Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro

Autor:

_____

Romulo Vanzillotta Castello

Orientador:

_____

Prof. Antônio Cláudio Gómez de Sousa, M. Sc.

Examinador:

_____

Prof. Sérgio Barbosa Villas Boas, D. Sc.

Examinador:

_____

Prof. Aloysio de Castro Pinto Pedroza, Dr.

DEL

Agosto de 2009

DEDICATÓRIA

Dedico o presente trabalho aos meus pais, que tanto se empenharam na minha formação enquanto pessoa e em viabilizar para que eu pudesse chegar até aqui, e ao cão Astro, que me acompanha desde menino.

AGRADECIMENTO

Agradeço ao povo brasileiro que contribuiu de forma significativa ao financiamento da minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir os recursos financeiros e humanos e a confiança investidos em minha pessoa.

RESUMO

Este trabalho compreende a descrição de um sistema de computação e a documentação de seu projeto, assim como o acompanhamento ao longo de seu desenvolvimento.

O projeto utiliza técnicas de excelência para o desenvolvimento do projeto de um software, independente de seu tamanho, baseando-se em normas confiáveis e objetivas, e nas atuais tendências na área do desenvolvimento de sistemas online e colaborativos.

O sistema destina-se à prefeitura do município do Rio de Janeiro para o cadastro de irregularidades percebidas pelos cidadãos e pelos servidores, sendo uma espécie de ouvidoria eletrônica moderna.

Palavras-Chave: desordem urbana, c++, javascript, colaborativo, mash-ups.

ABSTRACT

This article consists of the description of a computing system and the documentation of its project, as well as the tracking along of its development.

The project uses excellent techniques for the development of a software project, regardless of its size, based on reliable standards, and current trends in the development of online systems and collaborative.

The system is intended to the city hall of Rio de Janeiro to record irregularities observed by citizens and by officials, similar to a modern electronic ombudsman service.

Key-words: urban disorder, c++, javascript, collaborative, mash-ups.

SIGLAS

API – Application Programming Interface, ou Interface de Programação de Aplicativos. Interface que define os meios através dos quais um sistema pode requerer serviços ou bibliotecas de outros sistemas.

CGI – Common Gateway Interface, ou Interface de Passagem Comum. Mecanismo de acesso através dos provedores http para programação e registros de informações na Internet;

HTML – HyperText Markup Language ou linguagem HTML. Código para veicular textos e imagens na Internet.

SMDU-RJ – Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro, nome do software;

UML – Unified Modeling Language, acronismo inglês para Linguagem de Modelagem Unificada. Ela é não-proprietária e de terceira geração, o que significa que é aberta, sem custos e auxilia a visualizar a comunicação entre os objetos como um todo;

UFRJ – Universidade Federal do Rio de Janeiro

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 – Tema

O tema do trabalho é a confecção de um sistema de computação para tratar da desordem urbana existente no município do Rio de Janeiro [1]. Desta grande área da computação, serão abordados principalmente os conhecimentos sobre bancos de dados, programação orientada a objeto, sistemas online, colaborativos e que se comunicam com outros através de interfaces, entre outros.

1.2 – Delimitação

O escopo do trabalho são as irregularidades percebidas no município do Rio de Janeiro, não estando previstas as de outras localidades. Ele deve ser usado tanto pelos cidadãos quanto pelos funcionários da prefeitura como base de dados. A quantidade de módulos (tipos de irregularidades) é ilimitada, porém, no momento, foi implementado o módulo de cadastro de buracos nas vias do município.

1.3 – Justificativa

A organização do espaço público proporciona maior qualidade de vida para as pessoas em geral, mais impostos ao governo (que por conseguinte voltariam como investimento, gerando um círculo virtuoso), evita a queda de encostas, como a acontecida sobre o Túnel Rebouças, e inundações, minimiza engarrafamentos e prejuízos aos veículos, estimula uma melhor prestação de serviços e a transferência de trabalhadores ilegais à legalidade (através de capacitações, por exemplo, não fazendo parte do escopo deste trabalho), entre outras melhorias. Em resumo, contribui fortemente para o desenvolvimento da nação, já que a cidade do Rio de Janeiro está na vanguarda do país e certamente serve de exemplo para as outras.

1.4 – Objetivos

O objetivo geral foi, então, propor um sistema capaz de servir como base de dados das irregularidades existentes, tanto aos cidadãos como aos funcionários da prefeitura, de forma prática e usável. Aos funcionários, se presta a guiar seu trabalho de ações corretivas e preventivas, e aos cidadãos se presta como um alerta enquanto as situações não forem resolvidas, além de um canal direto com a prefeitura. Desta forma, pretende-se contribuir para conter e eliminar a crescente desordem presente na cidade do Rio de Janeiro.

1.5 – Metodologia

O projeto seguiu o paradigma orientado a objetos de codificação, usando a UML [2] para sua modelagem. Tem seu ciclo de vida em espiral [3], tendo dado uma volta. Nesta primeira volta, foi produzida a primeira versão do software.

