UNIVERSIDADE DE BRASÍLIA – UNB
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

A utilização de AJAX - "Asynchronous JavaScript and XML" - para melhoria da usabilidade de interfaces complexas.

Alberto Campos Siqueira
e
Wanderley G. Freitas

BRASÍLIA – DF
2006

ALBERTO CAMPOS SIQUEIRA

e

WANDERLEY G. FREITAS

A utilização de AJAX
- "Asynchronous JavaScript and XML" -
para melhoria da usabilidade de interfaces complexas.

Monografia apresentada à Universidade de Brasília como requisito para conclusão do Curso de Especialização em Ciência da Computação: Desenvolvimento de Sistemas Distribuídos com Orientação a Objetos.

Orientador: Prof. Dr. Hervaldo Carvalho

BRASÍLIA – DF
2006

SUMÁRIO

ÍNDICE DE FIGURAS

SIGLAS

AJAX	:	Asynchronous JavaScript and XML
XML	:	*Extensible Markup Language*
HTML	:	*Hipertext Markup Language*
MVC	:	*Model-View Controller*
DOM	:	*Document Object Model*
XSLT	:	*Extensible Stylesheet Language Transformations*
XHTML	:	*Extensible HyperText Markup Language*
CSS	:	*Cascading Style Sheets*
ECMA	:	*European Computer Manufacturers Association.*

1. INTRODUÇÃO

AJAX é a sigla de “Asynchronous JavaScript and XML”. Não se trata de uma tecnologia, mas sim do uso combinado de várias tecnologias, que juntas permitem a criação de aplicações com um alto grau de interatividade, resultando em aplicações que oferecem recursos similares aos encontrados em plugins, applets ou até mesmo em aplicações desktop. Ajax é basicamente uma técnica para construção de interface rica, onde a lógica de interação é executada no navegador. As tecnologias utilizadas são:

Apresentação usando XHTML e CSS;
Exibição e interação usando DOM;
Manipulação de dados usando XML e XSLT;
Comunicação assíncrona de dados usando XMLHttpRequest;
Programação usando JavaScript. [1,2,6]

1.1. Motivação

No modelo tradicional, uma aplicação web faz requisições HTTP ao servidor, que processa a solicitação manipulando os dados, banco de dados e sistemas legados, para então retornar uma nova página HTML para o solicitante. Na verdade, esse é um modelo adaptado do uso original da Web baseado em páginas de hipertexto, o que não é necessariamente bom para aplicações de software.

Apesar desse modelo uma das razões do sucesso da Internet, ele limita as possibilidades de interação do usuário com a interface, pois enquanto o servidor está fazendo o processando das requisições, o usuário precisa esperar a resposta, uma vez que a comunicação se dá de forma síncrona alternando entre o cliente e o servidor.

Já no modelo de aplicação Ajax, o processo de comunicação é assíncrono e transparente ao usuário. Uma camada adicional acomoda um controlador de interações escrito em Javascript que se encarrega de capturar os eventos do DOM, fazer requisições ao servidor usando o XMLHttpRequest, manipular o objeto de resposta em XML e atualizar as partes do HTML que se fizerem necessário. [6]

Esse modelo é similar ao que se processa em uma interface desktop tradicional, contudo persistem questões inerentes a Web, tais como o tempo de resposta, indisponibilidade de serviço e segurança. Os principais desafios em criar aplicações Ajax não são técnicas, pois tanto as limitações quanto as tecnologias utilizadas já são bem conhecidas. Na verdade, o desafio está em se libertar das chamadas limitações web e começar a imaginar todo o tipo de possibilidades.

1.2. Hipótese

Nesta dissertação será verificada a seguinte hipótese:

É possível desenvolver sistemas baseados em arquitetura WEB que dependam de interfaces complexas e com alto grau de interatividade, mantendo-se a usabilidade, manutenibilidade e desempenho em níveis satisfatórios.

1.3. Objetivos

1.3.1. Objetivo Geral

Realizar um estudo sobre a arquitetura e tecnologias envolvidas no desenvolvimento WEB com AJAX.

1.3.2. Objetivos Específicos

- Realizar estudo sobre as tecnologias envolvidas no desenvolvimento de aplicações AJAX;
- Realizar um comparativo entre o desenvolvimento tradicional e com AJAX;
- Desenvolver um protótipo em AJAX.
- Avaliar as vantagens e desvantagens do usa dessa técnica.

1.4. Relevância

O desenvolvimento de aplicações complexas baseadas na arquitetura Web tem se tornado cada vez freqüentes, contudo o desenvolvimento clássico apresenta tem se mostrado inadequado para o desenvolvimento de aplicações complexas que dependam de um grande volume de interações, tais como os visualizadores de mapas.

1.5. Organização da Dissertação

Esta dissertação foi dividida em cinco capítulos, visando alcançar os objetivos propostos incluindo o desenvolvimento de um protótipo para demonstrar o potencial da técnica. A seguir, cada um dos capítulos é descrito sucintamente:

O primeiro capítulo é introdutório, que contextualiza esta monografia e apresenta a motivação desta pesquisa, os objetivos que são esperados e a sua relevância.

O segundo capítulo trata da apresentação da técnica AJAX “Asynchronous JavaScript and XML” para o desenvolvimento de aplicações dinâmicas e dasdiferenças para um desenvolvimento tradicional.

O terceiro capítulo apresenta dos padrões: HTML (HyperText Markup Language), CSS (Cascading Style Sheets), XML (eXtensible Markup Language), e a linguagem JavaScript, que juntos formam a base tecnológica para a implementação em AJAX

O sexto capítulo apresenta um estudo de caso onde foi implementado um protótipo de solução na arquitetura web com AJAX visando o uso de geoprocessamente no controle de solicitação de ambulância do HUB.

O último capítulo, traz as conclusões obtidas nesta pesquisa, apresentando ainda recomendações para futuros trabalhos.

2. AJAX "ASYNCHRONOUS JAVASCRIPT AND XML"

AJAX é o acrônimo em língua inglesa de Asyncronous Javascript And XML, que significa o uso de Javascript e XML para tornar o navegador mais interativo com o usuário. AJAX é um novo modelo para a construção de aplicações web mais dinâmicas e criativas. Contudo, AJAX não é uma tecnologia, são na verdade várias tecnologias combinadas, cada uma fazendo sua parte, oferecendo novas funcionalidades. [6,28]

A apresentação é baseada nos padrões HTML e CSS, o DOM é a API responsável pelo acesso à interface com o usuário, o XML e seus derivados são utilizados como padrão para a comunicação entre os processos, essa comunicação é feita de forma assíncrona através do objeto XMLHttpRequest e, finalmente, o JavaScript é a linguagem que controla toda a interação. [2]

No modelo clássico, a aplicação web trabalha de forma que a maioria das ações do usuário na interface dispare uma requisição HTTP para o servidor [6]. O servidor processa algo e então retorna uma nova página HTML para o cliente. É um modelo adaptado do uso original da Web como um agente de hipertexto, o que não é necessariamente boa para aplicações. [21]

A maior vantagem das aplicações AJAX é que elas permitem que um controlador rode no próprio navegador web. Então, para estar hábil a executar aplicações AJAX, bastar possuir algum dos navegadores modernos, ou seja, lançados após 2001. São eles: Mozilla Firefox, Internet Explorer 5+, Opera, Konqueror e Safari. [1]

2.1.1. Aplicação versus Conteúdo

Em uma aplicação web clássica baseada em páginas, o navegador é efetivamente um terminal burro. Ele não sabe nada sobre o que o usuário está realmente realizando em suas ações conseqüentes. Todas essas informações são retidas no servidor web, tipicamente na sessão do usuário. Se você está trabalhando em Java ou .NET, a sessão no lado servidor faz

parte da API padrão, assim como controle de solicitações, respostas, e tipos de conteúdo (MIME). [6]

Quando o usuário entra ou de outra maneira inicia uma sessão, vários objetos são criados no lado servidor, representando, por exemplo, a cesta de compras e as credenciais de cliente, isso em um site de comércio eletrônico. Ao mesmo tempo, a página inicial é servida ao navegador, em um fluxo de marcações HTML que mistura um anúncio de apresentação padrão e dados específicos do usuário juntos com o conteúdo, como por exemplo, uma lista de itens exibidos recentemente.

Toda vez que o usuário interage com o site, um outro documento é enviado para o navegador, contendo a mesma mistura de cabeçalhos e dados. Então o navegador retira o documento anterior e exibe o novo, independentemente da semelhança, nesse momento apenas o cachê do navegador pode fazer alguma pequena diferença.

Quando o usuário efetua a saída ou fecha o navegador, a aplicação sai e a sessão é destruída. Qualquer informação que o usuário necessite ver na próxima vez que ele entrar terá que ser passada para a camada de persistência de dados em cada visita.

Em uma aplicação Ajax, parte da lógica da aplicação é movida para o navegador. Neste novo cenário, quando o usuário entra, um documento mais complexo é entregue ao navegador, uma grande proporção do qual é um controlador escrito em JavaScript. Este documento permanecerá com o usuário por toda a sessão. Ele sabe como responder às informações inseridas pelo usuário e é capaz de decidir se manipula a entrada do usuário ele mesmo ou se passa uma solicitação para o servidor web (o qual tem acesso ao banco de dados do sistema e outros recursos), ou ainda, se faz uma combinação de ambos.

Ele também pode armazenar o estado, porque o documento continua persistindo sobre toda a sessão do usuário. Por exemplo, o conteúdo de uma cesta de compras pode ser armazenado no navegador, em vez de ser armazenado na sessão do servidor.

2.1.2. O servidor fornece dados, e não conteúdo.

Como observamos, uma aplicação web clássica oferece a mesma mistura de alegorias, conteúdos e dados em todos os passos. Quando nosso usuário adiciona um item na cesta de compras, tudo que precisamos realmente é responder com o valor atualizado da cesta ou informar se alguma coisa deu errada.

Um carrinho de compras baseado em Ajax pode comportar-se de forma mais esperta, por meio de remessas de solicitações assíncronas ao servidor. O cabeçalho, o histórico de navegação, e outras características do layout da página estão todas carregadas, portanto o servidor necessita enviar de volta somente os dados relevantes.

Uma aplicação Ajax poderia fazer isto de vários modos, como, por exemplo, devolver um fragmento de JavaScript, um fluxo de texto simples, ou um pequeno documento XML Em uma aplicação Ajax, o tráfego tem sua maior intensidade no início, com um largo e complexo cliente sendo entregue em uma única explosão, quando o usuário entra. As comunicações subseqüentes com o servidor são muito mais eficientes, de qualquer forma. Para uma aplicação breve, o tráfego cumulativo pode ser menor em uma aplicação de página web convencional. Mas conforme o tamanho médio do tempo de interação aumentar, o custo de largura de banda da aplicação Ajax se torna menor do que sua aplicação clássica equivalente.

2.1.3. Comunicação continua e flexível

Um navegador web oferece duas maneiras de enviar entradas de dados para um outro computador: com os hyperlinks e formulários HTML.

Os hyperlinks podem ser carregados com parâmetros CGI (Common Gateway Interface – Interface de Comunicação Comum) apontando para páginas dinâmicas ou servlets. Eles podem estar vinculados com imagens e folhas de estilo (CSS) para oferecer uma pequena melhoria na interface, como por exemplo, definir efeitos quando o mouse estiver sobre eles. [2,6]

Os controles de formulário oferecem um subconjunto básico de componentes padrões de interface com o usuário: caixas de texto, caixas de checagem e botões de rádio, além de listas de seleção. Entretanto estes controles não são suficientes. Não existem controles de seleção em árvores, grades para edição, ou caixas de combinação. Os formulários, assim como os hyperlinks, apontam para URLs resididas no servidor.

Alternativamente, os hyperlinks e os controles de formulário podem ser apontados para funções JavaScript. Isto é uma técnica comum em páginas web para prover uma validação de formulário rudimentar em JavaScript, verificando por campos vazios, valores fora de intervalo, e assim por diante, antes de submeter os dados para o servidor. Estas funções JavaScript existem somente enquanto a própria página existe e é substituída quando a página efetuar o seu envio.

Enquanto a página está sendo enviada, o usuário aguarda a sua resposta. A página anterior pode ainda estar visível por algum tempo, e o navegador pode até permitir que o usuário clique em qualquer um dos links visíveis, mas se assim for feito, produzirá resultados imprevisíveis e até entornar em uma confusão com a sessão no servidor. O usuário está normalmente aguardando a página ser atualizada que, freqüentemente, possuem quase que as mesmas informações que lhes foram apanhadas instantes atrás. Adicionando um par de calças à cesta de compras não é razoável modificar as categorias em um nível acima por “roupas masculinas”, “roupas femininas”, “infantis” e “acessórios”.

Devido ao fato de que uma aplicação web em Ajax poder enviar dados assincronamente, os usuários podem soltar os objetos dentro dele tão rápido quanto eles podem clicar. Se o código de nosso carrinho no lado cliente for robusto, ele tratará esta tarefa facilmente, e os usuários podem continuar com o que eles estão fazendo.

A troca de contextos é uma distração para usuários, pois aguardar que uma página seja atualizada levará o usuário à realidade de estar sentado em um computador a espera de algo que esta sendo feito em outro lugar, a utilização de Ajax permite que isso seja evitado pelo meio de um fluxo continuo de pequenas chamadas ao servidor de forma transparente .

A segunda vantagem de Ajax é que podemos associar eventos a um maior número de ações do usuário. Os conceitos mais sofisticados de interface com o usuário, assim como “arrastar e soltar”, se tornam praticáveis, trazendo as experiências dessas interfaces em pé de igualdade com os controles de aplicações desktop. Da perspectiva de usabilidade, esta liberdade não é importante somente porque ela permite exercer nossa imaginação, mas porque ela nos permite combinar a interação do usuário e as solicitações ao servidor de maneira mais completa.

Para comunicar com o servidor em uma aplicação web clássica, necessitamos clicar em um hyperlink ou submeter um formulário, e então aguardar. No entanto, este método interrompe a interação com o usuário. Em contraste, a possibilidade de se comunicar com o servidor em resposta a um movimento ou arraste do mouse, ou até quando digitamos, habilita o servidor a trabalhar juntamente com o usuário. Google Suggest é um exemplo muito simples e efetivo disto: responder às teclas pressionadas enquanto ele digita dentro da caixa de pesquisa, e então, comunicar com o servidor para recuperar e exibir uma lista de possíveis finalizações para as expressões, baseada nas pesquisas feitas por outros usuários do mecanismo de busca em todo o mundo.

2.1.4. Boas práticas

Neste momento, as clássicas aplicações web fazem uso de JavaScript em certas ocasiões, para adicionar características avançadas e exageradas de um programa, as agregando nas páginas. O modelo baseado em páginas previne qualquer uma destas melhorias que consista em um atraso longo demais, o qual limita as utilidades para as quais elas podem ser oferecidas. Isto fez com que JavaScript recebesse injustamente, uma reputação de algo banal – por má sorte da linguagem – e não sendo bem vista pelos desenvolvedores sérios.

Codificar uma aplicação Ajax é algo completamente diferente. O código que você fornece quando os usuários iniciam a aplicação deve executar até que eles encerrem-na, sem interrupção, sem diminuição de velocidade, e sem produção de escapes de memória. Se estivermos mirando no mercado de aplicações poderosas, então temos em vista muitas horas de intenso uso. Para atingir este objetivo é necessário escrever códigos de alto-desempenho, e

manuteníveis, usando a mesma disciplina e entendimento que é aplicado com sucesso às camadas do servidor.

A base de código será tipicamente mais ampla que qualquer código escrito para uma aplicação web clássica. Boas práticas na construção da base de código se tornam muito importante. O código deve tornar-se, de preferência, responsabilidade de uma equipe do que apenas um indivíduo, criando edições de manutenibilidade, separações de interesses, e estilos e padrões de codificação comum. Uma aplicação Ajax, portanto, é uma porção de código funcionalmente complexa que comunica eficientemente com o servidor enquanto o usuário continua com seu trabalho. Ela é claramente uma descendência da aplicação clássica baseada em páginas, mas a similaridade não é mais forte do que entre um cavalinho de madeira e uma moderna bicicleta de passeio. Sustentando essas diferenças em mente nos ajudará a criar aplicações web verdadeiramente convincentes. [6,21,28]

3. Padrões utilizados em AJAX

3.1. HTML (HyperText Markup Language)

O HTML (HyperText Markup Language) é uma linguagem de formatação de hipertexto fruto do união dos padrões HyTime e SGML. [11,18]

HyTime (Hypermedia/Time-based Document Structuring Language - ISO 10744:1992) - é um padrão para representação estruturada de hipermídia e informação baseada em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (áudio, vídeo, etc.), conectados por webs ou hiperlinks. O padrão HyTime é independente dos padrões de processamento de texto em geral. Ele fornece a base para a construção de sistemas hipertexto padronizados, consistindo de documentos que ligam os padrões de maneira particular. [11]

SGML (Standard Generalized Markup Language - ISO 8879) é um padrão de formatação de textos, ele não foi desenvolvido para hipertexto, mas tornou-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações. O SGML não é um padrão aplicado de maneira padronizada, sendo que todos os produtos SGML têm seu próprio sistema para traduzir as etiquetas para um particular formatador de texto. Um DTD (Document Type Definition) ou uma referência para um DTD deve estar contido em qualquer documento conforme o padrão SGML. [11]

Portanto, o HTML é definido segundo um DTD de SGML.

Todo documento HTML apresenta elementos entre parênteses angulares (< e >); esses elementos são as etiquetas (tags) de HTML, que são os comandos de formatação da linguagem. A maioria das etiquetas tem sua correspondente de fechamento:

<etiqueta>...</etiqueta>

Isso é necessário porque as etiquetas servem para definir a formatação de uma porção de texto, e assim marcamos onde começa e termina o texto com a formatação especificada por ela.

Alguns elementos são chamados “vazios”, pois não marcam uma região de texto, apenas inserem alguma coisa no documento:

<etiqueta>

Todos os elementos podem ter atributos:

<etiqueta atributo1=valor1 atributo2=valor2>...</etiqueta>

HTML é um recurso muito simples e acessível para a produção de documentos.

3.1.1. Formatação

Há dois tipos de formatação em HTML: lógico e físico. Os efeitos de apresentação na tela são os mesmos: o motivo da distinção entre eles se deve à idéia básica de independência entre especificação e apresentação.

Quando formatamos um trecho de texto como cabeçalho de nível 1, não explicitamos se esse tipo de cabeçalho deve ser em alguma fonte determinada, em um tamanho determinado, justificado à esquerda ou à direita, ou centralizado. Esses detalhes de apresentação são deixados para o browser - o dispositivo de apresentação do documento - que pode ser configurado de acordo com o leitor (usuário final).

Desse modo, além de facilitar enormemente o trabalho de quem escreve os documentos, a linguagem garante a uniformidade de apresentação de cabeçalhos, parágrafos, listas, etc.

A formatação lógica segue o significado lógico do texto marcado: um endereço de e-mail, uma citação etc. Sua apresentação final varia conforme o browser, podendo oferecer resultados mais ricos.

A formatação física especifica explicitamente o estilo que se quer para o texto: itálico, grifado etc. Sua apresentação final não sofre grandes variações.

3.1.2. Ligações (“links”)

Com HTML é possível fazermos ligações de uma região de texto (ou imagem) a um outro documento. Nestas páginas, temos visto exemplos dessas ligações: o browser destaca essas regiões e imagens do texto, indicando que são ligações de hipertexto - também chamadas hypertext links ou hiperlinks ou simplesmente links.

Para inserir um link em um documento, utilizamos a etiqueta <A>, da seguinte forma:

```
<A HREF = "arq_destino">âncora</A>
```

onde:

arq_destino é o URL do documento de destino;

âncora é o texto ou imagem que servirá de ligação hipertexto do documento sendo apresentado para o documento de destino.

Atributos

<A> tem vários atributos, utilizados de acordo com a ação associada ao link. Os mais usados são:

HREF

Indica o arquivo de destino da ligação de hipertexto.

TARGET

Indica o frame em que será carregado o arq_destino. Maiores detalhes na seção sobre frames.

NAME

Marca um indicador, isto é, uma região de um documento como destino de uma ligação.

3.1.3. Imagens

O elemento IMG insere imagens que são apresentadas junto com os textos. Um atributo SRC deve estar presente, da seguinte forma:

```
<IMG SRC="URL_imagem">
```

onde URL_imagem é o URL do arquivo que contém a imagem que se quer inserir; pode ser referenciada uma imagem que esteja em um outro servidor (o que, logicamente, não é conveniente).

Assim, escrevendo:

```
<IMG SRC = "/icones/newred.gif">
```

inserimos a figura no documento.

As imagens usadas na Web são armazenadas em arquivos nos padrões gif, xbm, jpg , png, etc..

3.2. CSS (Cascading Style Sheets)

Um avanço interessante na linguagem HTML após a versão 3.2 foi a introdução das Style Sheets ou Cascading Style Sheets. Essas folhas de estilo permitem o uso de formatações uniformes em um site, de maneira reutilizável. [5,11]

Para se poder formatar um texto, é preciso escrever uma marcação parecida com:

```
<h3><font face="Arial" color="#4A7D7B">Um título genérico</font></h3>
<p><font face="Arial" size="2">Um texto genérico com algum </font><font face="Arial" size="2" color="red">destaque qualquer</font><font face="Arial" size="2"> junto, após um título genérico, etc.</font></p>
```

tendo como resultado:

Um texto genérico com algum destaque qualquer junto, após um título genérico, etc.

Com as folhas de estilo, podemos declarar estilos apropriados para seções de texto, aplicando esses estilos sem nos preocuparmos com detalhes de fontes, cor e tamanhos.

E mais: se for necessário modificar algum dia as cores de todos os títulos ou dos destaques ao longo dos textos, simplesmente alteramos a folha de estilo, atualizando imediatamente a apresentação de todos os documentos que fazem referência a ela.

