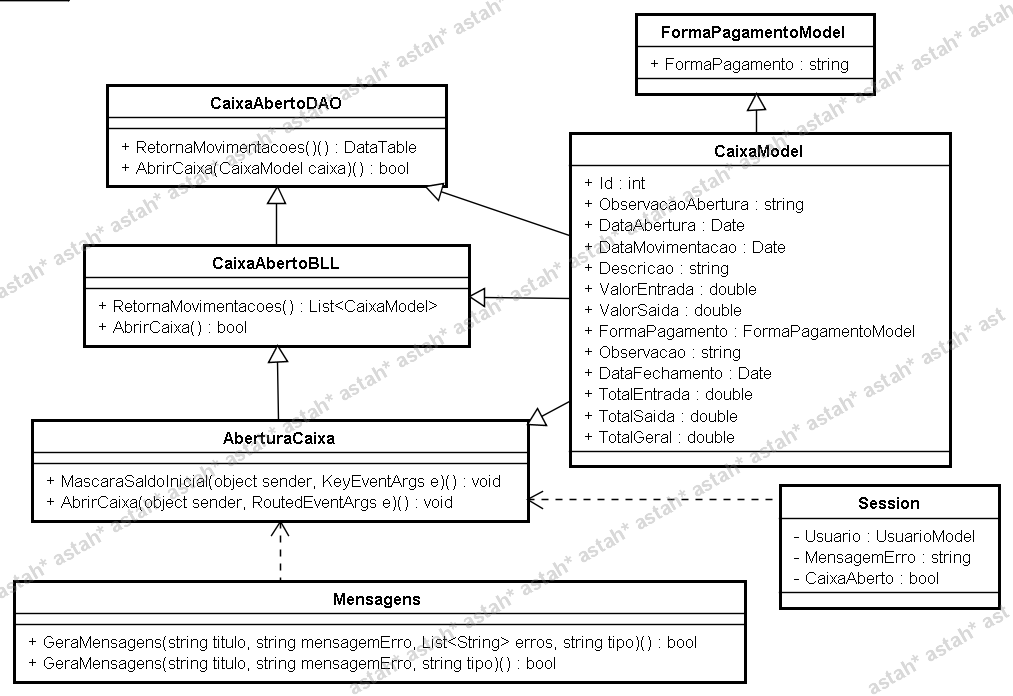


**Visão Dinâmica**

A visão dinâmica é dada através do diagrama de sequência a seguir (Fluxo Principal)

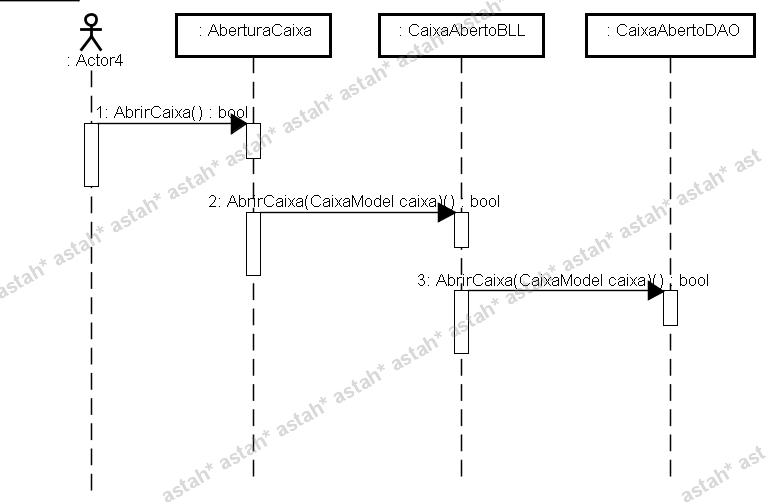
**[Abrir Caixa]**

**Visão Estática**

****A visão estática é dada através dos diagramas de classe a seguir

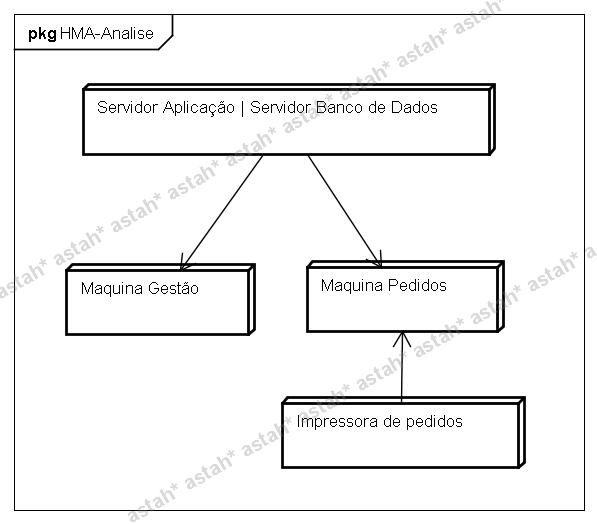
**Visão Dinâmica**

A visão dinâmica é dada através do diagrama de sequência a seguir (Fluxo Principal)



**Visão de implantação**

A implantação do HMA, deverá ser realizado conforme diagrama abaixo.



Entendemos que a separação dos servidores de aplicação e banco de dados seria de grande valia, tanto pela segurança quanto pela manutenção dos mesmos. Porém, por se tratar de uma aplicação com baixa quantidade de requisições simultâneas, e pelo estabelecimento não possuir estrutura suficiente para suportar tal divisão, acreditamos que a utilização de um único servidor multi-core, poderá suprir as necessidades iniciais de nosso cliente.

**Visão de Dados**

Por utilizarmos a camada ENTITY, todas as classes são cópias dos objetos de banco. Por este motivo, torna-se desnecessário a diagramação de tabelas. No arquivo *HMA\_Project.juth (em anexo),* a camada *Model* está disponível para consulta.

**Tamanho e Performance**

O sistema em sua totalidade deverá suportar 10 acessos simultâneos, sendo que em média 4 acessos serão realizados. Por não haver armazenamento de arquivos no banco de dados, a quantidade de informações (*em sua grande maioria VARCHAR*) armazenadas diariamente não possui relevância em relação a utilização do BD.

**Qualidade**

* **Características Positivas**
  + A instanciação de regras de negócio criadas diretamente em procedures favorece a manutenção da aplicação, não havendo necessidade de parar a utilização para realizar correções e/ou pequenas atualizações;
  + O desenvolvimento de ferramentas próprias também favorece caso futuramente alguma aplicação de terceiro deixe de prestar manutenção e/ou suporte;
  + O desenvolvimento de um WCF favorece o desenvolvimento para novas plataformas, não ficando restrito a apenas uma tecnologia;
* **Características Negativas**
  + A criação de regra de negócio em banco de dados possui um ponto negativo. A necessidade de profissionais específicos, faz com que o custo de manutenção ou seja superior de uma aplicação com toda regra em código;
  + Como os arquivos de NFe e emissão de recibos serão salvas no próprio servidor, caso haja qualquer problema no hardware de armazenamento, o risco de perda de arquivos é alto. Sendo necessário backup manual dos mesmos.