Este trabalho procurou seguir as atuais tendências da tecnologia de desenvolvimento para *softwares online* e *open-source*. Eles são muito mais dinâmicos no sentido de não necessitarem qualquer tipo de instalação, sistema operacional ou navegador específico. Contanto que se tenha acesso a um navegador contemporâneo, faz-se possível o uso do sistema de uma máquina em qualquer lugar do mundo.

Além disto, ele é colaborativo e segue as novas tendências da chamada *web 2.0*. Nesta linha, são os próprios usuários que constroem sua base de dados a ser processada e visualizada, o que os faz se sentirem parte da construção do *software* em questão, tornando seu uso mais agradável, consciente e até mesmo democrático.

Outra tendência atual levada em consideração é a dos *mash-ups*, programação que utiliza *APIs* desenvolvidas previamente para se adicionarem novos recursos e funcionalidades, e as usam como base. Neste caso, a *API* utilizada foi a do *Google Maps* [4]. Esta tecnologia é muito útil e permite uma gama de novas possibilidades.

Para a codificação foi usada a linguagem C++ [5], por ser complexa, possuir variadas funcionalidades e ter se desenvolvido o bastante a ponto de poder ser usada em larga escala. A interface no navegador foi programada em HTML e Javascript [6], os quais permitem um carregamento dinâmico das páginas, que mudam na ocorrência de determinados eventos sem a necessidade de se carregar outra página. O banco de dados

é o *open-source PostgreSQL* [7], reforçando a opção pelos programas gratuitos. As bibliotecas usadas foram a VBLib [8] e a VBMCgi [9], ambas gratuitas. A última permite uma divisão em três camadas da programação orientada à *web*. Já para fazer a ligação entre o banco de dados e as CGIs, foi usada a interface SQLAPI++. O sistema está hospedado *online* utilizando o *HTTP Server*, também gratuito. A validação faz uso da já experimentada tecnologia de *cookies*.

Para a confiabilidade, espera-se que o mesmo seja integrado futuramente com o cadastro de contribuintes da prefeitura e também desenvolvido um sistema de pontuação para avaliar a validade (e importância) das irregularidades cadastradas.

Desta forma, o objetivo foi produzir um *software* confiável, de baixo custo e com utilização de baixa complexidade, sem, no entanto, comprometer sua usabilidade por essas condições.

1.6 – Descrição do Documento

Os capítulos de 2 a 4 descrevem o que será encontrado nos documentos do apêndice e as atividades realizadas durante a sua confecção.

No capítulo 5 será descrito o desenvolvimento do software em questão, seus acertos e problemas, e, no capítulo 6, serão descritos os resultados.

No apêndice A será apresentado o Plano para o Gerenciamento de Projeto de Software. Este documento visa a definir regras claras para o projeto deste software, a fim de mitigar dúvidas e riscos e possibilitar um acompanhamento mais estreito do mesmo.

Já o apêndice B contempla o Especificação de Requisitos de Software, o qual se destina a definir claramente os requisitos do software em questão, incluindo as perspectivas, funções, dependências e interfaces do produto para a minimização das insatisfações por parte do usuário e de erros de implementação. Tem como público alvo os clientes e os analistas e programadores responsáveis pelo sistema.

No apêndice C, por sua vez, pode ser verificada a Descrição do Projeto de Software. Nela é explicitada a decomposição e detalhamento do sistema em módulos, em processos concorrentes e em dados, como também as dependências e interfaces destas entidades.

Finalmente, a conclusão expõe os resultados e as percepções ao decorrer do projeto, analisando se o objetivo foi atingido.

Capítulo 2

Descrição do Plano para o Gerenciamento de Projeto de Software

2.1 – Atividades

Esta é a etapa inicial do projeto, onde se desenvolvem os mecanismos para o planejamento e acompanhamento do mesmo. A seguir uma descrição detalhada dos principais itens que compõem o Plano.

2.2 – Sumário do Projeto

No item Sumário do Projeto são listados as finalidades, o escopo e o objetivo do projeto e do produto a ser desenvolvido, as restrições impostas ao projeto, como prazo, orçamento e tecnologias, os produtos liberados durante o desenvolvimento e o cronograma e orçamento macros.

2.3 – Organização do Projeto

No item Organização do Projeto podem ser encontradas as interfaces externas, ou seja, as fronteiras entre o projeto e as organizações externas; a estrutura interna do projeto, que se resume no relacionamento com entidades que suportam seus processos; e a definição das grandes atividades para seus processos e atividades, com papéis e responsabilidades.

2.4 – Processos de Gerenciamento

Na parte de Processos de Gerenciamento são verificados os itens da partida do projeto (*kick-off*), como previsões, equipe, aquisição de recursos e treinamento da equipe, os

itens do plano de trabalho, como o detalhamento das atividades, prazos e alocação de recursos e orçamento, os planos de controle de requisitos, prazos, orçamento, riscos e qualidade, e os planos de relatórios, medidas e encerramento.

2.5 – Processos Técnicos

Neste item serão explicitados os modelos de processo, com as principais atividades, revisões e marcos; as metodologias de desenvolvimento, ferramentas e técnicas para a realização do projeto, como também a infra-estrutura necessária, ambiente, bibliotecas, normas, etc.; e, finalmente, o plano para aceitação do produto.