Para o exemplo acima, poderíamos criar a seguinte folha de estilo:

```
BODY { font: 10pt Arial }
H3 { color:#4A7D7B }
.destaque { color: red }
```

E assim, para ter o mesmo resultado do exemplo acima, a formatação seria muito mais simples que a primeira:

```
<h3>Um título genérico</h3><p>Um texto genérico com algum <span class="destaque">destaque qualquer</span> junto, após um título genérico, etc.</p>
```

A definição da folha de estilo deve ficar dentro de <HEAD>, da seguinte forma, se a folha for definida dentro do mesmo documento em que está sendo usada:

```
<HEAD>
…
<STYLE TYPE="text/css">
    BODY { font: 10pt Arial }
    H3 { color:#4A7D7B }
    .destaque { color: red }
</STYLE>
...
</HEAD>
```

Ou então,quando a folha de estilo é definida externamente:

```
<HEAD>
...
<LINK REL="stylesheet" HREF="folha_de_estilo.css">
...
</HEAD>
```

Assim, vários documentos HTML podem usar uma mesma folha de estilo, que ainda poderão ter suas próprias folhas de estilo internas.

3.3. DOM (Documento Object Model)

O Modelo de Objeto de Documentos (DOM) é uma interface de programação de aplicativos (API) para documentos HTML e XML. É a definição da estrutura lógica dos documentos e o meio pelo qual um documento é acessado e manipulado. Na especificação DOM, o termo "documento" é utilizado no seu sentido mais amplo - XML é usado como o meio de representação de muitos tipos diferentes de informação que podem ser armazenados em sistemas diversos e muitos seriam tradicionalmente considerados como informação no lugar de documentos. Assim mesmo, o XML apresenta estas informações como documentos e o DOM pode ser utlizado para o gerenciamento destas informações. [11]

Com o Modelo de Objeto de Documentos é possível criar documentos, navegar pela sua estrutura e adicionar, modificar ou apagar elementos e conteúdo. Tudo o que seja encontrado em um documento HTML ou XML pode ser acessado, alterado, apagado ou adicionado através do Modelo de Objeto de Documentos com apenas algumas exceções - em particular as interfaces DOM para sub-conjuntos internos e externos XML ainda não foram especificados. [11]

No DOM, os documentos possuem uma estrutura lógica que é muito parecida com uma árvore. Contudo, o DOM não especifica como os relacionamentos entre os objetos são implementados. O DOM é um modelo lógico que pode ser implementado de qualquer forma conveniente. Nesta especificação é utilizado a terminologia *modelo de estrutura* para descrever a representação de um documento parecida com uma árvore. Uma propriedade importante do modelo de estrutura DOM é o *isomorfismo estrutural*: se quaisquer duas implementações de Modelo de Objeto de Documentos são utilizadas para criar uma representação de um mesmo documento elas irão criar um memos modelo de estrutura com exatamente os mesmos objetos e relacionamentos. [11]

O nome "Modelo de Objeto de Documentos" foi escolhido porque é um "modelo de objetos" no sentido tradicional de desenho orientado a objeto: documentos são modelados usando objetos e o modelo engloba não apenas a estrutura do documento mas também seu comportamento e os dos objetos que o compõem. Em outras palavras, os nós do diagrama acima não representam uma estrutura de dados, eles representam objetos que possuem funcionalidades e identidades. Como um modelo de objetos o DOM identifica:

- As interfaces e os objetos utilizados na representação e manipulação de um documento.
- As semânticas destas interfaces e objetos - incluindo-se o comportamento e atributos.
- Os relacionamentos e colaborações entre estas interfaces e os objetos.

A estrutura de documentos SGML é usualmente representada através de um modelo de dados abstrato e não por um modelo de objeto. No modelo de dados abstrato o modelo é centrado sobre dados. Nas linguagens de programação orientadas a objeto, os dados são encapsulados em objetos que escondem os dados, protegendo-os de manipulação externa direta. As funcionalidades associadas com estes objetos determinam como os objetos podem ser manipulados e são partes do modelo de objeto.

O Modelo de Objeto de Documentos consiste atualmente em duas partes: DOM Core e DOM HTML. O DOM Core representa a funcionalidade utilizada para documentos XML e também serve de base para o DOM HTML. Uma implementação em conformidade com o DOM necessita implementar todas as interfaces fundamentais no capítulo CORE com as semânticas conforme a definição. Além de que necessita implementar pelo menos um DOM HTML e as interfaces estendidas (XML) com as semânticas, conforme a definição. [11]

3.3.1. Origem do DOM

O DOM começou como uma especificação para permitir que scripts em JavaScript e programas em Java fossem portáveis entre browsers Web. O "HTML dinâmico" foi o antecessor imediato do Modelo de Objeto de Documentos e foi inicialmente concebido com uma idéia de browsers. Contudo, o Grupo de Trabalho DOM foi formado no W3C e contou com a colaboração de vendedores de outras áreas, incluindo editores de HTML e XML e repositórios de documentos. Muitos destes vendedores tinham trabalhado com SGML antes que o XML tivesse sido desenvolvido, assim o DOM recebeu influências dos standards SGML Groves e de HyTime. Alguns destes vendedores também desenvolveram os seus próprios modelos de objeto de documentos para o fornecimento de API para editores SGML/XML ou repositórios de documentos e estes modelos de objeto também influenciaram o DOM. [11]

3.3.2. Entidades

Nas interfaces fundamentais DOM não existem objetos representando entidades. As referências a caracteres numéricos e as referências a entidades pré-definidas em HTML e XML são substituidas por um único caracter que representa a substituição de entidade. Por exemplo, no caso seguinte :

```
<p>This is a dog &amp;amp; a cat</p>
```

a expressão "&" será substituida pelo caracter "&", e o texto no elemento P irá formar uma sequência contínua de caracteres. Como as referências a caracteres numéricos e entidades pré-definidas não são reconhecidas como tais nas seções CDATA, ou o SCRIPT e elementos STYLE em HTML, eles não podem ser substituidos por um único caracter que parecem fazer referência. Se o exemplo acima estivesse dentro de uma seção CDATA, a expressão "&" não seria substituida por "&"; nem o <p> seria reconhecido como um

início de instrução. A representação de entidades genéricas, tanto internas como externas, são definidas com as interfaces extendidas (XML) da especificação de Nível 1.

3.3.3. Objetos [11]

Object	Description
Anchor	Represents an HTML a element (a hyperlink)
Applet	Represents an HTML applet element. The applet element is used to place executable content on a page
Área	Represents an area of an image-map. An image-map is an image with clickable regions
Base	Represents an HTML base element
Basefont	Represents an HTML basefont element
Body	Represents the body of the document (the HTML body)
Button	Represents a push button on an HTML form. For each instance of an HTML <input type="button"> tag on an HTML form, a Button object is created
Checkbox	Represents a checkbox on an HTML form. For each instance of an HTML <input type="checkbox"> tag on an HTML form, a Checkbox object is created
Document	Used to access all elements in a page
Event	Represents the state of an event, such as the element in which the event occurred, the state of the keyboard keys, the location of the mouse, and the state of the mouse buttons
FileUpload	For each instance of an HTML <input type="file"> tag on a form, a FileUpload object is created
Form	Forms are used to prompt users for input. Represents an HTML form element
Frame	Represents an HTML frame
Frameset	Represents an HTML frameset
Hidden	Represents a hidden field on an HTML form. For each instance of an HTML <input type="hidden"> tag on a form, a Hidden object is created
History	A predefined object which can be accessed through the history property of the Window object. This object consists of an array of URLs. These URLs are all the URLs the user has visited within a browser window
Iframe	Represents an HTML inline-frame
Image	Represents an HTML img element
Link	Represents an HTML link element. The link element can only be used within the <head> tag

Object	Description
Location	Contains information about the current URL
Meta	Represents an HTML meta element
Navigator	Contains information about the client browser
Option	Represents an option in a selection list on an HTML form. For each instance of an HTML <option> tag in a selection list on a form, an Option object is created
Password	Represents a password field on an HTML form. For each instance of an HTML <input type="password"> tag on a form, a Password object is created
Radio	Represents radio buttons on an HTML form. For each instance of an HTML <input type="radio"> tag on a form, a Radio object is created
Reset	Represents a reset button on an HTML form. For each instance of an HTML <input type="reset"> tag on a form, a Reset object is created
Screen	Automatically created by the JavaScript runtime engine and it contains information about the client's display screen
Select	Represents a selection list on an HTML form. For each instance of an HTML <select> tag on a form, a Select object is created
Style	Represents an individual style statement. This object can be accessed from the document or from the elements to which that style is applied
Submit	Represents a submit button on an HTML form. For each instance of an HTML <input type="submit"> tag on a form, a Submit object is created
Table	Represents an HTML table element
TableData	Represents an HTML td element
TableHeader	Represents an HTML th element
TableRow	Represents an HTML tr element
Text	Represents a text field on an HTML form. For each instance of an HTML <input type="text"> tag on a form, a Text object is created
Textarea	Represents an HTML textarea element
Window	Corresponds to the browser window. A Window object is created automatically with every instance of a <body> or <frameset> tag

3.4. XML (Extensible Markup Language)

O XML (Extensible Markup Language) é linguagem de marcação de dados (meta-markup language) que provê um formato para descrever dados estruturados. Isso facilita declarações mais precisas do conteúdo e resultados mais significativos de busca através de múltiplas plataformas. O XML também vai permitir o surgimento de uma nova geração de aplicações de manipulação e visualização de dados via internet. [20]

O XML permite a definição de um número infinito de tags. Enquanto no HTML, se as tags podem ser usadas para definir a formatação de caracteres e parágrafos, o XML provê um sistema para criar tags para dados estruturados.

Um elemento XML pode ter dados declarados como sendo preços de venda, taxas de preço, um título de livro, a quantidade de chuva, ou qualquer outro tipo de elemento de dado. Como as tags XML são adotadas por intranets de organizações, e também via Internet, haverá uma correspondente habilidade em manipular e procurar por dados independentemente das aplicações onde os quais são encontrados. Uma vez que o dado foi encontrado, ele pode ser distribuído pela rede e apresentado em um browser como o Internet Explorer 5 de várias formas possíveis, ou então esse dado pode ser transferido para outras aplicações para processamento futuro e visualização.

3.4.1. Representação

O XML provê uma representação estruturada dos dados que mostrou ser amplamentemente implementável e fácil de ser desenvolvida. Implementações comerciais na linguagem SGML (Standard Generalized Markup Language) mostraram a qualidade intrínseca e a força do formato estruturado em árvore dos documentos XML. [20]

O XML é um subconjunto do SGML, o qual é otimizado para distribuição através da web, e é definido pelo Word Wide Web Consortium(W3C), assegurando que os dados estruturados serão uniformes e independentes de aplicações e fornecedores. [20]

O XML provê um padrão que pode codificar o conteúdo, as semânticas e os esquemas para uma grande variedade de aplicações, dentre elas:

- Um simples documento.
- Um registro estruturado tal como uma ordem de compra de produtos.
- Um objeto com métodos e dados como objetos Java ou controles ActiveX.
- Um registro de dados. Um exemplo seria o resultado de uma consulta a bancos de dados.
- Apresentação gráfica, como interface de aplicações de usuário.
- Entidades e tipos de esquema padrões.
- Todos os links entre informações e pessoas na web.

Uma característica importante é que uma vez que tendo sido recebido o dado pelo cliente, tal dado pode ser manipulado, editado e visualizado sem a necessidade de acionar o servidor. Dessa forma, os servidores têm menor sobrecarga, reduzindo a necessidade de computação e reduzindo também a requisição de banda passante para as comunicações entre cliente e servidor.

O XML é considerado de grande importância na Internet porque provê a capacidade de inter-operação dos computadores por ser um padrão flexível, aberto e independente de dispositivo. As aplicações podem ser construídas e atualizadas mais rapidamente e também permitem múltiplas formas de visualização dos dados estruturados.

3.4.2. Separação

A mais importante característica do XML se resume em separar a interface com o usuário (apresentação) dos dados estruturados. O HTML especifica como o documento deve ser apresentado na tela por um navegador. Já o XML define o conteúdo do documento. Por exemplo, em HTML são utilizadas tags para definir tamanho e cor de fonte, assim como formatação de parágrafo. No XML você utiliza as tags para descrever os dados, como exemplo tags de assunto, título, autor, conteúdo, referências, datas, etc...

O XML ainda conta com recursos tais como Extensible Style Language (XSL) e Cascading Style Sheets(CSS) para transformações e apresentação de dados em um navegador. O XML separa os dados da apresentação e processo, o que permite visualizar e processar o dado como quiser, utilizando diferentes folhas de estilo e aplicações.

3.4.3. XML versus HTML

O HTML e XML derivam do SGML. Ambos identificam elementos em uma página e ambos utilizam sintaxes similares. Se você é familiar com HTML, também o será com o XML. A grande diferença entre HTML e XML é que o HTML descreve a aparência e a ações em uma página na rede enquanto o XML não descreve nem aparência e ações, mas sim o que cada trecho de dados é ou representa ! Em outras palavras, o XML descreve o conteúdo do documento.

Como o HTML, o XML também faz uso de tags (palavras encapsuladas por sinais '<' e '>') e atributos (definidos com name="value"), mas enquanto o HTML especifica cada sentido para as tags e atributos (e freqüentemente a maneira pela qual o texto entre eles será exibido em um navegador), o XML usa as tags somente para delimitar trechos de dados, e deixa a interpretação do dado a ser realizada completamente para a aplicação que o está lendo. Resumindo, enquanto em um documento HTML uma tag <p> indica um parágrafo, no XML essa tag pode indicar um preço, um parâmetro, uma pessoa, ou qualquer outra coisa que se possa imaginar.

Os arquivos XML são arquivos texto, mas não são tão destinados à leitura por um ser humano como o HTML é. Os documentos XML são arquivos texto porque facilitam que os programadores ou desenvolvedores "debuguem" mais facilmente as aplicações, de forma que um simples editor de textos pode ser usado para corrigir um erro em um arquivo XML. Mas as regras de formatação para documentos XML são muito mais rígidas do que para documentos HTML. Uma tag esquecida ou um atributo sem aspas torna o documento inutilizável, enquanto que no HTML isso é tolerado. As especificações oficiais do XML determinam que as aplicações não podem tentar adivinhar o que está errado em um arquivo (no HTML isso acontece), mas sim devem parar de interpretá-lo e reportar o erro.

3.5. JavaScript

A primeira versão de JavaScript foi criada por Brendan Eich na Netscape, e ela tem sido atualizada desde então visando conformidade com o padrão ECMA-262 revisão 3 (ECMAScript também conhecido como JS 1.5). Este mecanismo, nome código SpiderMonkey, está implementado em C. O mecanismo Rhino, criado primariamente por Norris Boyd (também na Netscape) é uma implementação JavaScript em Java, também em conformidade com ECMA-262 rev. 3. [9]

JavaScript é uma linguagem que permite injetar lógica em páginas escritas em HTML (HiperText Mark-up Language). As páginas HTML podem ser escritas utilizando-se editores de texto, como o NotePad, Write, etc. Porém, existem editores próprios para gerar HTML, tais como HotDog e (mais recomendado) Microsoft FrontPage. Os parágrafos de lógica do javaScript podem estar "soltos" ou atrelados a ocorrência de eventos. Os parágrafos soltos são executados na sequência em que aparecem na página (documento) e os atrelados a eventos são executados apenas quando o evento ocorre. [9,23]

Para inserir parágrafos de programação dentro do HTML é necessário identificar o início e o fim do set de JavaScript, da seguinte forma:

```
<SCRIPT> Set de instruções </SCRIPT>
```

Este procedimento pode ser adotado em qualquer local da página. Entretanto, para melhor visualização e facilidade de manutenção, recomenda-se que toda a lógica seja escrita no início do documento, através da criação de funções a serem invocadas quando se fizer necessário (normalmente atreladas a eventos).

Se a lógica é escrita a partir de um determinado evento, não é necessário o uso dos comandos <SCRIPT> e </SCRIPT>.

Os comandos JavaScript são sensíveis ao tipo de letra (maiúsculas e minúsculas) em sua sintaxe. Portanto, é necessário que seja obedecida a forma de escrever os comandos, de acordo com a forma apresentada ao longo deste manual. Caso seja cometido algum erro de sintaxe quando da escrita de um comando, o JavaScript interpretará, o que seria um comando, como sendo o nome de uma variável.

3.5.1. Utilização

Para inserir scripts em um documento HTML é necessário colocá-los entre as tags <Script> e </Script>, ficando mais ou menos assim:

```
<Script>
    comandos
</Script>
```

Pode-se também usar códigos em arquivos externos e usar o atributo SRC=, contudo é obrigatório o uso da extensão .js exemplo :

```
<SCRIPT SRC="meucodigo.js"></SCRIPT>
```

As tags podem ser colocadas em qualquer lugar de seu documento HTML, ou, de acordo com a necessidade. Se houver a necessita de que o código JavaScript seja interpretado antes que do código HTML, ele deve ser colocado dentro do cabeçalho das páginas HTML que são as tags <HEAD> e </HEAD>.

A tag <Script> pode também especificar qual é a versão do JavaScript a ser usada, através do atributo "LANGUAGE=". O que vai neste atributo pode ser apenas o comum "JavaScript", "JavaScript1.1", "JavaScript1.2". A maioria dos browsers irão ignorar qualquer Script que ele não reconheça ou não suporte. Isto permite à você separar os scripts para diferentes versões de browsers. esta lista abaixo irá mostrar as versões do JavaScript suportadas por diferentes versões do Netscape.[9,12]

3.5.2. ECAMScript

Em 1996 os desenvolvedores web começaram a reclamar que a Netscape estava indo em uma direção com o JavaScript e a Microsoft se encaminhava em uma direção de certa forma compatível, mas diferente com o JScript. Ninguém gosta de ter que codificar páginas que manipulem dialetos diferentes do JavaScript, ou que tenham seu código funcionando em um navegador, mas não em outro. [9,12]

Assim a Netscape foi até um corpo internacional de padrões chamado ECMA e submeteu a eles a especificação da linguagem JavaScript, enquanto a Microsoft apresentava seus próprios comentários e sugestões. A ECMA fez o que costumam fazer os corpos de padrões e, em junho de 1997, produziu um padrão chamado ECMA-262 (também conhecido como ECMAScript). [9,12]

3.5.3. XMLHttpRequest

Com o objeto XMLHttpRequest e um programa no servidor escrito em qualquer linguagem avessível via o protocolo Http, é possível criar uma página realmente interativa, que se atualize de maneira totalmente transparente, sem a necessidade do reload na página. O XMLHttpRequest permite executar validações enquanto o usuário termina de preencher um determinado formulário, sem que isso interrompa o seu trabalho. Isso ocorre porque a comunicação entre o cliente e o servidor se dá de forma assíncrona.

Existem diferenças entre as implementações nos navegadores atuais, no IE o XmlHttpRequest é um ActiveX, já Mozilla e no Safari ele é um objeto nativo, portanto a chamada precisa verificar o tipo do navegador. [2,6]

Chamada no IE:

```
var req = new ActiveXObject("Microsoft.XMLHTTP");
```

Chamada no Mozilla/Safari:

```
var req = new XMLHttpRequest();
```

4. Estudo de Caso: “O uso de Geoprocessamento para o Controle de Solicitação de Ambulância do HUB”

4.1. Análise do Problema

4.1.1. Oportunidade

Nos dias atuais, a Hospital Universitário de Brasília – HUB, não possui um mecanismo eficiente de monitoração das solicitações de ambulância executadas pelos seus técnicos e clientes externos, o que impossibilita o controle dos prazos de atendimento. Por isso faz-se necessário o desenvolvimento de uma ferramenta que permita acompanhar estas solicitações, controlando assim o cumprimento dos prazos de atendimento.

4.1.2. Descrição do Problema

A seguinte declaração sintetiza o problema.

O problema da	Longa demora no atendimento do serviço solicitação de ambulância
afeta	O Hospital Universitário de Brasília
O impacto disso é	Não permitir que a HUB atenda suas atribuições legais nos prazos determinados, podendo gerar insatisfação dos técnicos e dos usuários externos.
Uma solução bem sucedida deve	Prover o controle sobre tempo de atendimento do serviço de solicitação de ambulância.

4.2. Decomposição do Problema

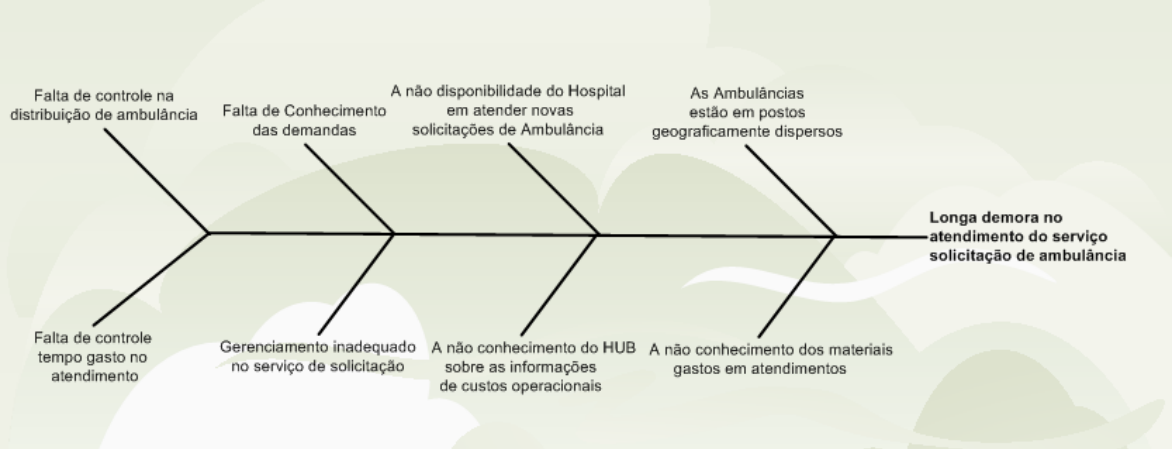

Figura 1 - Decomposição do problema

4.3. Descrições dos Usuários e Principais Interessados

4.3.1. Público-Alvo do Produto

- Hospital Universitário de Brasília - HUB

4.3.2. Principais Interessados

Interessado	Tipo de Interesse	Função/Participação
Diretor	Gerenciar melhor os processos de solicitações de ambulância.	Validar ações e soluções, tirar dúvidas da equipe.
HUB – Gerência de Informática.	Padronização dos sistemas desenvolvidos para a HUB.	Subsidiar com informações dos padrões de desenvolvimento. - Definição de arquitetura - Definição de ambiente - Definição de integralidade (em conjunto com a equipe).
Equipe de Sistemas - UNB	Promover uma solução para o problema.	Coletar informações, harmonizar os interesses dos envolvidos e apresentar uma proposta de solução e capacitação se necessário.

4.3.3. Usuários do Sistema

O quadro a seguir identifica os usuários que atuarão diretamente com o sistema, indicando o tipo de interação.