2.6 – Planos para os processos de Suporte

O gerenciamento da configuração, o plano de verificação e de validação, a organização e formato da documentação gerada, o plano para assegurar a qualidade do processo e do produto, as revisões e auditorias, o plano para resolução de problemas, o gerenciamento de possíveis sub-contratações e o plano de aperfeiçoamento dos processos, ferramentas e pessoal serão apresentados neste tópico.

2.7 – Norma e Produto Final

Este planejamento segue a norma "PLANO PARA O GERENCIAMENTO DE PROJETO DE SOFTWARE – PGPS" constante da página online de Antônio de Sousa [10].
O produto final pode ser encontrado no apêndice A.

Capítulo 3

Descrição da Especificação de Requisitos de Software

3.1 – Atividades

Esta é a etapa intermediária do projeto, traduzindo-se como a análise do problema. Nesta fase foram entrevistados vários futuros clientes do sistema, cidadãos do município do Rio de Janeiro, e listada suas necessidades e opiniões sobre os requisitos sugeridos.

Sua finalidade é definir claramente os requisitos do software em questão para a minimização das insatisfações por parte do usuário e erros de implementação. Destina-se, portanto, aos clientes, analistas e programadores responsáveis pelo sistema.

3.2 – Descrição Geral

Neste item são descritos a perspectiva do produto, com as possíveis dependências a outros aplicativos ou sistemas, suas funções, as características dos pretensos usuários, as restrições, como limitações de hardware, interfaces, segurança, etc., os pressupostos que podem causar impactos ao projeto e os requisitos que podem ser postergados.

3.3 – Requisitos Específicos

Esta parte apresenta as interfaces externas (com usuários, hardware, software e comunicação), os requisitos funcionais, os requisitos de desempenho, as restrições do projeto e os atributos necessários, como amigabilidade, segurança, confiabilidade, etc.

Como o paradigma é orientado a objeto, a análise dos requisitos funcionais for feita por meio da criação do diagrama de classes do sistema e do diagrama dos casos de uso, os quais são mostrados a seguir.

❖ Diagrama de Casos de Uso

A figura abaixo representa o diagrama de casos de uso. Existem 3 tipos de atores no sistema e 5 eventos principais.

Atores: Visitante, Cidadão e Servidor

Eventos: Visualizar Irregularidade, Cadastrar Usuário, Cadastrar Irregularidade, Módulo Buracos, Excluir Irregularidade

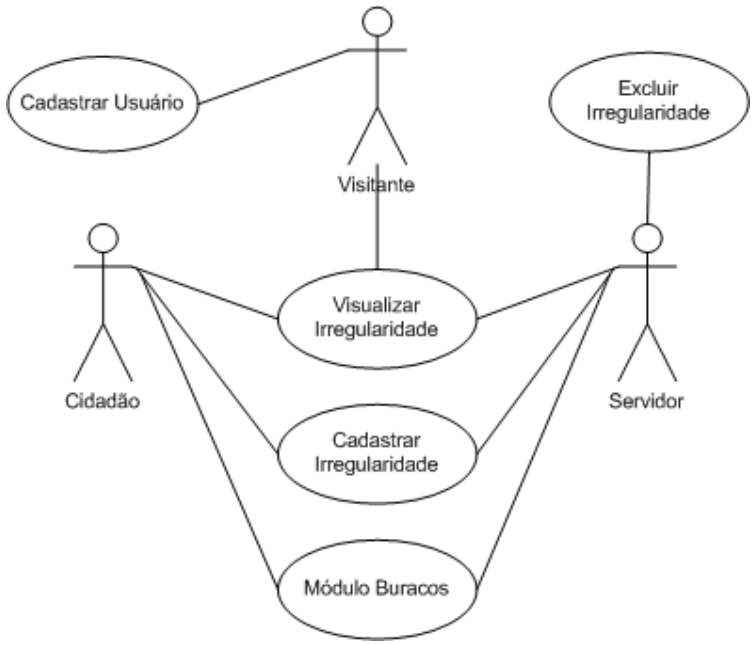


Figura 3.1 – Diagrama de Casos de Uso.

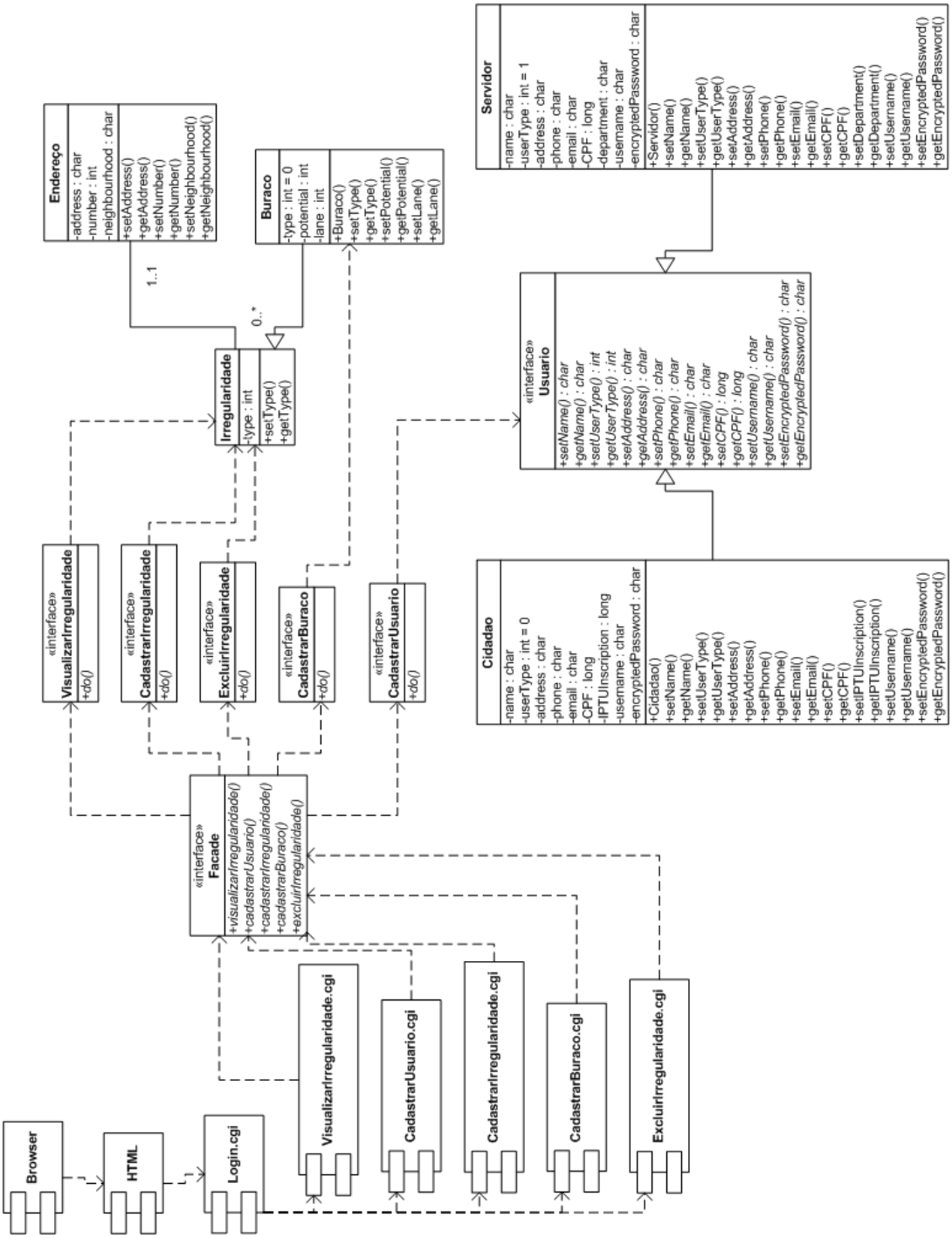


Figura 3.2 – Diagrama de Classes.

3.4 – Norma e Produto Final

Esta especificação segue a norma "ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE" constante da página online de Antônio de Sousa [11].
O produto final pode ser encontrado no apêndice B.

Capítulo 4

Descrição do Projeto de Software

4.1 – Atividades

Esta é a etapa do projeto imediatamente anterior ao desenvolvimento, em que se desenha a solução para o mesmo de acordo com a análise do problema.
A estrutura de trabalho básica foi montada e interligada para o módulo buraco de modo que sua expansão (adição de novos módulos e rotinas) se dê com o mínimo de esforço possível.
As classes referentes à adição de irregularidades foram desenhadas de forma que ficassem separadas umas das outras, permitindo que a adição de novos módulos impacte o mínimo possível os já existentes. As bibliotecas VBMCGI [9] e VBLib [8] permitem a programação orientada à web estritamente dividida em três camadas: webdesign, regras de negócio e banco de dados.
A seguir pode ser encontrado o detalhamento dos tópicos amplos da descrição do projeto.

4.2 – Decomposição

Na parte de decomposição é encontrada a divisão do sistema em módulos, em processos concorrentes e em dados. A decomposição em módulos pode ser visualizada a seguir.