Usuário (Ator)	**Tipo de Interação**
Diretor	- Consulta
Gerente Informática	- Consulta - Validação de entrada de dados - Inserção de dados - Manutenção de dados cadastrados no sistema
Técnicos	- Consulta - Inserção de solicitações de ambulância

4.3.4. Necessidades que o Sistema irá atender

As necessidades estão listadas no quadro a seguir, com indicação do respectivo grau de importância. Os graus de importância estão definidos como alta, média e baixa.

Necessidade	***Importância***
Administrar o serviço de solicitação de ambulância com suas respectivas característica mínima de operação : ▪ Tipo de ambulância ▪ Tipo médico necessário ▪ Equipamentos mínimos ▪ Situação do atendimento	**Alta**
Controlar os tipos de ambulância a ser deslocada para atendimento	**Alta**
Gerenciar a fila de espera do serviço solicitação de ambulância geoReferenciado	**Alta**
Visualizar georeferenciado o serviço de solicitação de ambulância	**Baixa**
Desenvolver indicadores de desempenho.	**Baixa**
Manter dados de cep georeferenciado com auxilio do mapa	**Media**
Controlar gastos de operacionais.	**Média**
Conhecer as demandas.	**Média**

4.3.5. Gestor do Sistema

O gestor do sistema será o Gerente de Informática do HUB (Dr. Hervaldo Carvalho). Dentre as responsabilidades dos gestores estão:

1) Zelar pelo bom funcionamento do sistema.
2) Zelar pela qualidade das informações registradas no sistema.
3) Receber, avaliar e organizar solicitações de alteração no sistema.
4) Propor evolução das funcionalidades.
5) Encaminhar ajustes de funcionamento.
6) Deliberar sobre necessidades de parada do sistema para manutenção.
7) Representar os interesses dos usuários do sistema junto à organização.

4.4. Visão Geral do Produto

4.4.1. Declaração de Posicionamento

As declarações a seguir descrevem o posicionamento do produto a ser gerado pelo projeto sob o ponto de vista de seus usuários e em relação à solução atual.

Para	A gerência de informática e coordenação do HUB
Que	Fiscaliza e controla a operação de serviços de transporte de ambulância em nível de HUB nas rodovias do Distrito Federal.
o	Controle de Ambulância – SCA
é um	Sistema
que	uma estrutura de dados integradora com todas as informações necessárias para atendimento do serviço de solicitação de ambulância.
Diferente da	Situação atual na qual não existe uma ferramenta e tão pouco conhecimento de prazos e demandas
nosso produto	Subsidiará o gerenciamento do serviço solicitação de ambulância dentro de seus prazos mínimo de atendimento

4.4.2. Fronteiras do Sistema

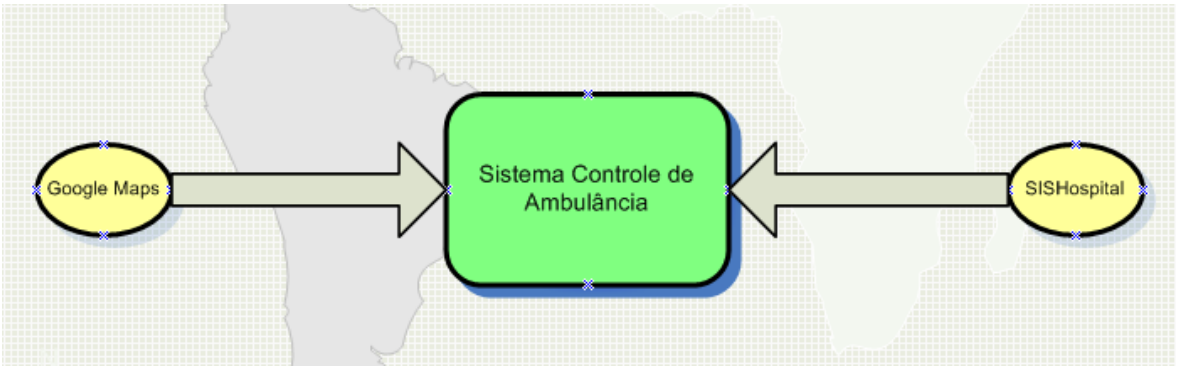

Figura 2 - Fronteiras do Sistema

4.4.3. Integração Com Outros Sistemas

- SIGHospital
- Banco de Dados Corporativo.

4.4.4. Escopo do Sistema

O sistema Controle de Ambulância é um sistema que controla as solicitações de ambulância no âmbito do Distrito Federal no escopo do Hospital Universitário de Brasília - HUB.

Será realizado um estudo junto a Gerencia para verificar a necessidade de integração com o SIGHospital – Sistema de Gestão de Hospitalar e com o Banco de Dados Corporativo.

4.4.5. Restrições do Sistema

- O sistema deve estar em conformidade com as diretrizes da HUB.
- Prazo de entrega 15/11/2006.
- Integrar com o SIGHospital.
- O modelo de dados do sistema Ambulância deverá ser compatível com o modelo do sistema SIGHospital. .

4.5. Requisitos do Produto

4.5.1. Recursos do Sistema

Código	*Prioridade*	*Título*
REC-001	Alta	Cadastrar as solicitações de ambulância
REC-002	Alta	Cadastrar os trajetos das ambulâncias
REC-003	Media	Cadastrar as categorias, equipamento e materiais na ambulância.
REC-004	Baixa	Cadastro as tabelas auxiliares
REC-005	Media	Cadastro das tabelas secundárias

4.6. Arquitetura Proposta

A proposta de arquitetura será na prática, o uso de mais de um estilo arquitetural. Essa heterogeneidade pode se dar de maneira hierarquia. Considere um sistema organizado em camada a qual pode ser composta de objetos. Logo verifica que a arquitetura é na realidade uma combinação de estilos.

A Web tem sido responsável por redesenhar a forma na qual os negócios têm sido realizados. Isso se deve à disponibilidade de uma infra-estrutura de distribuição de informações. É importante observar que aplicações com base na Web são aquelas que geralmente residem em um servidor num local central, tornando possível seu acesso em toda a Web a partir de qualquer computador. Para tanto, torna-se necessário a existência de um navegador e uma conexão com a Internet.

Uma arquitetura de múltiplas camadas com base na Web oferece as seguintes vantagens:

- Baixo custo de manutenção;
- Acesso universal através de navegadores para clientes;
- Acesso de qualquer local;

- Influência sobre a infra-estrutura de tecnologia existente;
- Aplicações/soluções com base em padrões.

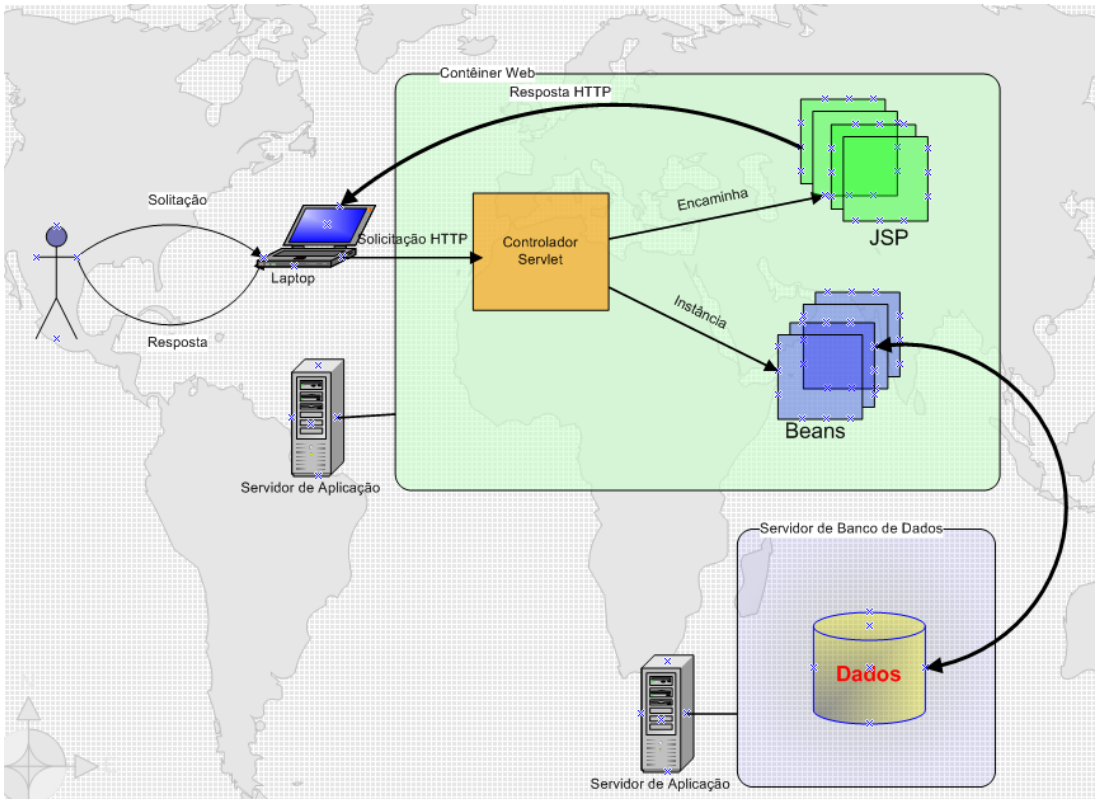

Figura 3 - Arquitetura de múltipla camada Web

A arquitetura em camadas tem o objetivo de permitir que aplicações sejam desenvolvidas de maneira produtiva e com facilidade de manutenção. Os objetivos principais são:

- **Modularidade** – Dividir a aplicação em módulos tão independentes quanto possível.
- **Manutenibilidade** – Reduzir o custo de manutenção da aplicação.
- **Extensibilidade** – Permitir que novas funcionalidades sejam adicionadas sem grande impacto nas já existentes.
- **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 têm os seguintes características:

- **Camada de Apresentação** – 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.

- **Camada de Negócio** – 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.

- **Camada de Persistência** – 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.

4.6.1. Representação da Arquitetura

A arquitetura do sistema Controle de Ambulância - SCA será representada através das seguintes visões arquiteturais:

- Visão de Casos de Usos
- Visão Lógica
- Visão de Processos
- Visão de Implantação
- Visão de Implementação
- Visão de Dados.

Para o desenvolvimento do Sistema de Controle de Ambulância - SCA considerando a adoção do desenvolvimento em três camadas. As camadas são a de apresentação, negócio e persistência, que se interligam para atender, de forma vertical, a uma solicitação de um usuário do sistema e estão independentes, na visão horizontal, para que possam ser incrementadas ou substituídas sem interferência entre elas. A figura abaixo ilustra a arquitetura.

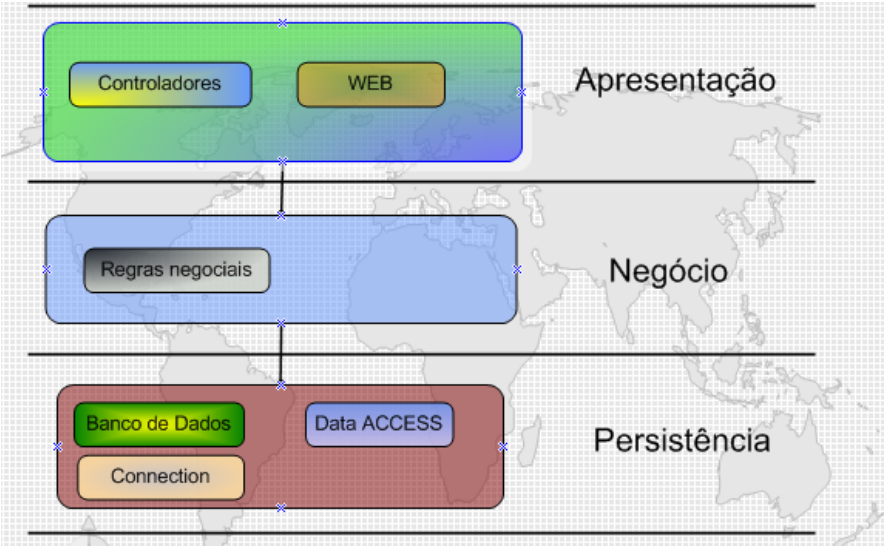

Figura 4 - Visão geral da arquitetura em camada

A camada de apresentação, representada pela cor verde, contém todos os artefatos para a interação com o usuário, sendo especializada para os ambientes em que este se encontra. A especialização WEB realiza a interação dentro do ambiente da Internet.

A camada de negócio, representada pela cor azul, contém todos os artefatos de negócio, todas as regras de negócio para a realização das tarefas disponíveis na camada de apresentação. Esta camada serve para execução e ligação entre a camada de persistência, que será descrita a seguir, com a camada de apresentação.

A camada de persistência tem como função transformar os objetos de negócio em dados relacionais, bem como decidir por qual via de acesso ao banco de dados deverá utilizar.

4.6.2. Visão de Casos de Uso

A Visão de Casos de Uso é uma entrada importante para a seleção do conjunto de cenários ou casos de uso que são o foco de uma iteração. Ela descreve o conjunto de cenários ou os casos de uso que representam alguma funcionalidade central e significativa. Também descreve o conjunto de cenários ou casos de uso que possuem cobertura arquitetural substancial (que experimenta vários elementos de arquitetura) ou que enfatizam ou ilustram um determinado ponto complexo da arquitetura.

4.6.3. Visão Lógica

Esta seção representa a descrição da Visão Lógica da arquitetura do sistema. Descreve as classes mais importantes, sua organização em pacotes de serviços e subsistemas e a organização destes subsistemas em camadas. Apresenta os elementos de software mais importantes. Diagramas de classes podem ser incluídos para ilustrar os relacionamentos entre classes arquiteturalmente significantes, subsistemas, pacotes e camadas.

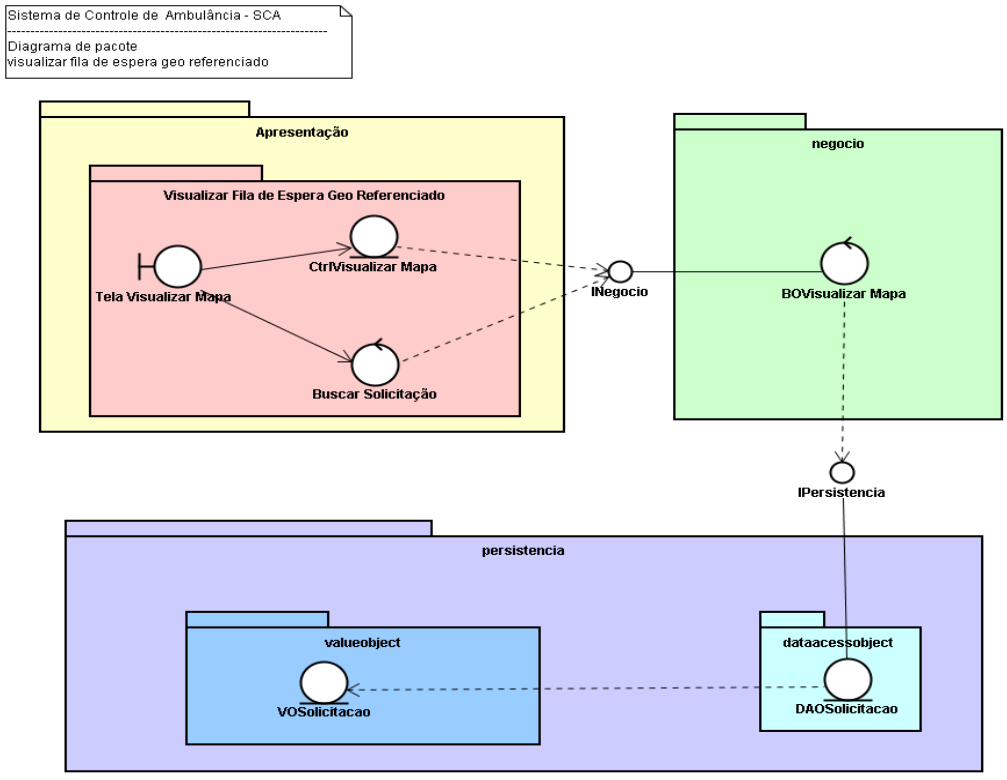

Figura 5 - Visão Lógica da Arquitetura do sistema

4.6.4. Visão Geral

O sistema SCA será dividido em três camadas. O desenvolvimento do sistema deverá seguir a seguinte norma de empacotamento.

src.br.unb.especializacao.sca.web.apresetacao.[sub-sistema].[nome_da_colaboracao]

src.br.unb.especializacao.sca.web.negocio.[sub-sistema].[nome_da_colaboracao]

src.br.unb.especializacao.sca.web.perisistencia.[sub-sistema].[nome_da_colaboracao]

4.6.5. Distribuição e reutilização

Extensibilidade

O Sistema deverá ter sua base de dados relacional integrada à base de dados do Sistema de Gestão Hospitalar.

Modularidade

O Sistema deverá ser projetado e implementado através de uma estrutura de módulos independentes e particionados logicamente.

Desenvolvimento baseado em componentes

O Sistema deverá ser desenvolvido utilizando metodologias e técnicas de componentes de software.

4.6.6. Segurança

Privacidade

O Sistema deverá possuir mecanismos de proteção e de controle de operações realizadas por um determinado perfil.

Autenticação de Usuários

O sistema deverá autenticar usuários autorizados ao seu manuseio.

Confidencialidade

O Sistema não poderá disponibilizar informações para indivíduos, entidades ou processos não autorizados.

4.6.7. Visão de Implantação

A Visão de implantação do projeto ainda não está definida na iteração atual.

4.6.8. Visão de Implementação

No diagrama abaixo está a definição da estrutura de pacotes e componentes distribuídos em camadas e modularizado em ambiente *Internet*

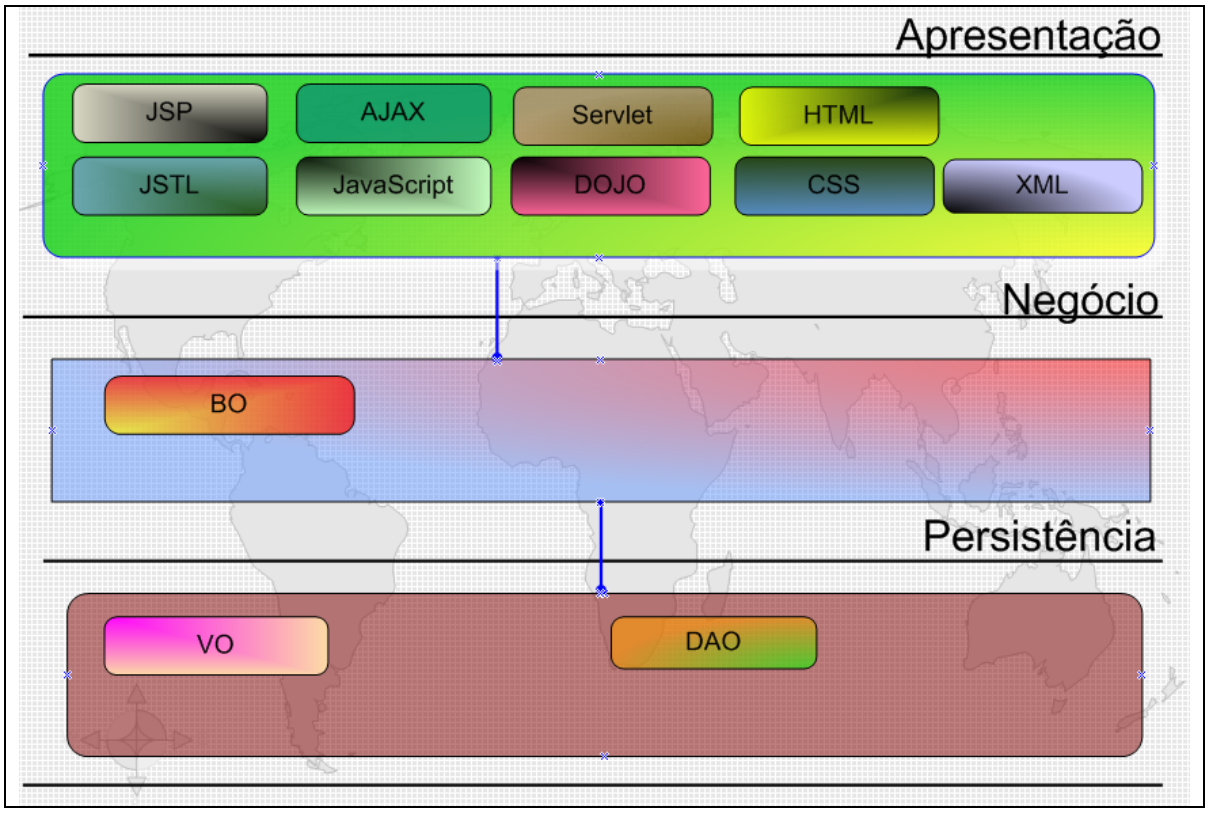

Figura 6 - Visão de implementação do sistema

4.6.9. Framework Dojo

Para garantir a qualidade do código do controlador da camada de apresentação que deve rodar em JavaScript no navegador do cliente foi utilizado o framework ***Dojo***, de código aberto, que é utilizado para criação de aplicações Web baseadas em AJAX.

O Dojo é um projeto comunitário desenvolvido para unificar os recursos do JavaScript com o DHTML (Dynamic Hypertext Markup Language) em uma única biblioteca JavaScript. A comunidade imaginou que não seria possível começar se todos não trabalhassem em conjunto, assim três "kits" independentes foram unificados dando origem ao *Dojo Foundation*, o qual mantém os direitos e fazem a manutenção do projeto.

4.6.10. Google Maps

Para fornecer os recursos de geoprocessamento necessários ao processo foi adotado a API de desenvolvimento do serviço Google Maps, que fornece a visualização de mapas e imagens de satélite da Terra de forma gratuita na Web.

Atualmente, o serviço disponibiliza mapas e rotas em apenas algumas cidades grandes dos Estados Unidos e outras na Europa. Disponibiliza também imagens via satélite do mundo todo, com possibilidade de um zoom com alta resolução das grandes cidades.

4.6.11. Mecanismos Arquiteturais

Um Padrão de Projeto pode ser definido como “a solução para um problema em um contexto” . Isto significa que os padrões de projeto nos oferecem soluções prontas para utilizarmos em determinados problemas que podemos enfrentar quando desenvolvemos um software orientado a objetos.