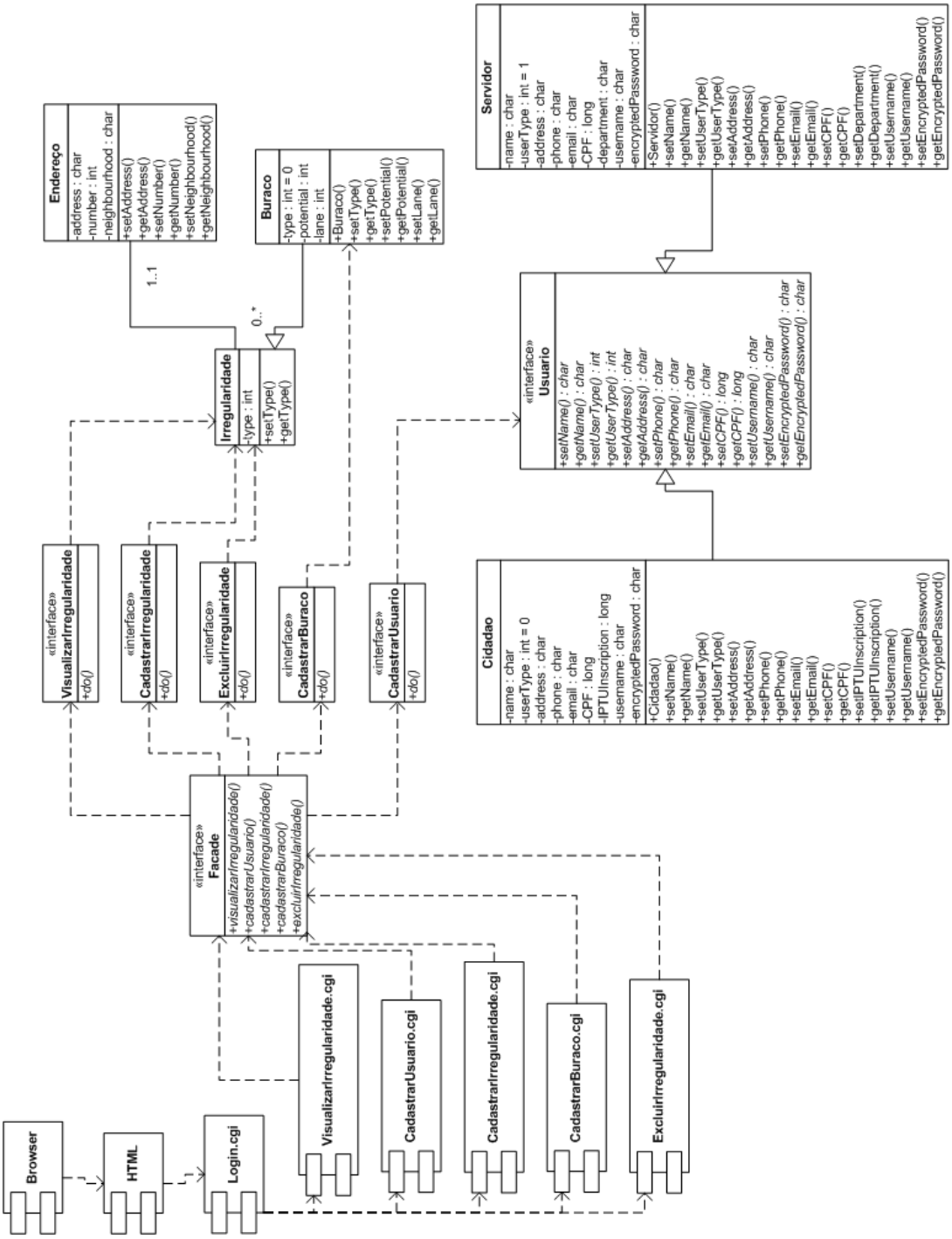


Figura 4.1 – Decomposição em módulos.

4.3 – Descrição das Dependências

São apresentadas as dependências entre módulos, processos e dados neste item. Para ilustrar melhor esta dependência, o diagrama de caso de uso Módulo Buraco pode ser visualizado a seguir.

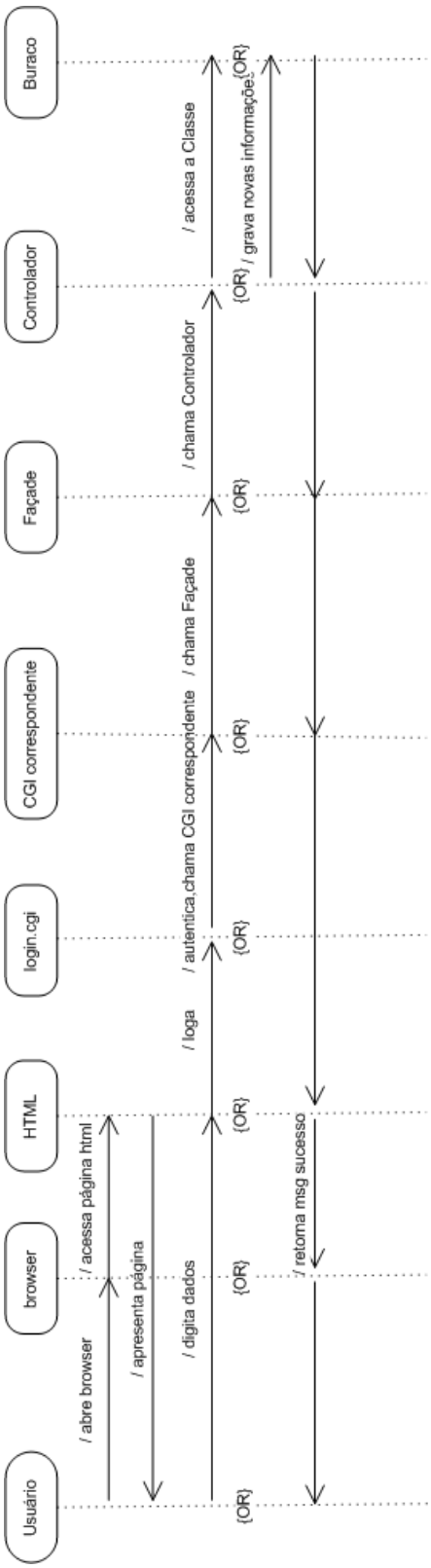


Figura 4.2 – Diagrama de Seqüência para o Caso de Uso Módulo Buraco.

4.4 – Norma e Produto Final

Esta descrição segue a norma "DESCRIÇÃO DE PROJETO DE SOFTWARE" constante da página online de Antônio de Sousa [12].
O produto final pode ser encontrado no apêndice C.

Capítulo 5

Desenvolvimento

5.1 – Ciclo de Vida

O projeto seguiu o ciclo de vida em espiral [3]. Isto é, a cada volta dada, é gerada uma nova versão. Foi gerada a primeira versão na primeira volta.

5.2 – Desenvolvimento das Páginas HTML

As páginas web foram desenvolvidas nas linguagens HTML e Javascript [6]. Na maioria das páginas, como cadastro de usuário e cadastro de buraco, foi usado HTML dinâmico para a mudança de elementos da página na ocorrência de determinados eventos sem a necessidade de carregá-la novamente.

Na página de cadastro de usuário, por exemplo, a descrição do campo Número de Inscrição de IPTU para o usuário é alterada para Departamento caso o tipo de usuário seja alterado de cidadão para servidor, e vice-versa, além de ser feita a verificação do CPF do usuário como forma de segurança. A página onde essa funcionalidade foi mais utilizada é a de cadastrar buraco, pois o endereço mostrado na tela, proveniente da API do Google Maps [4], tem de ser alterado a cada vez que o centro do mapa tem sua posição alterada ou a cada vez que o usuário clica no botão Pesquisar Endereço.

Nas páginas em que são listadas irregularidades na tela, foi usada uma string que é substituída pela regra de negócio com a VBMCgi [9], permitindo a separação das camadas entre o webdesign e as regras de negócio.

5.3 – Interface com a API do Google Maps

O grande diferencial do sistema é a integração com a API do Google Maps [4]. Desta maneira, os endereços cadastrados são padronizados e idênticos ao formato usado no banco de dados do site, não havendo a necessidade de um banco de dados de endereço local e evitando duplicatas e erros de digitação no endereço.

Assim, quando é feita uma busca na tela de visualizar irregularidade, ela é muito mais confiável. Isto é fundamental para a confecção de mapas de problemas por bairro, rua ou endereço.

Na tela de cadastrar buraco são feitas as solicitações à API chamadas de *Geocoding* e *Geocoding* reverso [4]. Se a busca for por endereço, há uma função Javascript [6] na página que concatena as informações digitadas exatamente no formato recomendado pelo Google, isto é: "Logradouro, número – bairro, cidade – estado, CEP, país". Neste formato, a chance de encontrar um endereço é muito maior. Após a concatenação, o serviço de *Geocoding* é solicitado à API, o mapa identifica a mudança e se centraliza na posição encontrada e as posições de latitude e longitude são passadas aos campos de formulário ocultos, os quais serão enviados junto com as outras informações ao se clicar no botão Ok. Finalmente, é requisitado um serviço de *Geocoding* reverso com a latitude e longitude encontrada e o endereço encontrado é escrito por extenso na tela, porém com uma faixa de numeração. Por este motivo, é usado o número digitado pelo usuário no campo correspondente no envio da informação de cadastro à CGI.

Se, por outro lado, o usuário estiver com dificuldades de encontrar o endereço ou preferir desta forma, o mapa pode ser centralizado na localidade da irregularidade. Da mesma forma, serão identificadas a latitude e a longitude do centro do mapa e será requisitado um *Geocoding* reverso para preencher o endereço encontrado por extenso na tela.

Na tela de visualizar irregularidades, esses serviços não são requisitados, pois o endereço correto e as coordenadas geográficas já estão cadastrados no banco de dados próprio do sistema. O procedimento é montar o mapa na tela adicionando marcadores naquelas coordenadas, e com o zoom e centro do mapa definidos pela regra de negócio (de acordo com a distância entre os marcadores) também repassados ao mapa para sua composição.

5.4 – Regras de Negócio

A classe que cadastra os buracos recebe, portanto, uma string que contém o endereço por extenso (desde o logradouro até o país) da localidade em que se encontra a irregularidade. O primeiro passo é desmembrar esta string usando um vetor de caracteres e laços de repetição condicionados. O logradouro e o bairro são, então,

gravados em variáveis distintas. O número a ser utilizado para o registro é o mesmo que foi digitado pelo usuário no campo número do formulário. Essas informações são, então, gravadas no banco de dados em colunas separadas na tabela Endereço, além das coordenadas geográficas, também em colunas separadas.

Os casos de uso de cadastro, tanto de usuário quanto irregularidade, testam se algum campo obrigatório foi deixado em branco e retornam uma mensagem de erro ao usuário.

Todos os casos de uso que requerem que o usuário esteja identificado no site testam, antes do acesso à tela correspondente, se o cookie presente em sua máquina realmente corresponde ao cadastrado no banco de dados e se o tipo de seu usuário tem privilégios suficientes para realizar aquela operação. Se o resultado for negativo, o usuário não consegue acessar as páginas, recebendo uma mensagem com a solicitação para se logar no site corretamente.