Padrão *Singleton*

Usado quando é necessário a existência de apenas uma instância de uma classe.

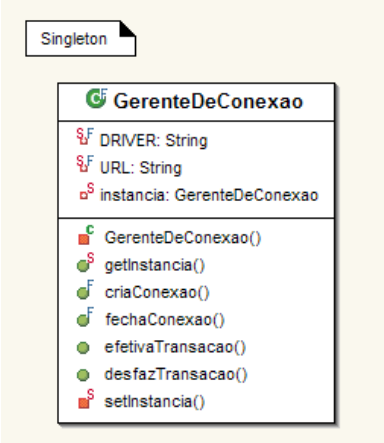

Figura 7 - Classe de Conexão com o Banco de dados

VO – Value Object

Usado para encapsular os valores de uma classe de forma a minimizar a sobrecarga na transferência de valores entre as várias camadas.

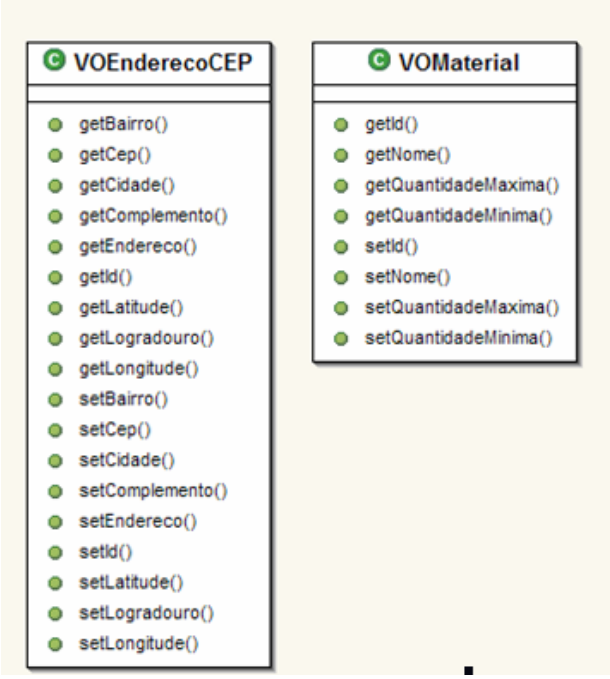

Figura 8 - Duas das Classes VO implementadas

DTO – Date Transfer Object

Usado como depósito de informações entre a interface do usuário e as camadas mais internas do sistema. Ela permite melhorar a comunicação entre as camadas evitando múltiplas chamadas.

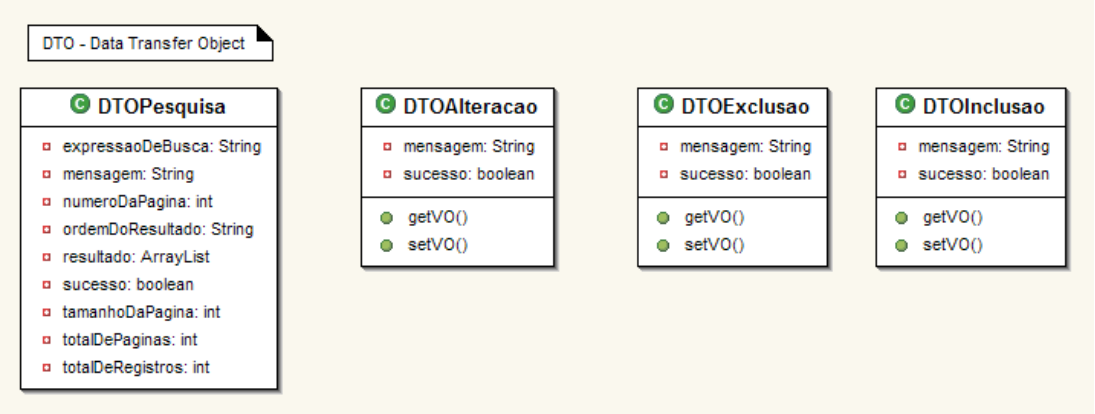

Figura 9 - Classes DTO implementadas

DAO – Data Access Object

Esse padrão abstrai a recuperação dos dados de forma a separar a relação do cliente de um recurso dos dados de seu mecanismo de acesso dos dados.

O DAO cria uma camada na aplicação separando as rotinas de acesso a banco de dados, geração de SQL, das rotinas de negócio de tal modo que os objetos de negócio não saibam como estão sendo gravados.

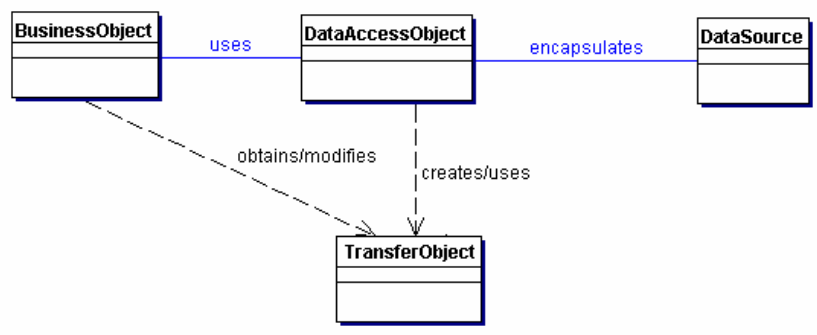

Figura 10 - Padrão DAO

- **BusinessObject** – São os objetos que precisam acessar informações de alguma fonte de dados para realizar suas operações

- **DataAccessObject** – É o núcleo do padrão. É encarregado de abstrair e encapsular o acesso a alguma base de dados para disponibilizar um acesso transparente à essa fonte de dados
- **DataSource** – Representa a implementação de uma fonte de dados, que pode ser um banco relacional, um banco OO, XML, texto, etc.
- **TransferObject** – Os objetos DAO podem utilizar esses tipos de objetos para retornar dados aos BusinessObject ou utilizá-los para atualizar um DataSource.

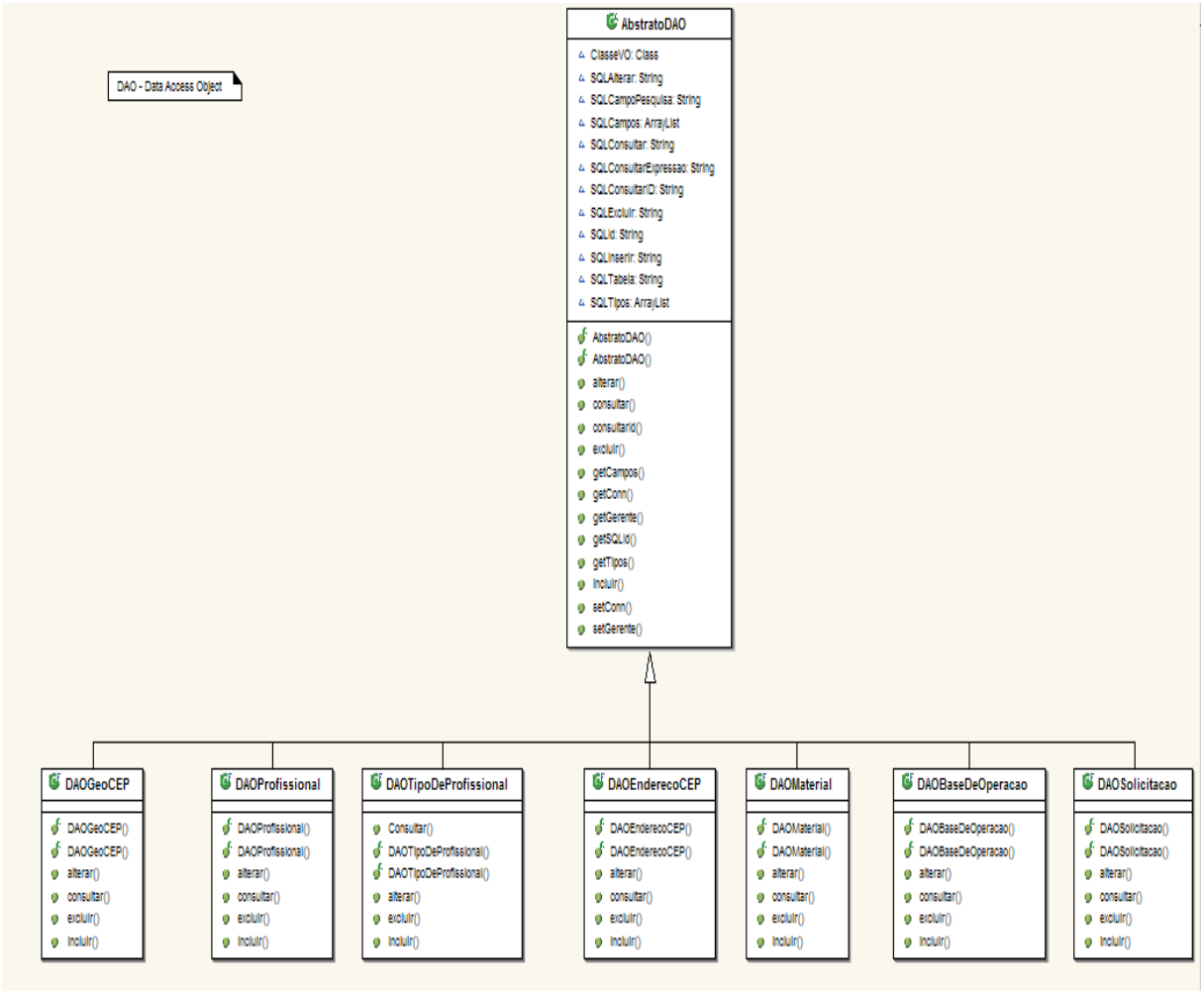

Figura 11 - Classes DAO implementadas

Factory Method

Define uma interface para a criação de objetos, mas deixa as subclasses decidirem qual classe irão instanciar. O FactoryMethod permite que uma classe transfira para as subclasses a responsabilidade pela criação de novas instâncias.

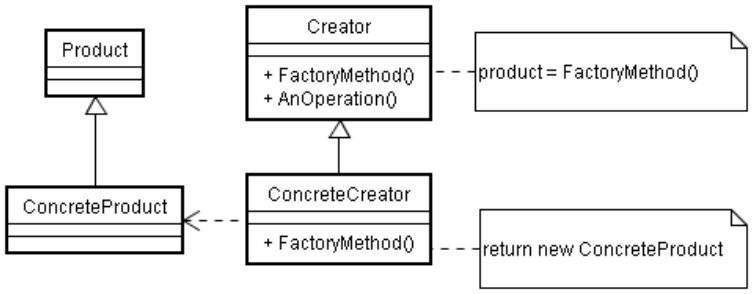

Figura 12 - padrão factoryMethod

- **Product** -Define a interface dos objetos que o FactoryMethod criará.
- **ConcreteProduct** -Implementações da interface de Product
- **Creator** - Declara o FactoryMethod, que retorna um objeto do tipo Product
- **ConcreteCreator** - Sobrepõe o FactoryMethod a fim de retornar uma instância de um produto concreto.

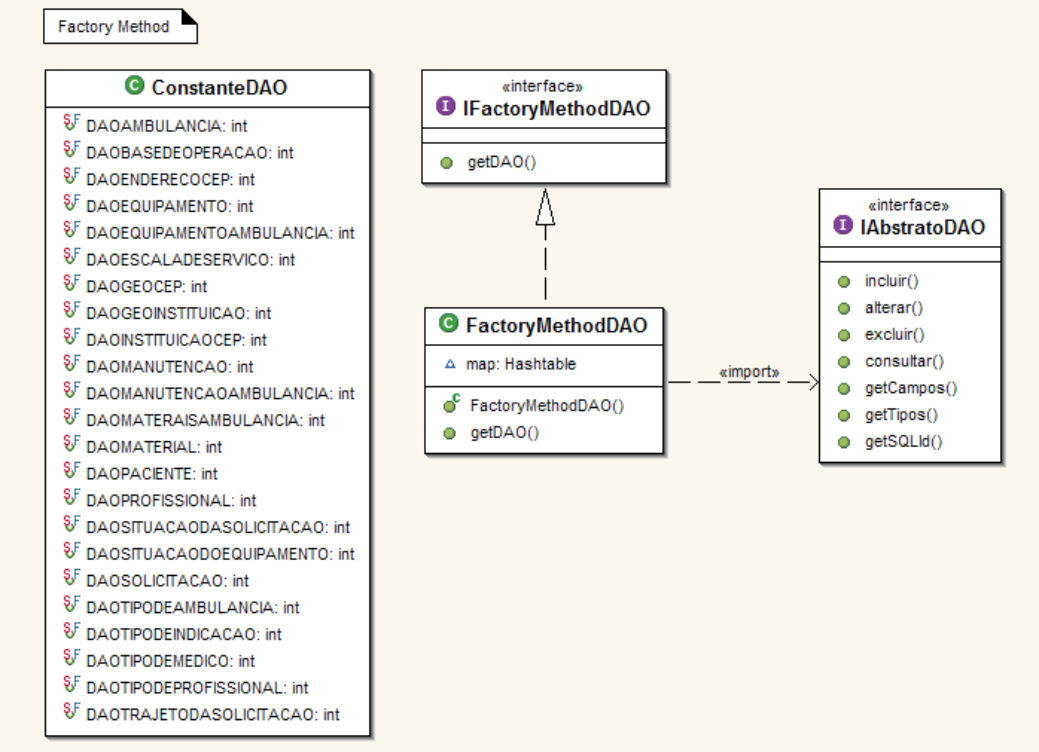

Figura 13 - Classes factoryMethod implementadas

Front Controller

O Front Controller é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação (Model), da interface do usuário (View) e do fluxo da aplicação (Controller). Permite que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces.

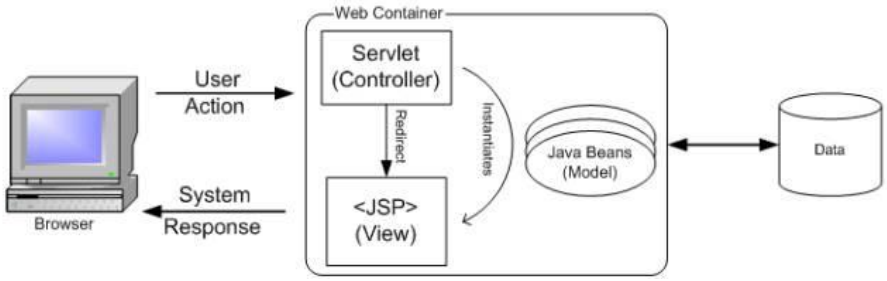

Figura 14 - Padrão Front Controller

O **Model** é tudo que envolve o Business Logic, sem considerações de interface

O **View** representa numa interface o estado corrente do Model

O **Controller** é usado para alimentar o Model quando ocorrem entradas na interface par o usuário.

Nossa aplicação tem dois controladores, um é uma Servlet configurada de forma a mapear as ações solicitadas pelo cliente. O outro controlador é executado do lado do cliente, sendo escrito em JavaScript tendo como principal responsabilidade o controle das interações do usuário de forma a capturar as solicitação, efetuar as chamadas a Servlet do lado do servidor, retornando os resultados de forma assíncrona.

Visão de dados

Esta visão tem objetivo de mostra um modelo relacional permite ao projetista criar um modelo lógico consistente da informação a ser armazenada no banco de dados.

Modelo de Entidade Relacionamento – Lógico

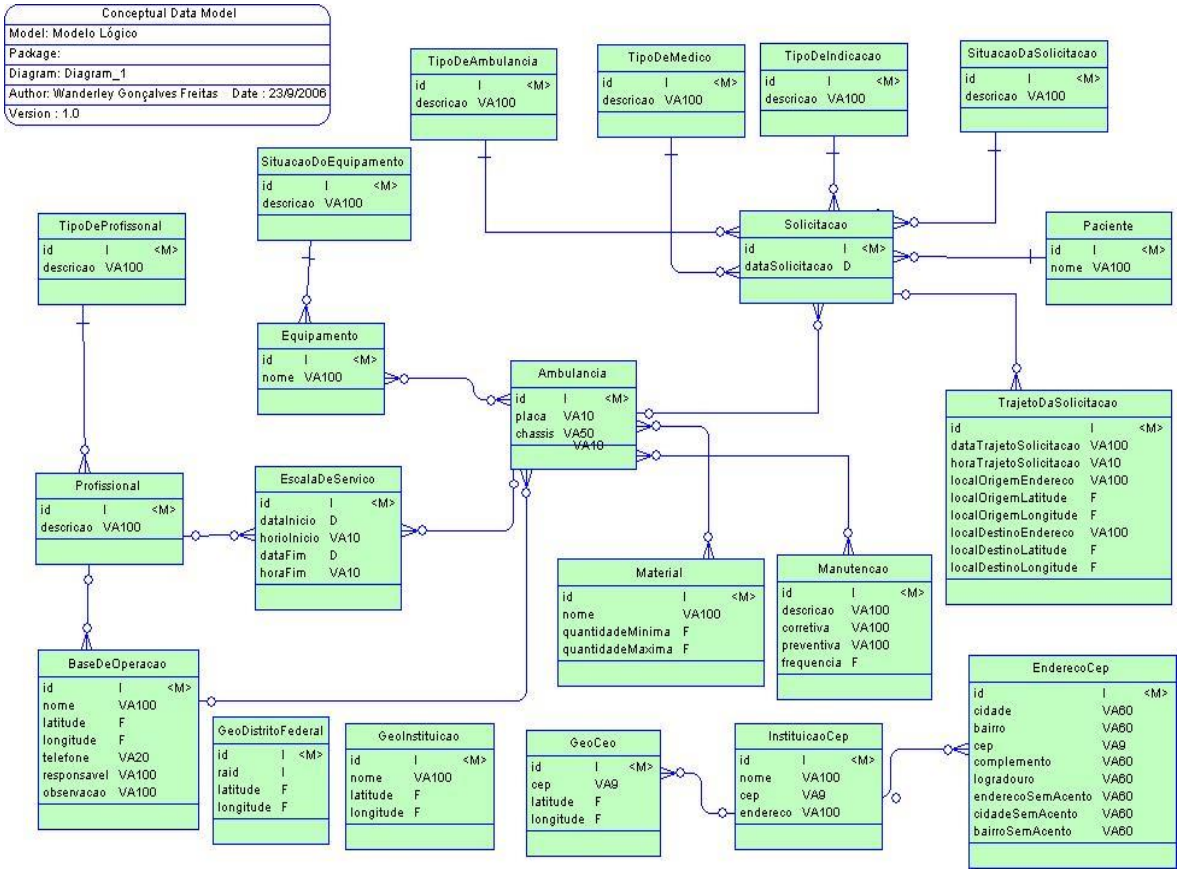

Figura 15 - Modelo Lógico

Modelo de Entidade Relacionamento – Físico

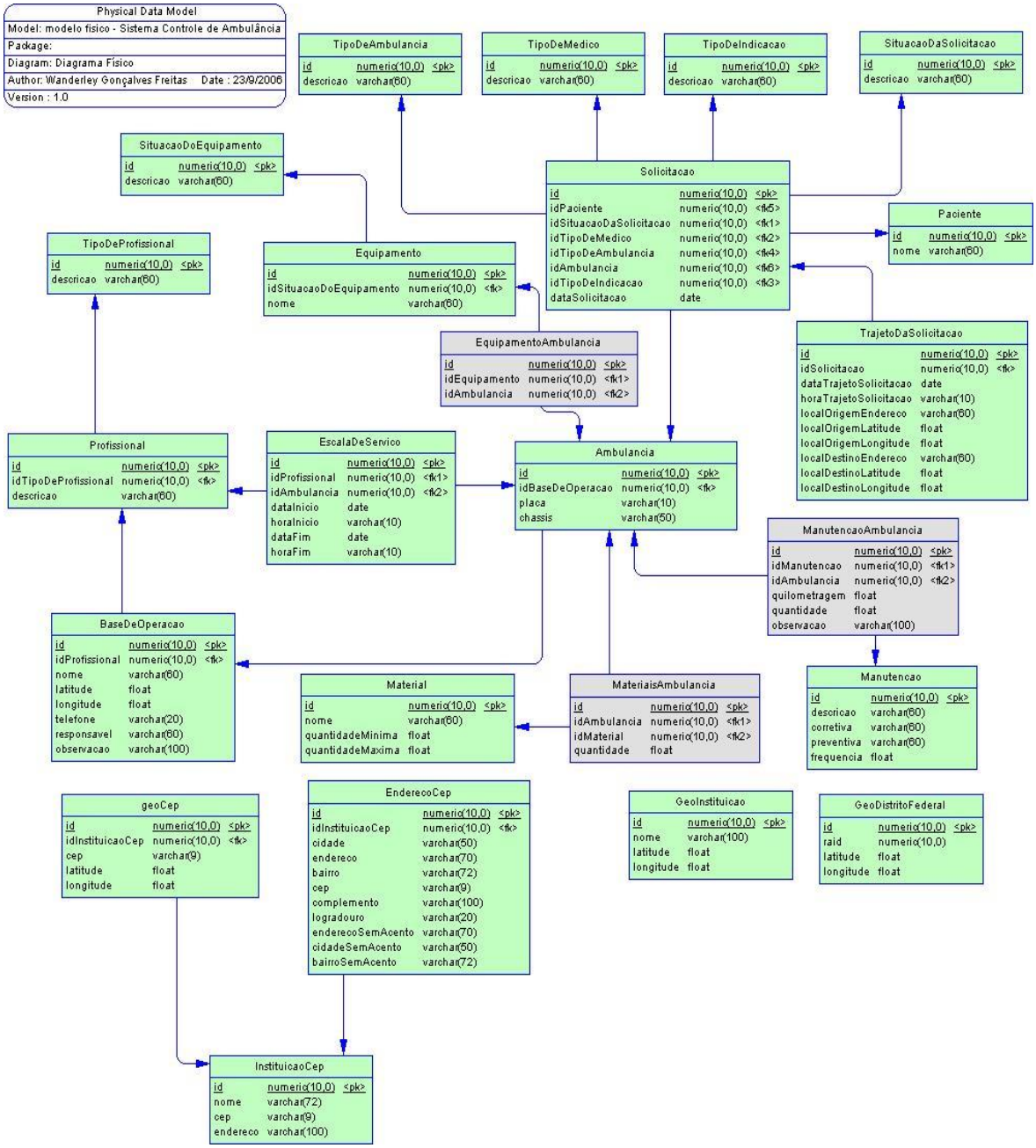

Figura 16 - Modelo Físico

4.7. Modelo de Domínio

4.7.1. Classes

Classe	Pacote
VOAmbulancia	Dominio
VOBaseDeOperacao	Dominio
VOEquipamento	Dominio
VOEquipamentoAmbulancia	Dominio
VOEscalaDeServico	Dominio
VOManutencao	Dominio
VOManutencaoAmbulancia	Dominio
VOMaterial	Dominio
VOMaterialAmbulancia	Dominio
VOPaciente	Dominio
VOProfissional	Dominio
VOSituacaoDaSolicitacao	Dominio
VOSituacaoDoEquipamento	Dominio
VOSolicitacao	Dominio
VOTipoDeAmbulancia	Dominio
VOTipoDeIndicacao	Dominio
VOTipoDeMedico	Dominio
VOTipoDeProfissional	Dominio
VOTipoGenerico	Dominio
VOTrajetoDaSolicitacao	Dominio
VOEnderecoCep	Domínio
VOGeoInstituicao	Domínio
VOInstituicaoCep	Domínio
VOGeoCep	Domínio
VOGeoDistritoFederal	Domínio

4.7.2. Diagrama do Modelo de Domínio

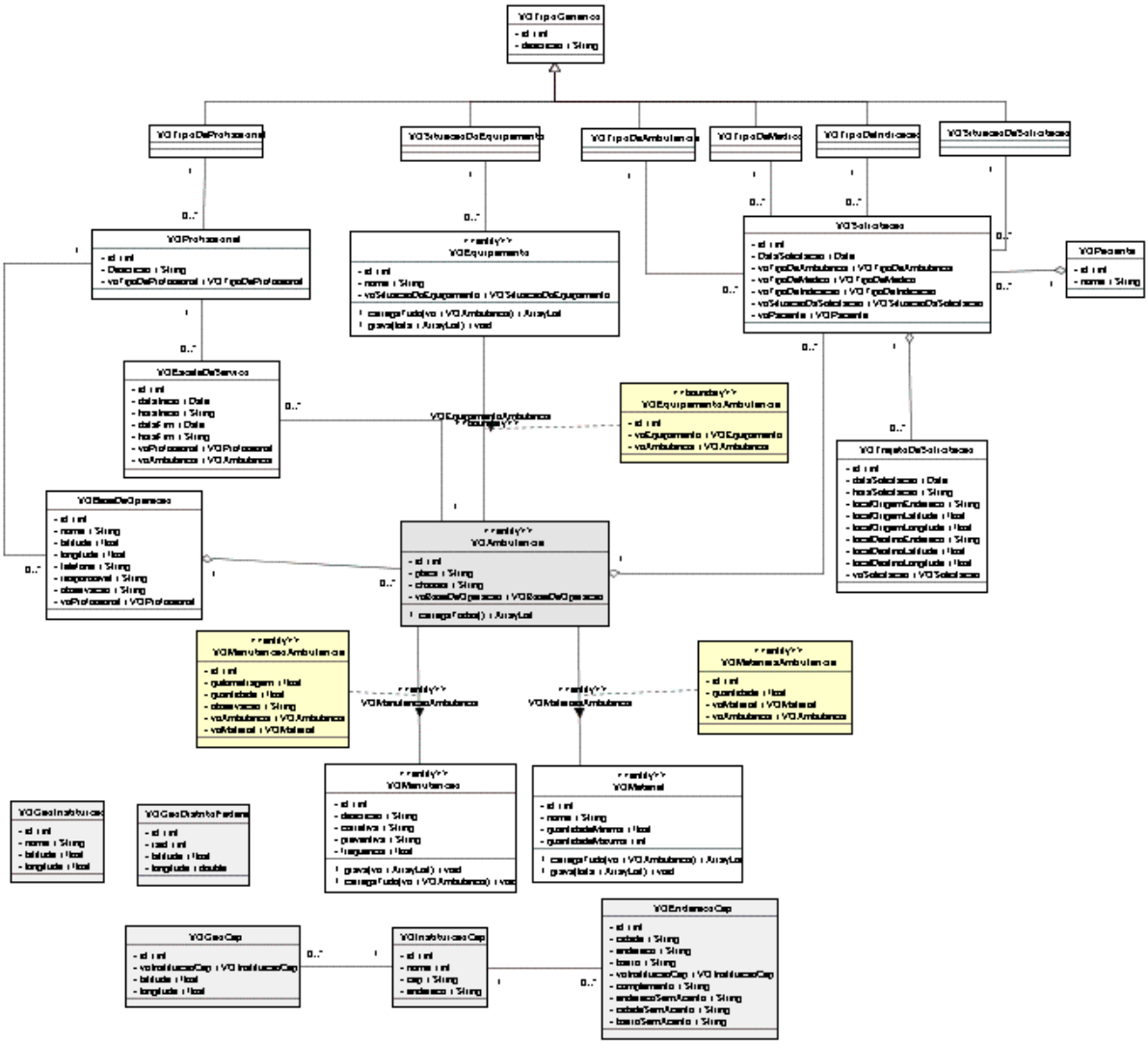

Figura 17 - Modelo de Domínio

4.8. Fluxo de Interface

A proposta deste documento é ilustrar o fluxo de interface do usuário das principais funcionalidades.

4.8.1. Diagrama

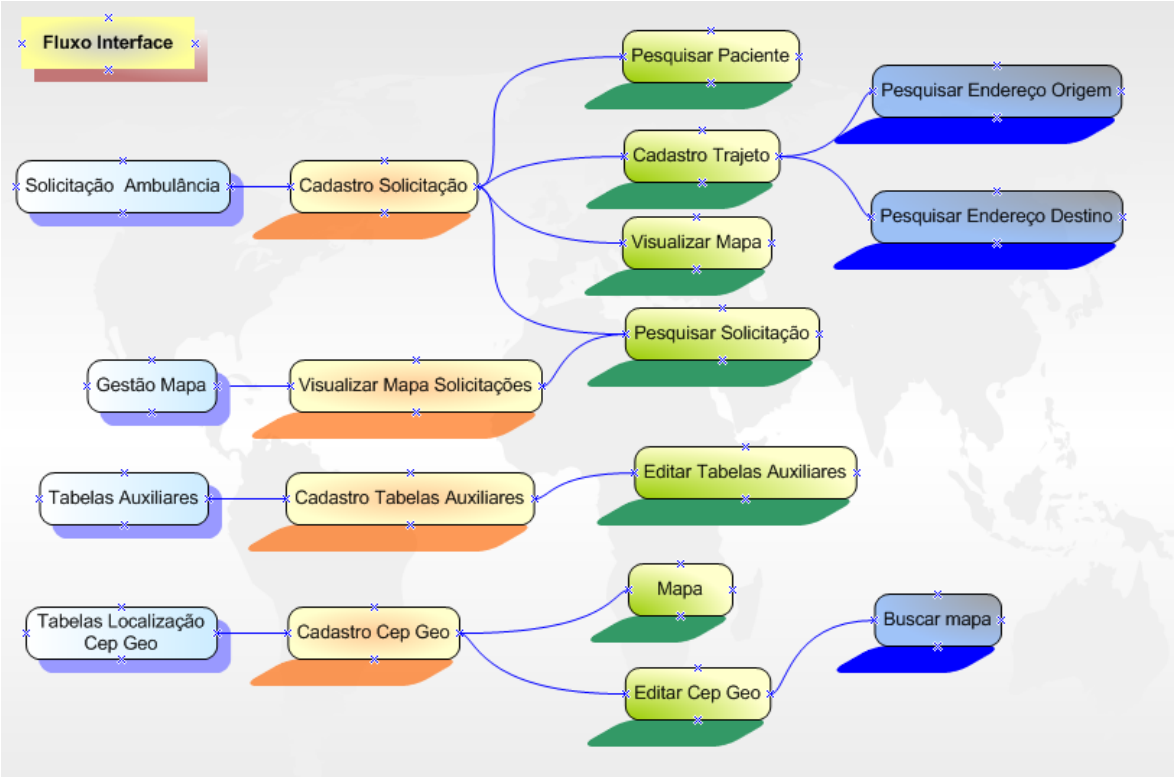

Figura 18 - Diagrama do Fluxo de interface

4.9. Atores

São papéis desempenhados por qualquer usuário de um caso de uso, ou seja, o ator é quem solicita os serviços disponíveis em caso de uso.

Um ator pode ser :

- Uma pessoa que interage com o sistema
- Uma hardware que interage com o sistema
- Um outro sistema que tenha a necessidade de utilizar o caso de uso.

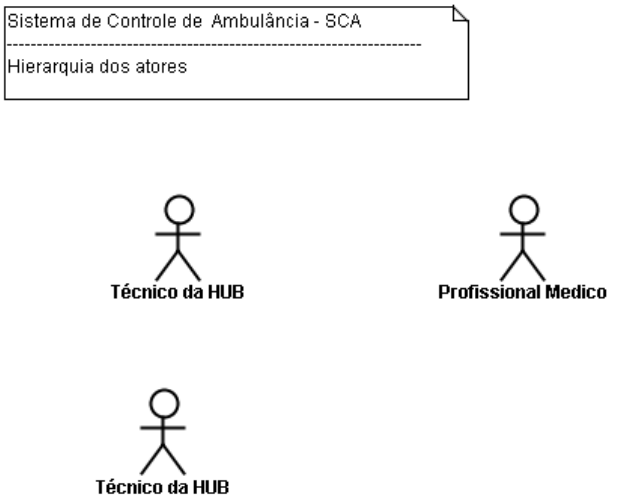

Figura 19 - Atores do sistema

Atores do Sistema	Descrição
Administrador do sistema	• Responsável pelas tarefas administrativas do sistema. • Responsável pela entrada de informações básica no sistema (tabelas secundárias e básica)
Tecnico do HUB	• Responsável pelo gerenciamento do serviço solicitação de ambulância junto ao HUB
Profissional Médico	• Responsável pela composição dos equipamentos, medicamento e matérias de enfermagem.

4.10. Casos de Uso

Um caso de uso é um documento que descreve os cenários pretendidos para um sistema, com o objetivo de atender às necessidades do usuário.

4.10.1. Diagrama de Casos de Uso

O diagrama de Caso de Uso representa um conjunto de cenários identificados, que seja útil aos usuários de um sistema. Um caso de uso descreve cada cenário possível para um sistema, composto por seqüência de passos em que há interação entre os usuários e o sistema.

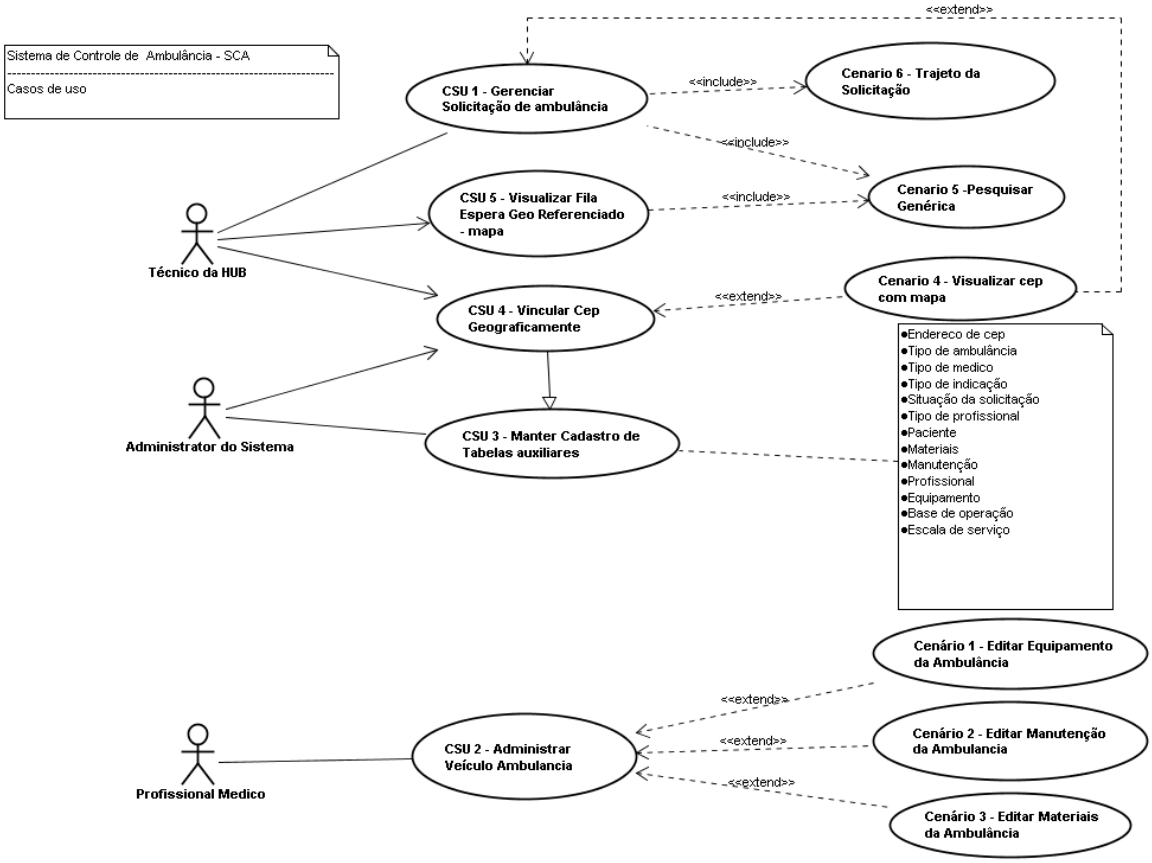

Figura 20 - Diagrama de Caso de uso

4.10.2. Caso de Uso Gerenciar Solicitação de ambulância

Identificador:	CSU-001.
Nome:	Gerenciar Solicitação de ambulância
Objetivo:	Este caso de uso ocorre quando o usuário (técnico) faz uma solicitação de ambulância junto ao departamento de transporte.
Ator:	Técnico do Setor de Transporte.
Fluxo de Eventos:	
Fluxo Básico:	
	O usuário solicita opção Ambulância - Solicitação. O sistema exibe um formulário (data solicitação, lista de paciente, lista de indicações[opções padrões selecionada], lista de médicos [opções padrões selecionada], lista de ambulâncias, lista de equipamento necessários [opções padrões selecionada])
Fluxo Alternativo:	
	Incluir O usuário seleciona o botão incluir. O sistema exibe um formulário(data solicitação, lista de paciente, lista de indicações[opções padrões selecionada], lista de médicos [opções padrões selecionada], lista de ambulâncias, lista de equipamento necessários [opções padrões selecionada]) de cadastro. O sistema habilita os botões Salvar e Cancelar os campos para edição. O usuário preenche os campos desejados. O usuário solicita Salvar. O sistema critica as informações, grava e habilita o botão INCLUIR TRAJETO.
	Alterar O usuário solicita Cenário 004 - Pesquisar Solicitação. O sistema executa o Cenário 004 - Pesquisar Solicitação. O sistema recebe a solicitação selecionada do Cenário 004 – Pesquisa Solicitação e carrega (data solicitação, paciente, indicações, médicos, ambulâncias, equipamento necessários) em formulário de cadastro com os campos para edição e habilita os botões Salvar e Cancelar. O usuário preenche os campos desejados. O usuário solicita salvar. O sistema critica as informações, grava e habilita o botão INCLUIR TRAJETO.

Fluxo Alternativo:	
	Excluir O usuário solicita Cenário 004 - Pesquisar Solicitação. O sistema executa o Cenário 004 - Pesquisar Solicitação. O sistema recebe a solicitação selecionada do Cenário 004 – Pesquisa Solicitação e carrega (data solicitação, paciente, indicações, médicos, ambulâncias, equipamento necessários) em formulário de cadastro com os campos para edição e habilita os botões CONFIRMAR e Cancelar. O usuário solicita CONFIRMA. O sistema critica as informações, excluir e habilita o botão INCLUIR TRAJETO.
	Incluir trajeto O usuário solicita INCLUIR TRAJETO - Cenário Trajeto de Solicitação. O sistema executa o Cenário 006-Trajeto de Solicitação. O sistema recebe o registro de trajeto efetuado pelo Cenário 006-Trajeto de Solicitação, efetua o calculo da distância entre ponto de origem e destino, atualiza a lista e habilita o botão INCLUIR TRAJETO.
	Alterar trajeto O usuário seleciona trajeto desejado na lista de trajeto. O usuário solicita o Cenário 006- Trajeto de Solicitação. O sistema executa o Cenário 006-Trajeto de Solicitação passando trajeto selecionado com parâmetro. O sistema recebe a novo trajeto o Cenário 006 - Trajeto de Solicitação e atualiza a lista de trajeto (data/hora, origem, destino e distancia) em formulário e habilita os botões INCLUIR TRAJETO, ALTERAR TRAJETO e EXCLUIR TRAJETO.
	Excluir trajeto O usuário seleciona trajeto desejado na lista de trajeto. O usuário solicita o Cenário 006 - Trajeto de Solicitação. O sistema executa o Cenário 006-Trajeto de Solicitação passando trajeto selecionado com parâmetro. O sistema atualiza a lista de trajeto (data/hora, origem, destino e distancia) em formulário e habilita os botões INCLUIR TRAJETO, ALTERAR TRAJETO e EXCLUIR TRAJETO.
Exceção:	
Ponto de extensão:	
Ponto de inclusão:	

Tela inicial

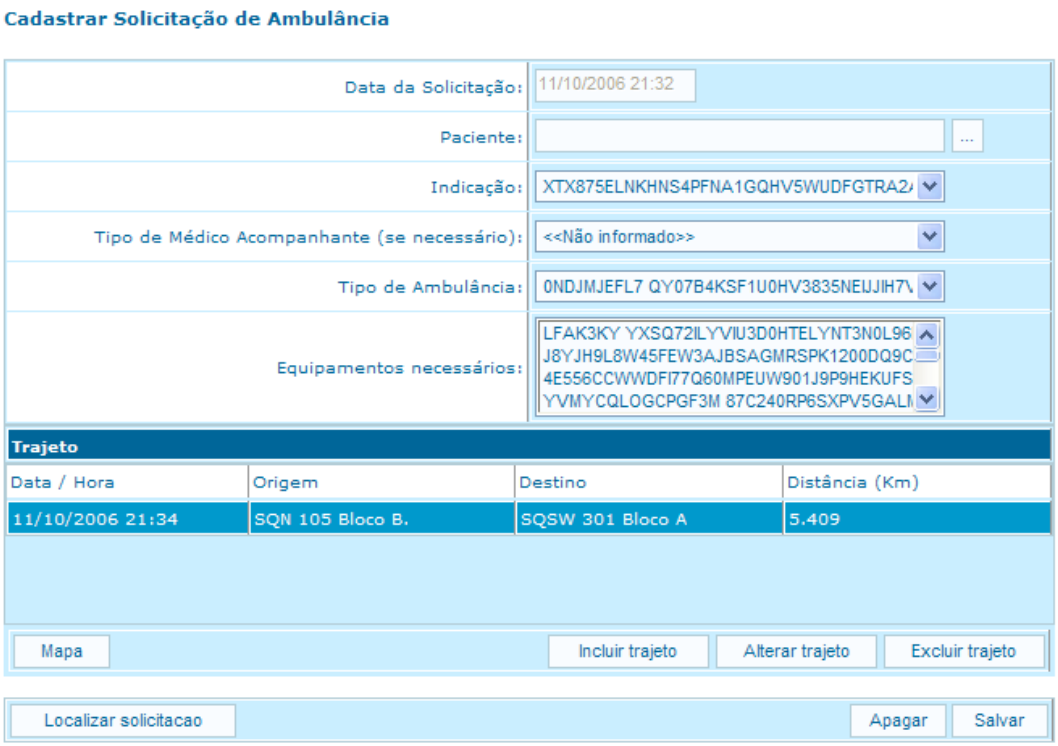

Figura 21 - Solicitação

Tela de pesquisa (Paciente e Endereço)

Figura 22 - Pesquisa

Tela para inclusão e alteração de trajeto

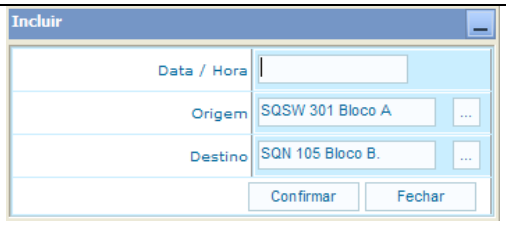

Figura 23 - Manutenção de trajeto

Tela de visualização do trajeto

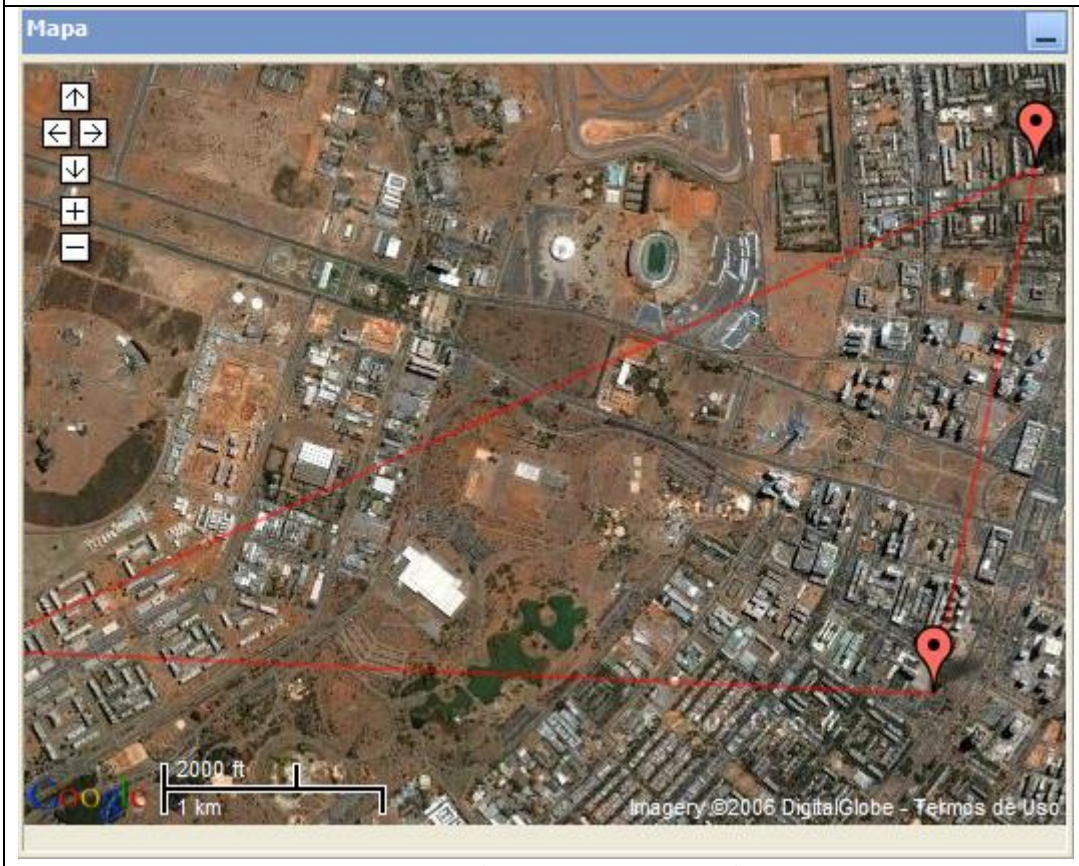

Figura 24 - Mapa Trajeto

.