A tela de login retorna erros pertinentes, dizendo se o usuário é inexistente ou se a senha digitada é incorreta.

5.5 – Comunicação com o banco de dados

Alguns recursos interessantes foram utilizados na formação dos comandos no banco de dados, principalmente a concatenação de strings, a palavra-chave ILIKE do PostgreSQL [7] e a funcionalidade *RETURNING* dos enunciados *select*. Na tela de busca do caso de uso Visualizar Irregularidade é feito um teste para cada campo proveniente do formulário para saber se ele foi preenchido ou não (já que todos são opcionais), de forma a concatenar a string do comando select para o banco de dados. A string contendo o campo logradouro é montada com a palavra-chave ILIKE e colocando-se um sinal de porcentagem antes e depois da string proveniente do formulário da página. Desta forma, o banco de dados procura entradas que apenas contenham a string digitada no formulário, sem levar em consideração maiúsculas ou minúsculas e o tipo de logradouro (rua, avenida, etc.). Exemplo: um logradouro cadastrado como “R. Haddock Lobo” é localizado pela string “haddock lobo”. O enunciado do select neste caso seria: “SELECT * FROM address, hole WHERE hole.address = address.id AND address.address ILIKE '%haddock lobo%' ORDER BY hole.type,address.neighbourhood,address.address,address.number”. O campo bairro também usa ILIKE para não ser sensível a minúsculas ou minúsculas. Sem o uso deste

recurso, seria necessário digitar o endereço exato e com maiúsculas nas iniciais, tornando o sistema ineficiente.

Sobre a terceira funcionalidade citada, a palavra-chave *RETURNING* no enunciado do comando *select* é bastante útil nas tabelas de buraco e endereço. Quando uma irregularidade é cadastrada num endereço que ainda não existe, é necessária uma nova entrada na tabela de endereço e outra na de buraco, contendo uma referência àquela entrada. Usando-se a palavra-chave *RETURNING* no enunciado do *insert*, a id do endereço recém-cadastrado é retornada e pode ser usada para cadastrar a irregularidade na tabela correspondente. Exemplo: "INSERT INTO address (address,number,neighbourhood,latitude,longitude) VALUES (address,number,neighbourhood,latitude,longitude) RETURNING id" e depois "INSERT INTO hole (type,potential,lane,address) VALUES (type,potential,lane,id)". Ela também é usada ao se excluir uma irregularidade: é feita uma busca com o endereço da irregularidade recém-deletada, e, se não houver outras entradas com aquele endereço, ele também é excluído de sua tabela correspondente. Como dito, não há significado em existir um endereço que não possua nenhuma irregularidade associada.

Capítulo 6

Resultados

6.1 – Descrição dos Resultados

Foi desenvolvido um sistema que atende às especificações de requisitos e ao desenho da solução. A seguir, podem-se verificar telas referentes ao uso do programa:

Figura 6.1 – Tela Principal.

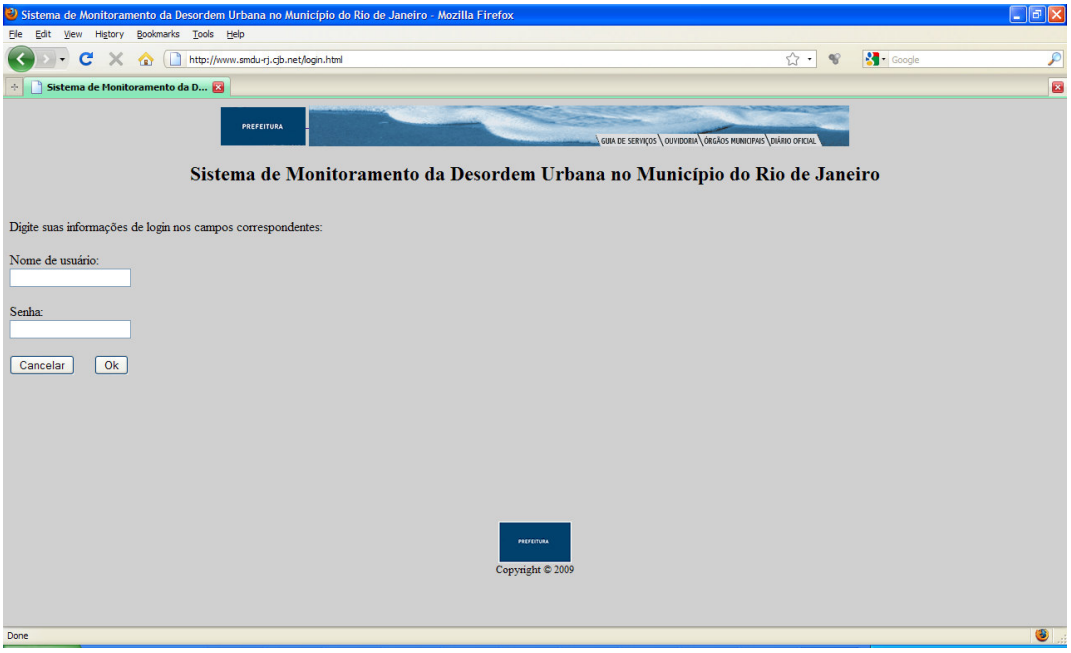


Figura 6.2 – Tela de Login.