4.10.3. Caso de Uso Administrar Veículo Ambulância

Identificador:	CSU-002
Nome:	Administrar Veículo Ambulância
Objetivo:	Este caso de uso ocorre quando o usuário(técnico) faz um solicitação de ambulância junto ao departamento de transporte.
Ator:	
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema carrega uma lista em um formulário com os principais atributos do veiculo ambulância (placa, chassi, base de operação) e habilitando as opções inclusão, alteração, exclusão, editar materiais, editar equipamentos, editar manutenção, atualização da lista de ambulância e pesquisa pela placa.
Fluxo Alternativo:	
	Inclusão da ambulância O usuário solicita incluir. O sistema exibe um formulário de cadastro de ambulância (placa, chassis e base de operação) habilitando os campos para edição. O usuário preenche os campos desejados e solicita a gravação. O sistema critica as informações, grava e retorna para formulário anterior atualizando a lista novamente.
	Alteração da Ambulância O usuário escolhe na lista o placa da ambulância que deseja fazer alteração. O sistema exibe um formulário de cadastro da ambulância (placa, chassis e base de operação) com os dados da ambulância selecionada e habilitando os campos para edição. O usuário altera os campos desejados e solicita a gravação. O sistema critica as informações, grava e retorna para formulário anterior atualizando a lista novamente.
	Exclusão do cliente O usuário escolhe na lista o nome do cliente que deseja fazer exclusão. O sistema exibe um formulário de cadastro da ambulância (placa, chassis e base de operação) . O sistema exibe uma mensagem de confirmação da exclusão. O usuário faz confirmação solicitando excluir. O sistema critica as informações, exclui no banco de dados e retorna para formulário anterior atualizando a lista novamente.

Fluxo Alternativo:	
	Atualização da lista de ambulância O usuário solicita atualização da lista de ambulância. O sistema carrega e exibe em lista todas as ambulância em formulário com os principais atributos do veiculo ambulância (placa, chassi, base de operação) ordenada pela placa do veiculo e habilitando as opções inclusão, alteração, exclusão, editar materiais, editar equipamentos, editar manutenção, atualização da lista de ambulância e pesquisa pela placa.
	Pesquisa pela placa do Veículo Ambulância O usuário digita a placa do veiculo no respectivo campo e solicita a busca. O sistema realiza a pesquisa e exibe uma lista em um formulário com os principais atributos do veiculo ambulância (placa, chassi, base de operação) que atendem aos critérios de busca (placa do veículo ambulância em qualquer parte do nome) O usuário poderá solicitar - Editar Materiais da Ambulância
	Opção materiais. O sistema executa o cenário - Editar Materiais da Ambulância passando como parâmetro a identificação da ambulância. O usuário poderá solicitar - Editar Equipamentos da Ambulância.
	Opção Equipamento. O sistema executa o cenário - Editar Equipamento da Ambulância passando como parâmetro a identificação da ambulância. O usuário poderá solicitar - Editar Manutencão da Ambulância
	Opção manutenção. O sistema executa o cenário - Editar Manutenção da Ambulância passando como parâmetro a identificação da ambulância.
Exceção:	
Ponto de extensão:	
Ponto de inclusão:	

Tela inicial

Cadastrar Ambulância

Lista de itens — Pesquisar: [ ] Ok

Id	Placa	Chassis	Base de Operação
0	RBE45F0BP6	YBWG1TVX9Y06K3U9S11C JJO	E10BPUD7
1	MBDNPU1 TX	OQJ4GUXPKIGDNEO4980ARW6OVCMQR	8KG
2	CU4VR809X3	V0TB 5A5WWKROPX	KG4N80I62WVPU46D2H31LAV47HAN463FWCT
3	EDTMNFNFXS	VGMXLD UPA9UO0ONBT410	8KG
4	4VNC7EXV1G	SREGHGGV74V5FOOJM01UJ0OSCXQ6PK	OLMMNB113EPRNIJS53L4EB055FLDXXKY9DE7
5	IS1JWP2T 5	EMBVMIW2V1UVOH19RNXU5A IY7	IHW3VRW0F GLN6TX7Y
6	A9ILAE5QW2	RFWIQ55F94VMUG5U6EAW	SCM6SVYS9H93GTUI0PG37LHD
7	XSGM1POLY9	A	BO0D6QIII1DOLAIGGL9R1LADH3CHNBKJW1Y8
8	GIRFNWL8P6	HOFCAY4KHLSTG3WRHQE37MJCWSWJH	E10BPUD7
10	BLA086WOI1	PEYDN3XPWR8GTBEOHYB803SJIT7CHHY	75U2XNNB7UJMRQJMYR J4GFENSRT1PVDQ
14	U1Q H8U58	AAXNIUHDM6QLF621FRSUS5DPLI2BQWH	UDW72AEL27NNDJQL9DF2 G7LC
15	VI8SOXCO3H	B0NACJCNGY3AOTCGTBM81AE86UMVMD	BO0D6QIII1DOLAIGGL9R1LADH3CHNBKJW1Y8
17	CDQ36XALRG	L0O R6C6PEWXO7747UQWP8MM	54GAKHB5O811M2WU5TSGS03VW
18	LX4GPS8PXY	3756QHdfdsY	O54JL6CS3A2VLKD40CO45S
19	YNQP8LL9YI	56LTEM4EY5JC	UDW72AEL27NNDJQL9DF2 G7LC

<< < Pág.: 1 de 2 [22 registro(s)] > >> Incluir Alterar Excluir

Figura 25 - Cadastro de ambulância

4.10.4. Caso de Uso Manter Tabelas Auxiliares

Identificador:	CSU-003
Nome:	Manter Tabelas Auxiliares
Objetivo:	Esse caso de uso especifica as funcionalidades de Incluir, Alterar, Excluir, Consultar e Pesquisar para as tabela: Base de operação, Endereço, Equipamento, Manutenção, Materiais, Paciente, Profissional, Situação da solicitação, Tipo de ambulância, Tipo de indicação, Tipo de Medico e Tipo de profissional.
Ator:	Administrador do sistema
Fluxo de Eventos:	
Fluxo Básico:	
	O usuário solicita a opção tabelas auxiliar - o tipo de tabela . O sistema carrega uma lista vazia em um formulário (id, endereço, bairro,cidade e cep). e habilita os botões INCLUIR e OK do PESQUISAR.
Fluxo Alternativo:	
	Incluir O usuário seleciona o botão incluir. O sistema exibe um formulário(id, endereço, bairro, cidade e CEP) de cadastro. O sistema habilita os botões CONFIRMAR e FECHAR e os campos para edição. O usuário preenche os campos desejados e solicita confirma. O sistema critica as informações, grava, fecha o formulário, atualiza a lista e habilita o botão INCLUIR.
	Alterar O usuário solicita a pesquisa. O sistema executa o cenário Pesquisar O usuário seleciona o registro desejado. O sistema habilita os botões INCLUIR, ALTERAR e EXCLUIR. O usuário solicita ALTERAR. O sistema carrega o registro selecionado (id, endereço, bairro,cidade e CEP) em um formulário com os campos para edição e habilita os botões CONFIRMAR E FECHAR. O usuário altera os campos desejados e solicita confirma. O sistema critica as informações, grava, fecha o formulário, atualiza a lista e habilita o botão INCLUIR.

Fluxo Alternativo:	
	Excluir O usuário solicita pesquisa. O sistema executa o cenário Pesquisar O usuário seleciona o registro desejado. O sistema habilita os botões INCLUIR, ALTERAR e EXCLUIR. O usuário solicita EXCLUIR. O sistema carrega o registro selecionado (id, endereço, bairro,cidade e CEP) em um formulário com os campos para edição e habilita os botões CONFIRMAR E FECHAR. O usuário solicita CONFIRMA. O sistema critica as informações, exclui o registro , fecha o formulário, atualiza a lista e habilita o botão INCLUIR.
	Pesquisar O usuário preenche o parâmetro de pesquisa. O sistema realiza simultaneamente a pesquisa que atende o critério de busca naquele momento de preenchimento e carrega em grade o resultado.
Exceção:	
Ponto de extensão:	
Ponto de inclusão:	

Tela inicial

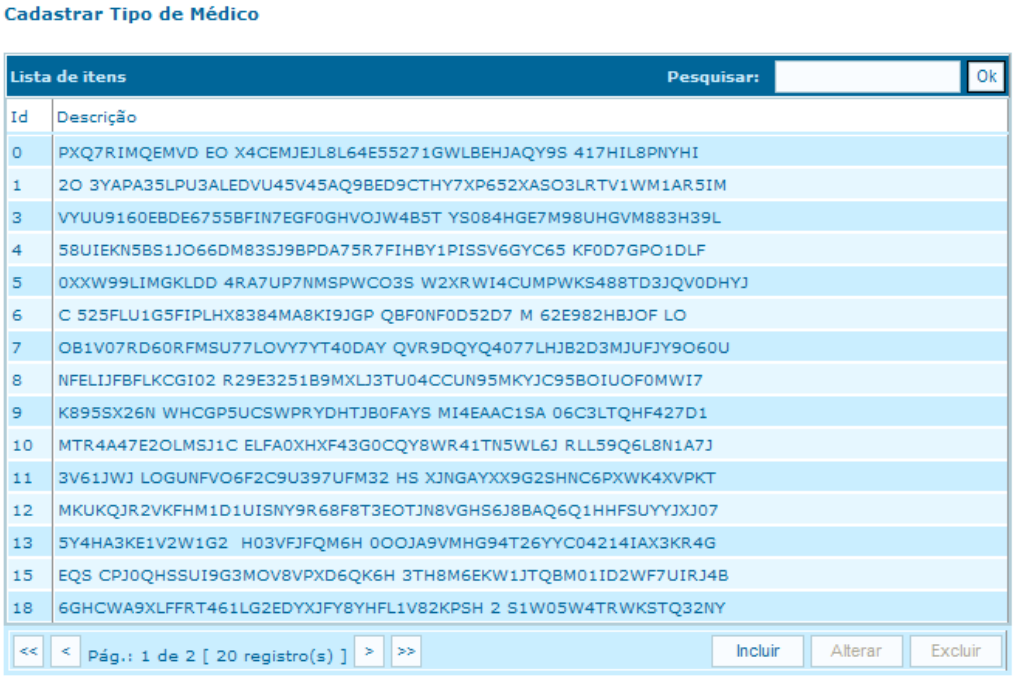

Figura 26 - Cadastro de tabela auxiliar

4.10.5. Caso de Uso Vincular CEP Georeferenciado

Identificador:	CSU-004
Nome:	Vincular CEP Georeferenciado
Objetivo:	Esse caso de uso especifica as funcionalidades básicas de auxilio para vinculação do Cep geo gerenciado com as informações de latitude e longitude na base de dados do sistema.
Ator:	Administrador do sistema ou Técnico de transporte
Fluxo de Eventos:	
Fluxo Básico:	
	O usuário solicita opção Tabela de localização - Cep Georeferenciado. O sistema carrega uma lista vazia em um formulário (id, CEP, latitude e longitude) e habilita os botões INCLUIR e OK do PESQUISAR.
Fluxo Alternativo:	
	Incluir O usuário seleciona o botão incluir. O sistema exibe um formulário(id, cep, latitude,longitude) de cadastro. O sistema habilita os botões BUSCAR NO MAPA, CONFIRMAR, FECHAR e os campos para edição. O usuário preenche os campos desejados. O usuário poderá solicita BUSCAR NO MAPA - Cenário Pesquisa Cep com Mapa. O usuário solicita CONFIRMA. O sistema critica as informações, grava, fecha o formulário, atualiza a lista e habilita o botão INCLUIR.
	Alterar O usuário solicita 4.2.4 - Pesquisar. O sistema executa o cenário Pesquisar O usuário seleciona o registro desejado. O sistema habilita os botões MAPA, INCLUIR, ALTERAR e EXCLUIR. O usuário solicita ALTERAR. O sistema carrega o registro selecionado (id, cep, latitude,longitude) em um formulário com os campos para edição e habilita os botões BUSCAR NO MAPA , CONFIRMAR e FECHAR.

Fluxo Alternativo:	
	O usuário preenche os campos desejados. O usuário poderá solicita BUSCAR NO MAPA - Cenário Pesquisa Cep com Mapa. O usuário solicita CONFIRMA. O sistema critica as informações, grava, fecha o formulário, atualiza a lista e habilita o botão INCLUIR.
	Excluir O usuário solicita a pesquisa. O sistema executa o cenário Pesquisar O usuário seleciona o registro desejado. O sistema habilita os botões INCLUIR, ALTERAR e EXCLUIR. O usuário solicita EXCLUIR. O sistema carrega o registro selecionado (id, cep, latitude,longitude) em um formulário com os campos para edição e habilita os botões CONFIRMAR e FECHAR. O usuário solicita CONFIRMA. O sistema critica as informações, exclui o registro , fecha o formulário, atualiza a lista e habilita o botão INCLUIR.
	Pesquisar O usuário preenche o parâmetro de pesquisa. O sistema realiza simultaneamente a pesquisa que atende o critério de busca naquele momento de preenchimento e carrega em grade o resultado.
	Mapa O usuário solicita a pesquisa O sistema executa o cenário Pesquisar O usuário seleciona o registro desejado. O sistema habilita os botões MAPA, INCLUIR, ALTERAR e EXCLUIR. O usuário poderá solicita Cenário Pesquisa CEP com Mapa. O sistema executa o Cenário Pesquisa Cep com Mapa passando com parâmetro latitude e longitude do registro corrente. O usuário poderá seleciona um novo registro. O sistema atualiza o Cenário Pesquisa CEP com Mapa.
Exceção:	
Ponto de extensão:	
Ponto de inclusão:	

Tela inicial

Cadastrar CEP Georeferenciado

Lista de itens — Pesquisar: [ ] Ok

Id	CEP	Latitude	Longitude
9	72240	-48.12734701	-15.81656966
14	72250	-48.12862428	-15.80085449
10	72260	-48.13523086	-15.79954323
11	72255	-48.13010177	-15.79259084
13	72262	-48.13648549	-15.79108447
12	72261	-48.14005519	-15.79090253
15	72251	-48.12360857	-15.78806547
6	72715	-48.19368437	-15.68647407
8	72705	-48.2022366	-15.68436597
7	72710	-48.19800963	-15.67878287
5	72720	-48.19583905	-15.67204367
4	72735	-48.19367211	-15.66397457
3	72736	-48.19693753	-15.66325901
2	72737	-48.19938145	-15.66304107
1	72738	-48.20216373	-15.66264268

<< < Pág.: 1 de 17 [242 registro(s)] > >> Mapa Incluir Alterar Excluir

Figura 27 - Cadastro de CEP Georeferenciado

Tela de inclusão, alteração e exclusão.

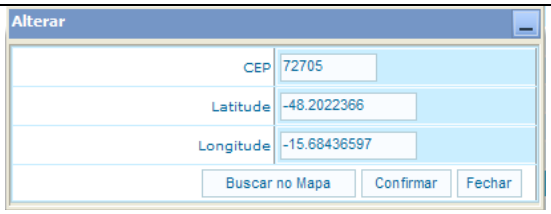

Figura 28 - Manutenção de CEP

Tela do Mapa

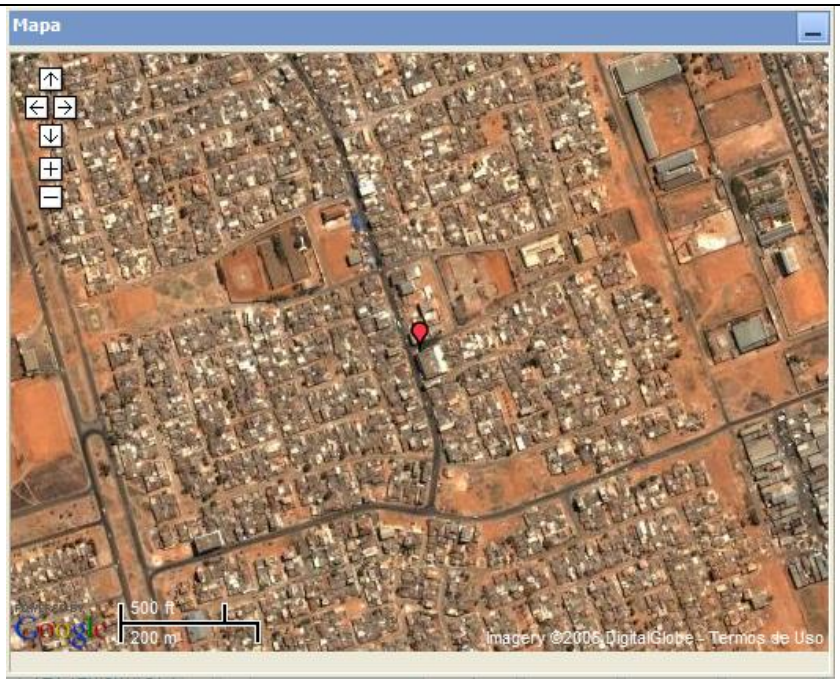

Figura 29 - Mapa CEP

4.10.6. Caso de Uso Visualizar Fila de Espera Georeferenciado

Identificador:	CSU-005
Nome:	Visualizar Fila de Espera Georeferenciado
Objetivo:	Esse caso de uso especifica as funcionalidades básicas de auxilio ao gerenciamento da fila solicitação de ambulância no HUB
Ator:	Técnico de transporte
Fluxo de Eventos:	
Fluxo Básico:	
	O usuário solicita na opção Gestão - Mapa. O sistema exibe mapa de modo padrão e habilita o botão BUSCAR SOLICITAÇÃO (FILA DE ESPERA).
Fluxo Alternativo:	
	Buscar Solicitação O usuário seleciona o botão BUSCAR SOLICITAÇÃO. O sistema executa - Cenário Pesquisar Solicitação. O sistema atualiza as informações de latitude e longitude no mapa recebida pelo cenário pesquisar solicitação e exibe no mapa um símbolo de um BALÃO NA COR VERMELHA correspondente a localidade. O usuário poderá alternar entre formulário principal e cenário pesquisar solicitação.
Exceção:	
Ponto de extensão:	
Ponto de inclusão:	

Tela inicial

Figura 30 - Mapa Gestão

Tela de pesquisa

Figura 31 - Pesquisa

4.10.7. Cenário 001 - Editar Equipamento

Identificador:	Cenário - 001.
Nome:	Editar Equipamento
Objetivo:	A finalidade demonstrar os passos para atualização dos equipamentos necessários para uma determinada tipo de ambulância.
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema verifica a existência de informações cadastrais sobre a veiculo ambulância recebido com parâmetro do **CSU – Administrar Ambulância**. • Caso seja verdadeira, o sistema carrega lista de equipamento(*nome)* em um formulário habilitando o campo para edição e opção de excluir. • Caso contrário, o sistema carrega lista de equipamento(*nome)* em um formulário habilitando o campo para edição e desabilitando opção de excluir. O usuário preenche os campos desejados e solicita a gravação. O sistema critica as informações, grava e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
Fluxo Alternativo:	
	Alteração de Equipamento • O usuário escolhe na lista de equipamento todos os equipamentos que deseja alterar e solicita a gravação. • O sistema critica as informações da lista, grava e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
	Exclusão do equipamento • O usuário seleciona na lista de equipamento todos os equipamentos que deseja excluir através da marcação da opção excluir. • O sistema percorre a lista excluindo esse equipamento e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
Exceção:	**Nenhum.**
Ponto de extensão:	**Nenhum.**
Ponto de inclusão:	**Nenhum.**

4.10.8. Cenário 002 - Editar Materiais

Identificador:	Cenário - 002.
Nome:	Editar Materiais
Objetivo:	A finalidade demonstrar os passos para atualização dos materiais necessários para uma determinada tipo de ambulância.
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema verifica a existência de informações cadastrais sobre a veiculo ambulância recebido com parâmetro do CSU – Administrar Ambulância. • Caso seja verdadeira, o sistema carrega lista de materiais(nome e quantidade) em um formulário habilitando o campo para edição e opção de excluir. • Caso contrário, o sistema carrega lista de materiais (nome e quantidade) em um formulário habilitando o campo para edição e desabilitando opção de excluir. O usuário preenche os campos desejados para cada material e solicita a gravação. O sistema critica as informações, grava e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
Fluxo Alternativo:	
	Alteração de materiais • O usuário escolhe na lista de materiais todos os materiais que deseja alterar e solicita a gravação. • O sistema critica as informações da lista, grava e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
	Exclusão do material • O usuário seleciona na lista de materiais todos os materiais que deseja excluir através da marcação da opção excluir. • O sistema percorre a lista excluindo esse materiais e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
Exceção:	**Nenhum.**
Ponto de extensão:	**Nenhum.**
Ponto de inclusão:	**Nenhum.**

4.10.9. Cenário 003 - Editar manutenção da ambulância

Identificador:	Cenário - 003.
Nome:	Editar manutenção da ambulância
Objetivo:	A finalidade demonstrar os passos para atualização das manutenções efetuada no veiculo ambulância.
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema verifica a existência de informações cadastrais sobre a veiculo ambulância recebido com parâmetro do CSU – Administrar Ambulância. • Caso seja verdadeira, o sistema carrega lista de manutenções(descrição, quilometragem, quantidade e observação) em um formulário habilitando o campo para edição e opção de excluir. • Caso contrário, o sistema carrega lista de manutenções(descrição, quilometragem, quantidade e observação) em um formulário habilitando o campo para edição e desabilitando opção de excluir. O usuário preenche os campos desejados para cada manutenção e solicita a gravação. O sistema critica as informações, grava e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
Fluxo Alternativo:	
	Alteração de manutenção O usuário escolhe na lista de manutenção todas as manutenções que deseja alterar e solicita a gravação. O sistema critica as informações da lista, grava e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
	Exclusão de manutenção O usuário seleciona na lista de manutenção todas as manutenções que deseja excluir através da marcação da opção excluir. O sistema percorre a lista excluindo essas manutenções e retorna para formulário atualizando a lista novamente. O sistema calcula o critério de paginação e exibe quantidade de pagina existente.
Exceção:	**Nenhum.**
Ponto de extensão:	**Nenhum.**
Ponto de inclusão:	**Nenhum.**

4.10.10. Cenário 004 - Pesquisar solicitação de ambulância no sistema

Identificador:	Cenário - 004.
Nome:	Pesquisar solicitação de ambulância no sistema
Objetivo:	A finalidade demonstrar os passos para recuperar as solicitações de ambulância com suas respectivas latitude e longitude para visualização georeferenciada
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema exibe um formulário de pesquisa. O usuário preenche o parâmetro de pesquisa. O sistema realiza simultaneamente a pesquisa que atende o critério de busca naquele momento de preenchimento e carrega em grade o resultado. O usuário seleciona um registro na grade O usuário poderá solicita fecha formulário
Exceção:	**Nenhum.**
Ponto de extensão:	**Nenhum.**
Ponto de inclusão:	**Nenhum.**

4.10.11. Cenário 005 - Pesquisar CEP com Mapa

Identificador:	Cenário - 005.
Nome:	Pesquisar CEP com Mapa
Objetivo:	A finalidade demonstrar os passos para buscar um cep com suas respectivas latitude e longitude com auxilio do mapa.
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema exibe um mapa com as suas características de funcionalidades básicas. O usuário seleciona a localidade desejada com um clique sobre a superfície do mapa. O sistema atualiza as informações de latitude e longitude na tela anterior e exibe no mapa um símbolo de um BALÃO NA COR VERMELHA. O usuário solicita fecha formulário. O sistema fecha o formulário.
Fluxo Alternativo:	
	Alterar Caso o sistema recebe os parâmetros de latitude e longitude exibe o mapa um símbolo de um BALÃO NA COR VERMELHA com suas respectivas referências de latitude e longitude. O usuário seleciona a localidade desejada com um clique sobre a superfície do mapa. Mas as informações anteriores continuam congeladas. O sistema atualiza as informações de latitude e longitude na tela anterior e exibe no mapa um símbolo de BALÃO NA COR VERMELHA. O usuário solicita fecha formulário. O sistema fecha o formulário.
	Visualizar latitude e longitude Caso o sistema recebe os parâmetros de latitude e longitude exibe o mapa um símbolo de um BALÃO NA COR VERMELHA com suas respectivas referências de latitude e longitude. O usuário solicita fecha formulário O sistema fecha o formulário
Exceção:	**Nenhum.**
Ponto de extensão:	**Nenhum.**
Ponto de inclusão:	**Nenhum.**

4.10.12. Cenário 006 - Trajeto da solicitação

Identificador:	Cenário - 006.
Nome:	Trajeto da solicitação
Objetivo:	Esse caso de uso especifica as funcionalidades básicas para registrar as informações do trajeto das ambulância para uma determinada solicitação.
Fluxo de Eventos:	
Fluxo Básico:	
	O sistema exibe um formulário(Data/hora, origem e destino) de cadastro. O sistema habilita os botões CONFIRMAR e FECHAR e os campos para edição. O usuário preenche os campos desejados e solicita confirma. O sistema critica as informações, grava, fecha o formulário.
Fluxo Alternativo:	
	Alterar O sistema carrega o trajeto recebido com parâmetro em um formulário(Data/hora, origem e destino) com os campos para edição e habilita os botões CONFIRMAR E FECHAR. O usuário altera os campos desejados e solicita confirma. O sistema critica as informações, grava e fecha o formulário.
	Excluir O sistema carrega o trajeto recebido com parâmetro em um formulário(Data/hora, origem e destino) com os campos para edição e habilita os botões CONFIRMAR E FECHAR. O usuário solicita confirma. O sistema critica as informações, excluir e fecha o formulário.
Exceção:	O sistema informa ao usuário uma mensagem de erro no formulário corrente a obrigatoriedade todos campos.
Ponto de extensão:	**Nenhum.**
Ponto de inclusão:	**Nenhum.**

4.11. Decomposição dos Casos de Uso

4.11.1. CSU 001 – Gerenciar Solicitação

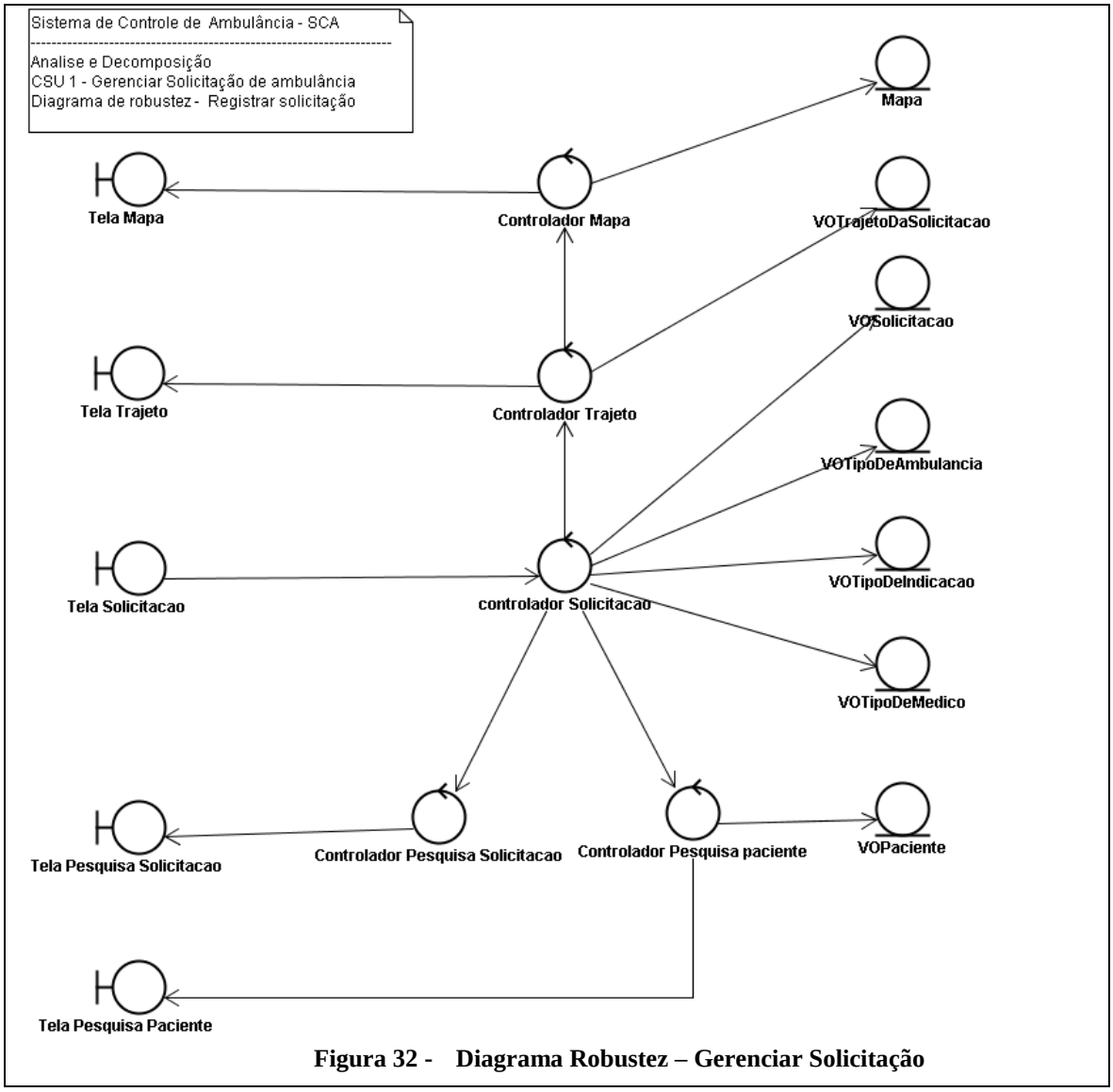

Figura 32 - Diagrama Robustez – Gerenciar Solicitação

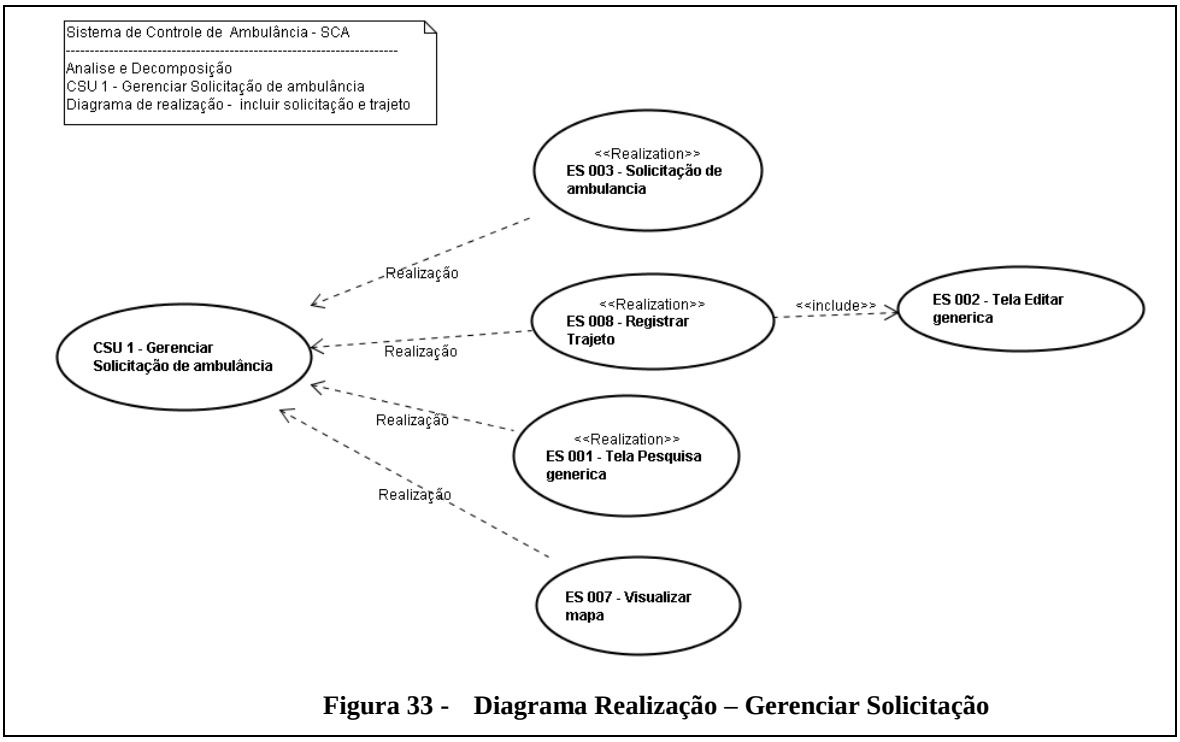

Figura 33 - Diagrama Realização – Gerenciar Solicitação

4.11.2. CSU 002 – Administrar Ambulância

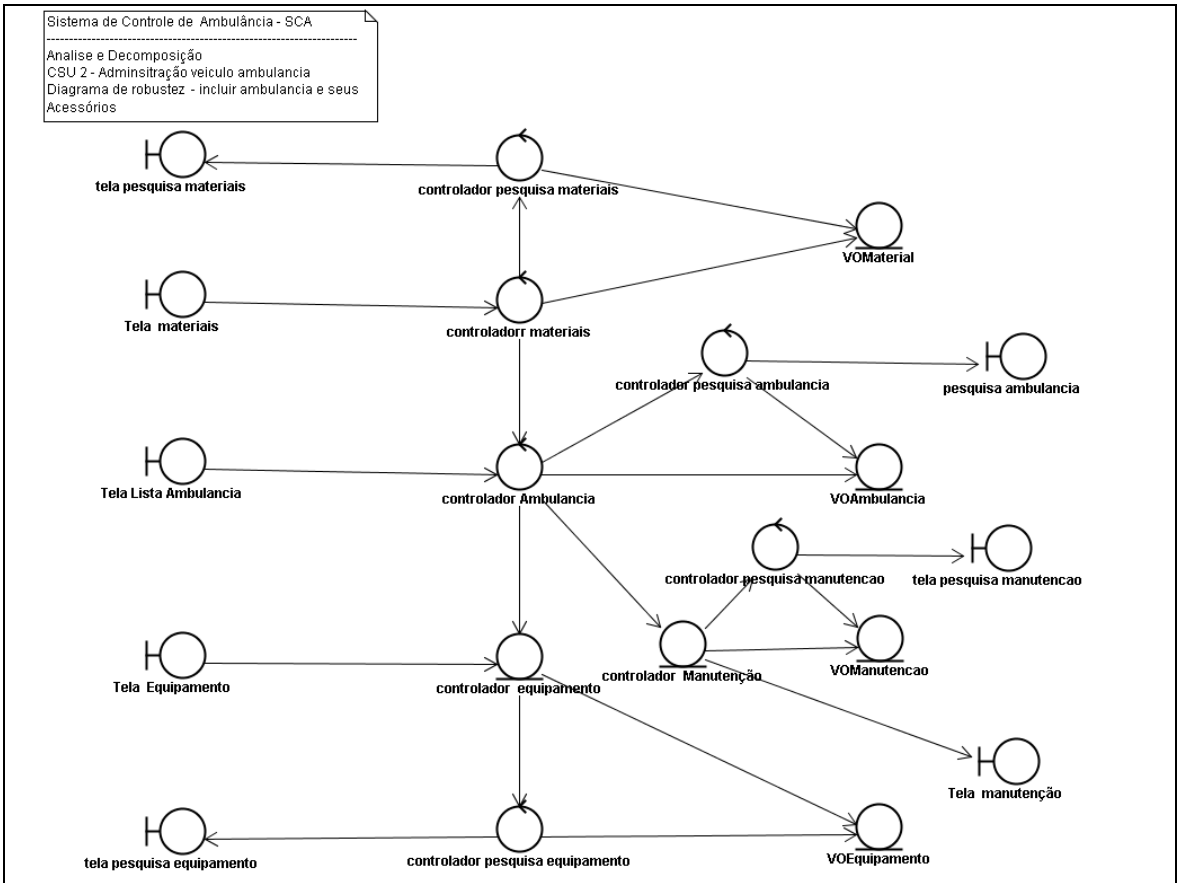

Figura 34 - Diagrama Robustez – Administrar Ambulância

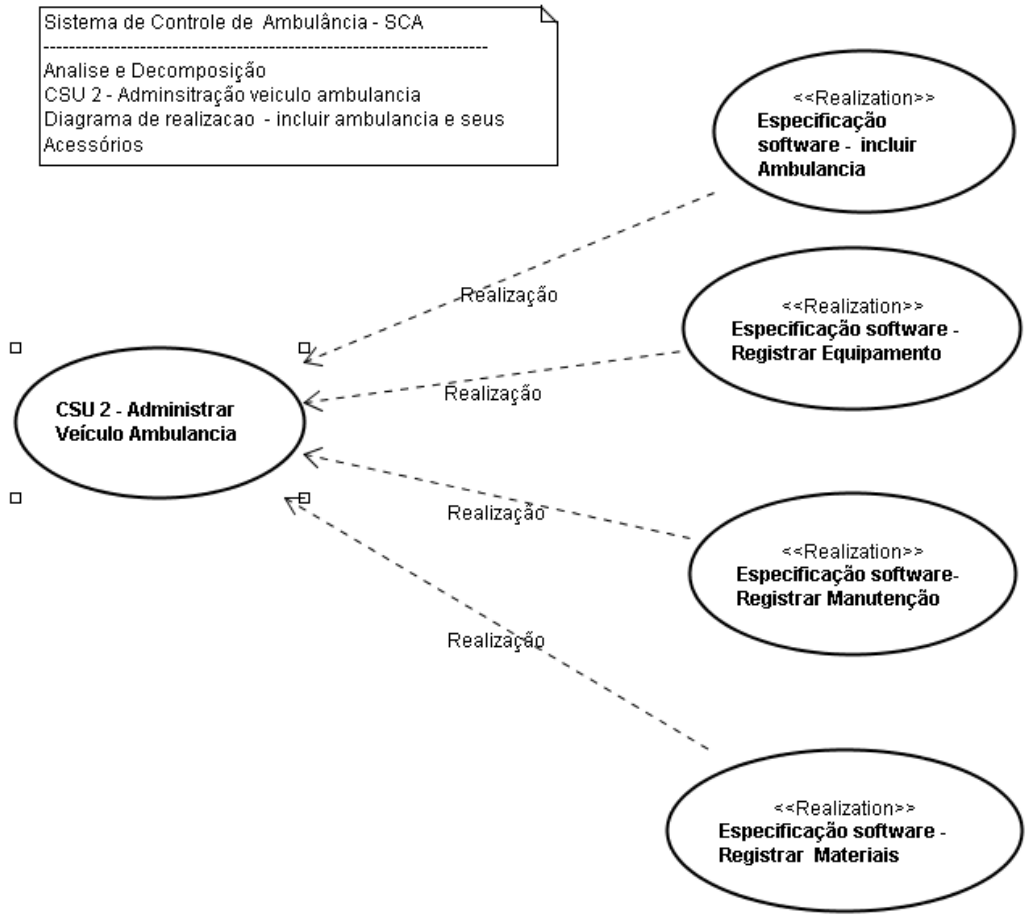

Figura 35 - Diagrama Realização – Administrar Ambulância

4.11.3. CSU 003 – Manter Tabelas Auxiliares

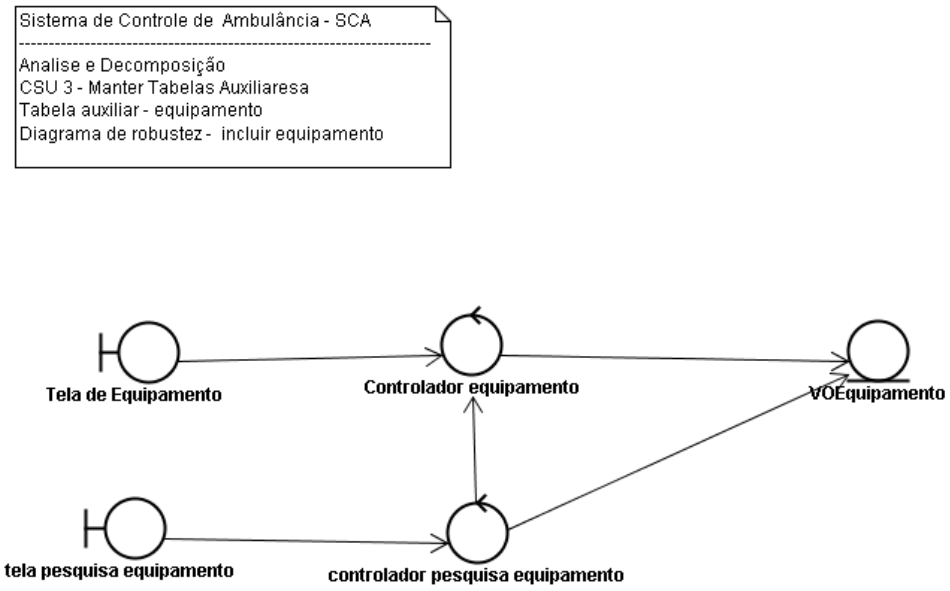

Figura 36 - Diagrama Robustez – Manter Tabelas Auxiliares

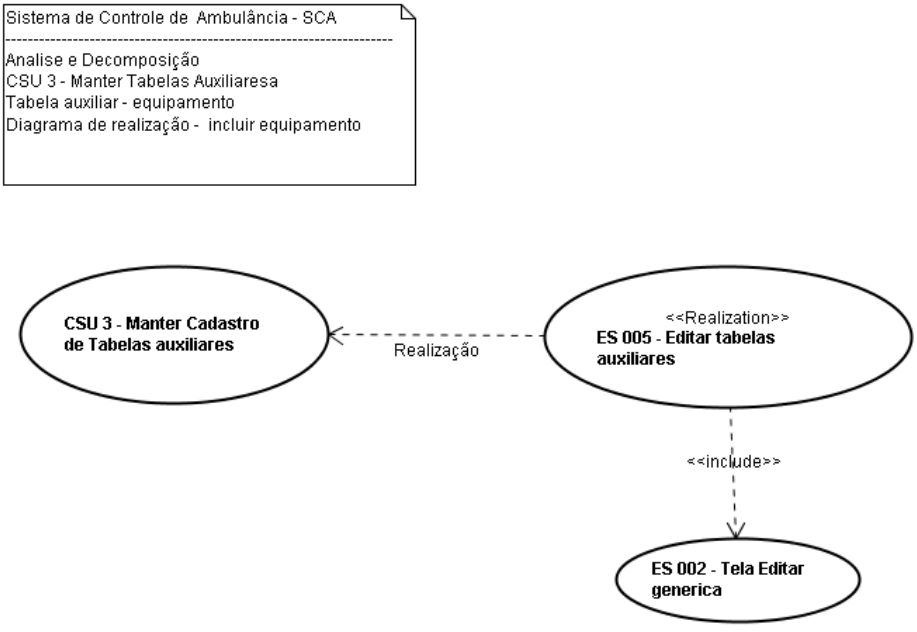

Figura 37 - Diagrama Realização – Manter Tabelas Auxiliares

4.11.4. CSU 004 – Vincular Cep Geograficamente

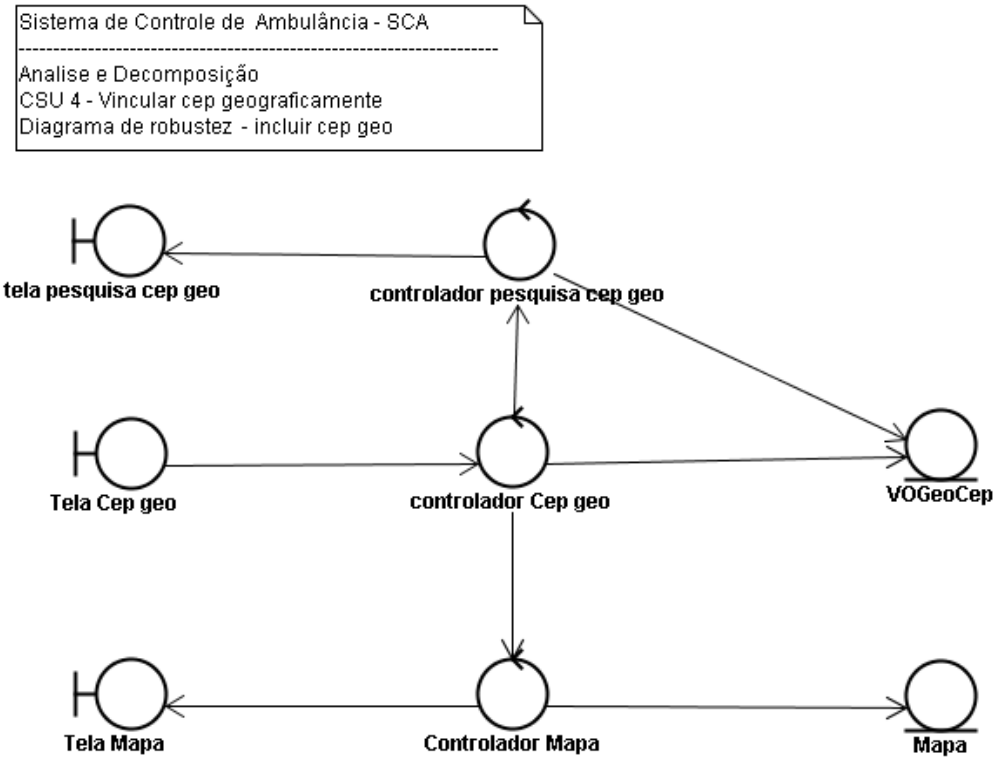

Figura 38 - Diagrama Robustez – Vincular Cep Geograficamente

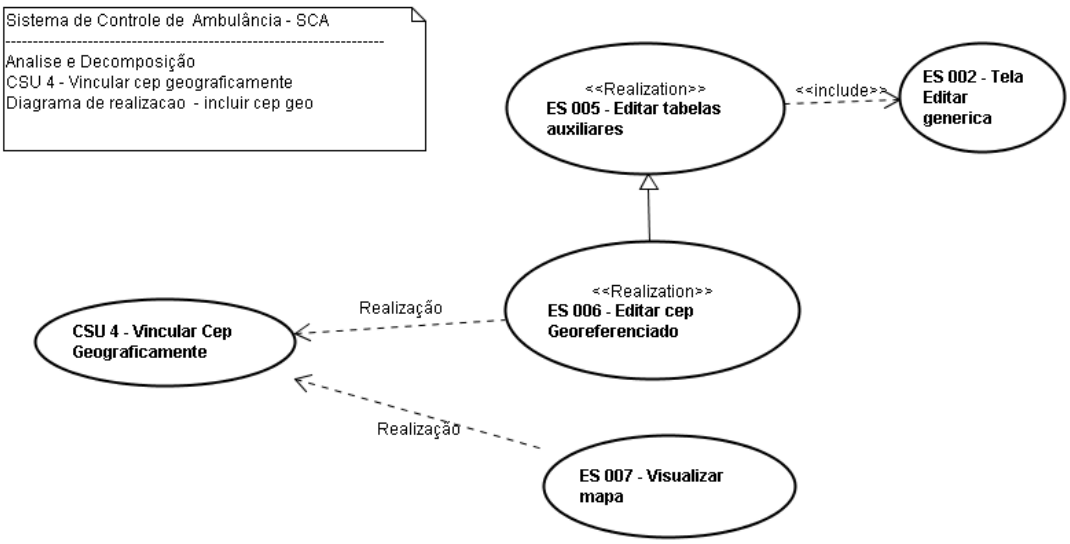

Figura 39 - Diagrama Realização – Vincular Cep Geograficamente

4.11.5. CSU 005 – Vincular Fila Espera Georeferenciado

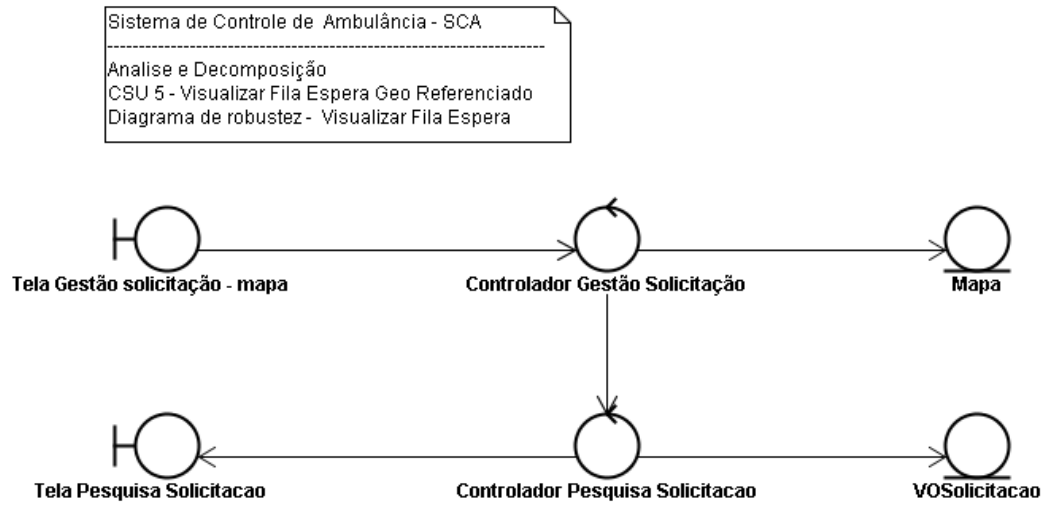

Figura 40 - Diagrama Robustez – Vincular Fila Esperar Georeferenciado

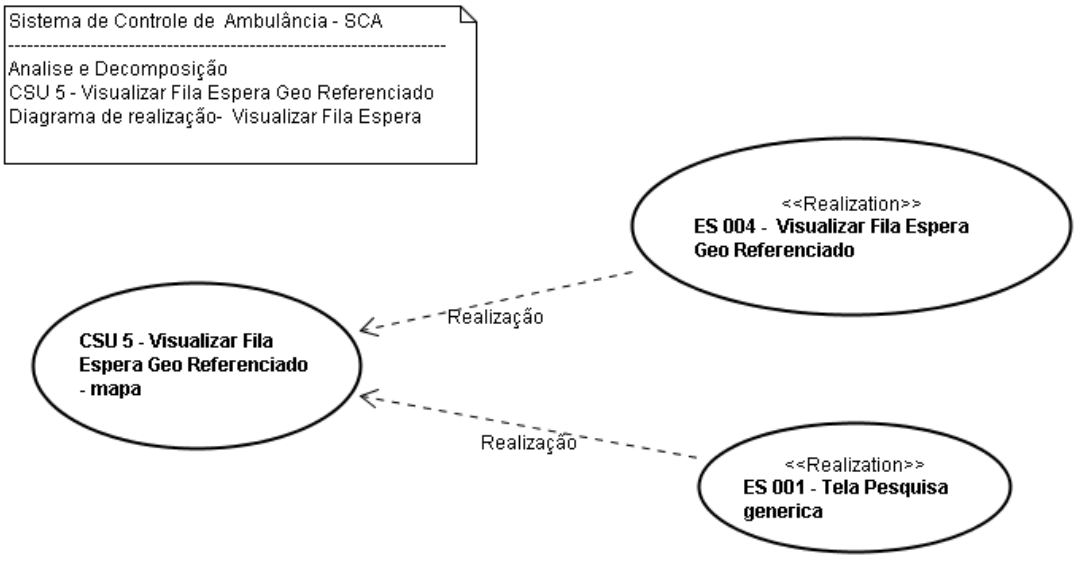

Figura 41 - Diagrama Realização – Vincular Fila Esperar Georeferenciado

4.12. Diagrama de Componentes

O diagrama de componentes representa os componentes que farão parte dos sistema em construção, demonstrando as dependências entre esses componentes. Todo o sistema orientado a objetos terá, dependendo do tamanho, dezenas ou centenas de componentes que em conjunto darão a todos os processos de negócios desse sistema. A principais características do diagrama de componentes :

- Demonstra os componentes de um sistema, com as dependências entre eles.
- Podem ser representados componentes do tipo arquivo, documentos, tabelas, bibliotecas, entre outros artefatos de um sistema.

O diagrama de componentes do tipo fontes de sistemas para fazer a modelagem dos códigos-fontes de um sistema. Dessa forma, terá especificar e documentar a estrutura de componentes utilizados na construção do sistema.

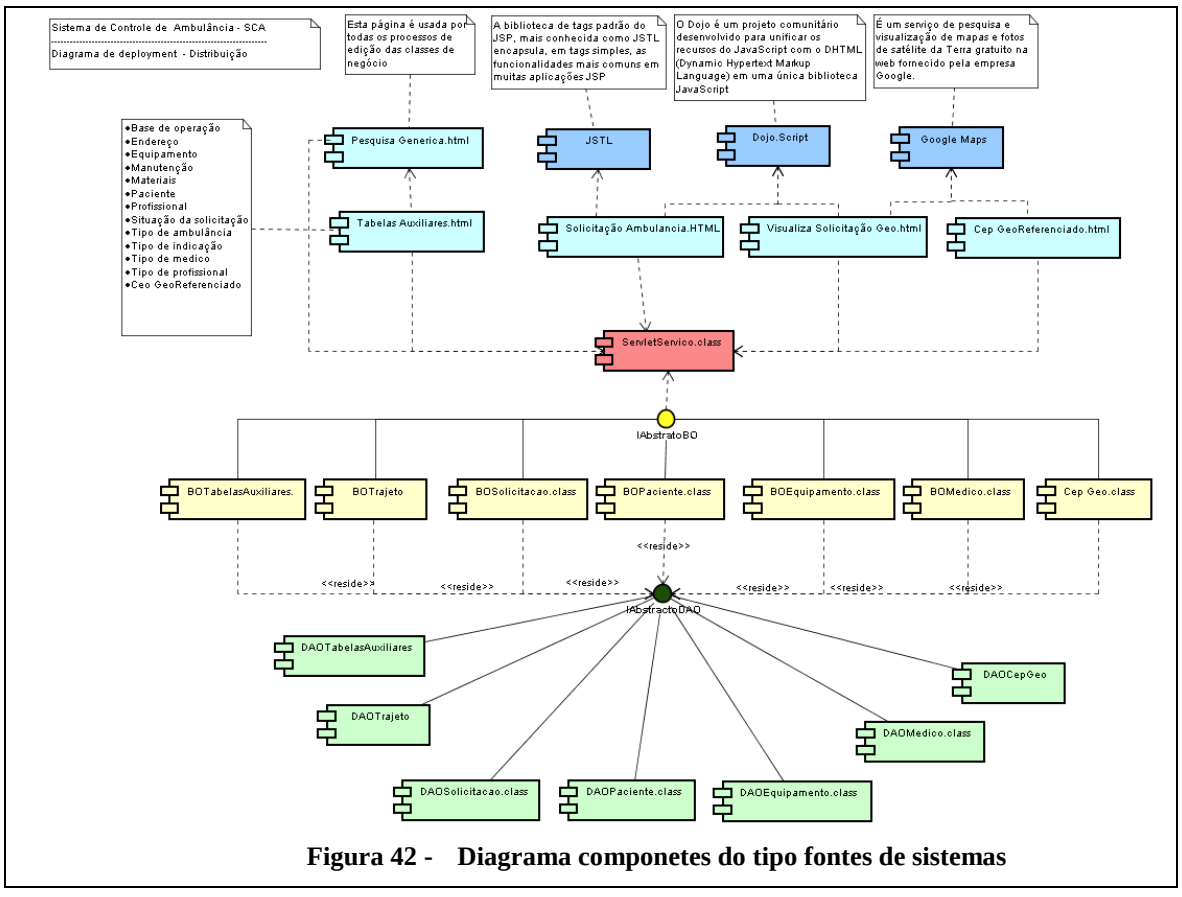

Figura 42 - Diagrama componetes do tipo fontes de sistemas

4.13. Diagrama de Implantação

O diagrama de implantação representa a configuração e a arquitetura de um sistema em que estarão ligados seus respectivos componentes, podendo ser representado também arquitetura física de hardware, processadores, etc. O diagrama de implantação de um sistema, envolvendo a modelagem da topologia de hardware em que o sistema será executado.

As principais características do diagrama de implentação:

- Pode representar a estrutura da plataforma em que o sistema será executado.
- Pode representar qualquer dispositivo, como gerenciador de banco de dados, servidores, computadores, ect.

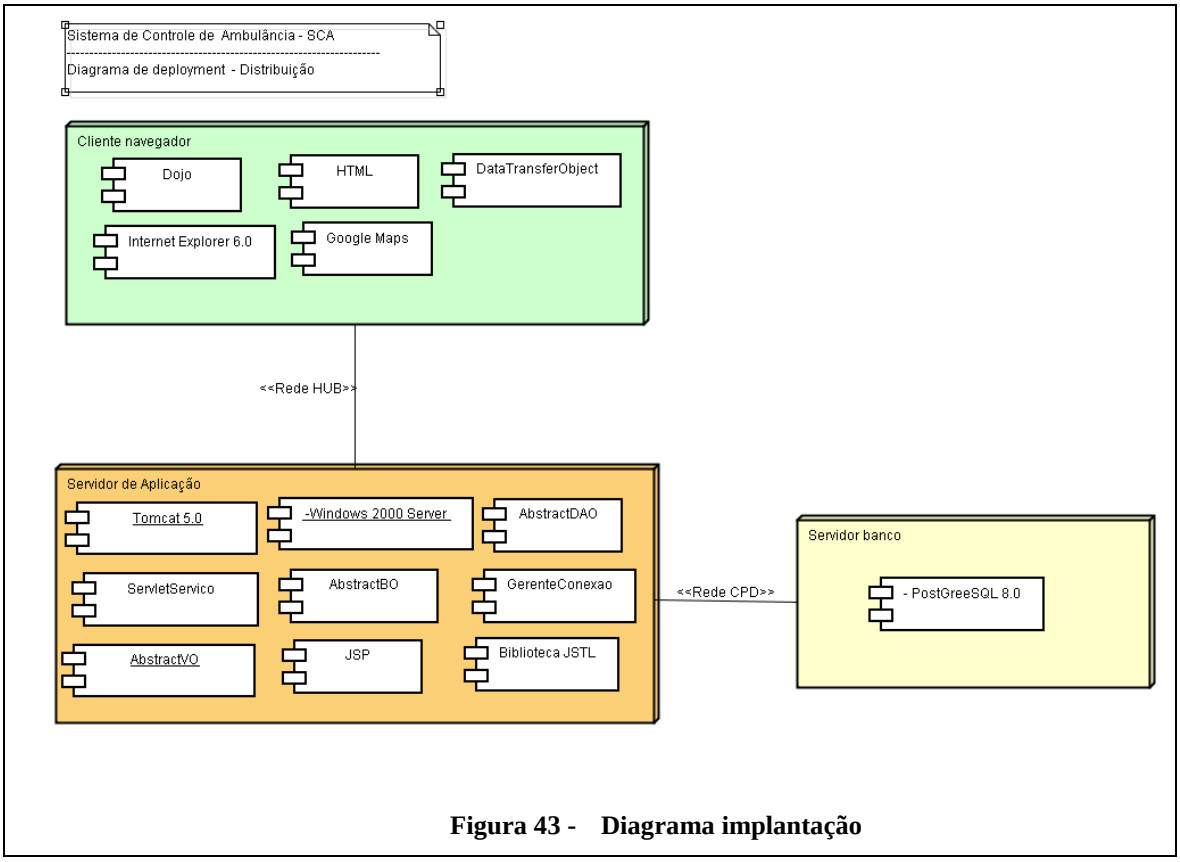

Figura 43 - Diagrama implantação

5. CONCLUSÕES

Neste capítulo serão apresentadas algumas considerações sobre o desenvolvimento da aplicação usando AJAX e os resultados obtidos com o estudo.

a) Usabilidade;

A usabilidade é potencialmente o grande beneficio a se obter no uso de Ajax para o desenvolvimento de aplicações complexas, pois o número de interações nesse tipo de tela é muito grande, mas o volume dados é geralmente pequeno.

b) Desempenho;

A melhoria do desempenho em AJAX é progressiva, pois a primeira página certamente será mais lenta, pois precisa carregar mais código do que em uma página clássica, mas em telas complexas isso é amplamente compensado pela diminuição e diluição das chamadas subseqüentes. Além disso, técnicas de compressão de código e de modularização, pode reduzir substancialmente essa sobrecarga inicial.

c) Segurança;

Os problemas de segurança em Ajax não estão ligados diretamente à técnica, pois a insegurança se deve ao meio em que o código é executado e a sua comunicação via TCP/IP. Entretanto, códigos mão elaborados abrem caminho para códigos maliciosos, via Javascript injection e SQL injection. A adoção de boas práticas e fundamental para desenvolver um código seguro em Ajax, da mesmo forma que é verdade para a web clássica.

e) Manutenibilidade;

A manutenibilidade é o grande risco no desenvolvimento em Ajax, pois se as melhores práticas não forem seguidas no momento do desenvolvimento existe a grande possibilidade de se ter código inconsistente por todas as páginas do sistema. Novamente, o problema aqui não é da técnica, mas sim do processo de desenvolvimento e da falta de ferramentas maduras.

6. Referências Bibliográficas.

[1] **AJAX para quem só ouviu falar**. Disponível em : <http://www.tableless.com.br/artigos/ajaxdemo/>. Acesso em : 19/05/2006

[2] ASLESON, Ryan; SCHUTTA, Nathaniel T. **Fundamentos do Ajax**. Alta Books. 2006.

[3] BRUCE, Eckel. **Thinking in Java**. 4 ed. Massachusetts: Editora Prentice hall: 2006.

[4] **COLOR Scheme**. Disponível em : <http://www.colorschemer.com/osx_info.php>. Acesso em: 18/05/2006

[5] **CSS para Web Design**. Disponível em : <http://www.maujor.com/index.php>. Acesso em: 20/05/2006

[6] DAVE, Crane; PASCARELLO, Eric; JAMES, Darren. **Ajax in Action**. Greenwich : Manning. 2006.

[7] DEITEL, H.M. DEITEL, P.J. **Java, como programar**. Tradução por Edson Furnankiewicy. 3 ed. Porto Alegre : Editora Bookman: 2001.

[8] FAEMAN, Julio. **Design Patterns Aplicados. Java Magazine**, Rio de Janeiro, v. 3, n.20, p. 52 a 58, 2005.

[9] FLANAGAN, David. **Javascript – O Guia Definitivo**. Artmed – Bookman. 2004.

[10] FOWLER, Martin. **UML essencial**: um breve guia para a linguagem – padrão de modelagem de objetos. Tradução Vera Pezerico e Christian Thomas Prices. 3. ed. Porto Alegre: Bookman, 2005.

[11] Freeman, Elisabeth. **Use a cabeça – HTML com CSS e XHTML**. Alta Books. 2006.

[12] GOOMAN, Danny. **Javascript – A Bíblia**. Elsevier. 2001.

[13] **HTMLHttpRequest v1.0beta2**. Disponível em: <http://twinhelix.com/javascript/htmlhttprequest/>. Acesso em : 19/05/2006

[14] JOHN, Metsker, Steven**. Padrões de projeto em java**. Porto Alegre: Editora Booklman. 2004.

[15] JUNIOR, Francisco B. **JSP : a Tecnologia Java na Internet**. São Paulo : Editora Érica. 2001.

[16] LEME, Fernando. **Servlets : parte 1: conceitos e técnicas básicas**. Java Magazine, Rio de Janeiro, v. 3, n.18, p. 32 a 42, 2005. Pente Fino.

[17] LOZANO, Fernando. **Aplicações Web no Tomcat 5: parte 1: Primeiros passos no desenvolvimento JSP.** Java Magazine, Rio de Janeiro, v. 3, n.18, p. 20 a 31, 2005. Java Livre.

[18] MAZETTI, Geraldo. **HTML com XML**. Pearson Education do Brasil. 2000.

[19] MENDES, Antonio. **Arquitetura de software: desenvolvimento orientado para arquitetura**. Rio de Janeiro: Editora Campus. 2002.

[20] **PADRÕES estrutura do XML**. Disponível em: <http://www.gta.ufrj.br/grad/00_1/miguel/link7.htm>. Acesso em: 20/05/2006

[21] REBITTE, Leonardo. **Dominando Tableless**. Alta Books. 2006.

[22] SANTOS, Alexandre Denes dos Santos. **Conhecendo AJAX : turbine a interatividade de suas aplicações web**. Java Magazine, Rio de Janeiro, v. 3, n.28, p. 43 a 49, 2005.

[23] **SCRIPTS Brasil**. Disponível em : <http://phpbrasil.com/scripts/index.php>. Acesso em : 20/05/2006

[24] SIERRA, kathy: BATES, Bert. **Head First Java 5.** 2 ed. Sebastopolo: Editora O' Reilly Media. Books. 2005.

[25] SIERRA, kathy: BATES, Bert. **Head First Servlets & JSP**. 1 ed. Sebastopolo: Editora O' Reilly Media. Books. 2004.

[26] SILVA, Osmar J. HTML 4.0 e XHTML 1.0**: Domínio e Transição**. São Paulo : Editora Érica. 2001.

[27] SILVEIRA, Paulo. **Eclipse 4 : Dicas e Truques**. Mundo Java, Curitiba, v. 2, n.09, p. 32 a 41, 2005. Capa.

[28] SOARES, Walace. **Ajax – Guia Prático para Windows**. Érica. 2006.

[29] TEAGUE, Jason C. DHTML **e CSS para a World Wide Web**. Elsevier. 2001.

[30] TODD, Nick, **Java Server Pages : O Guia do Desenvolvedor**. tradução de Edson Furmankewicy. Rio de Janeiro: Editora Campus. 2003.

[31] **TUTORIAL HTML**. Disponível em : <http://www.icmc.usp.br/ensino/material/html/>. Acesso em : 19/05/2006

[32] YUNG, Leandro. **Em dia com Java**. Mundo Java, Curitiba, v. 2, n.09, p. 12 a 17, 2005. Professor J.