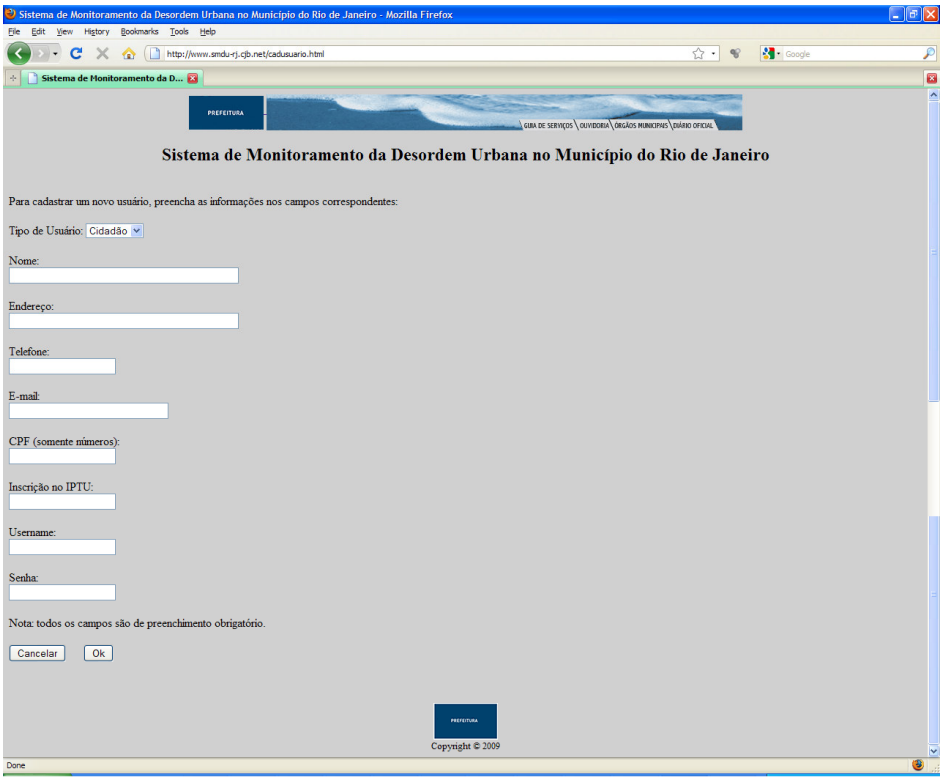


Figura 6.3 – Tela de Cadastro de Usuário.

Figura 6.4 – Tela de Busca por Irregularidades.

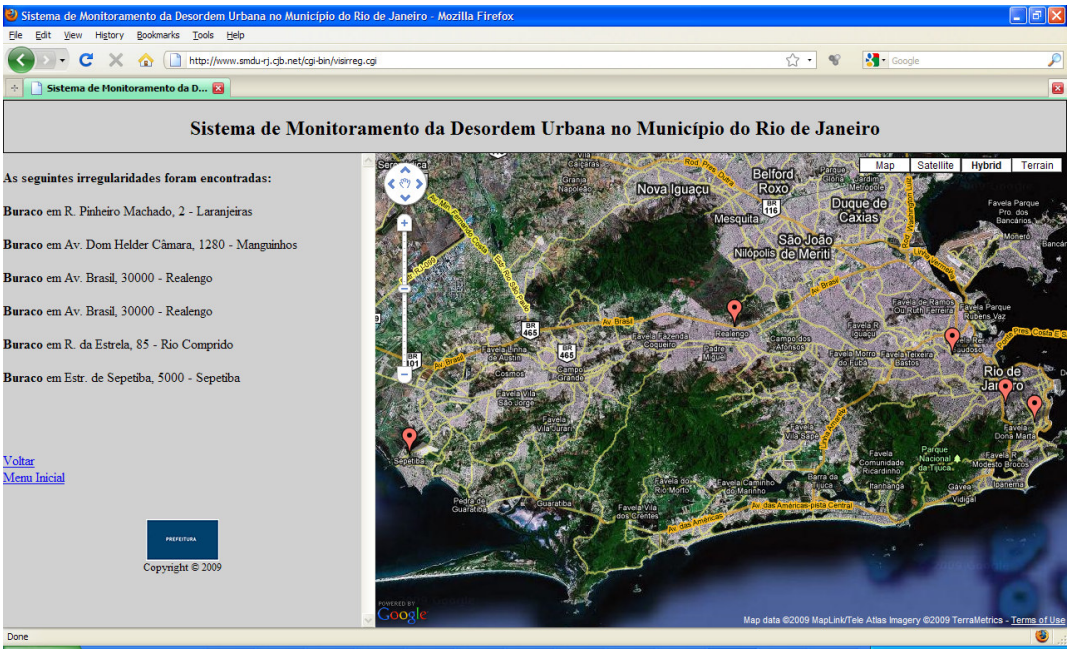


Figura 6.5 –Resultado da Busca por Irregularidades.

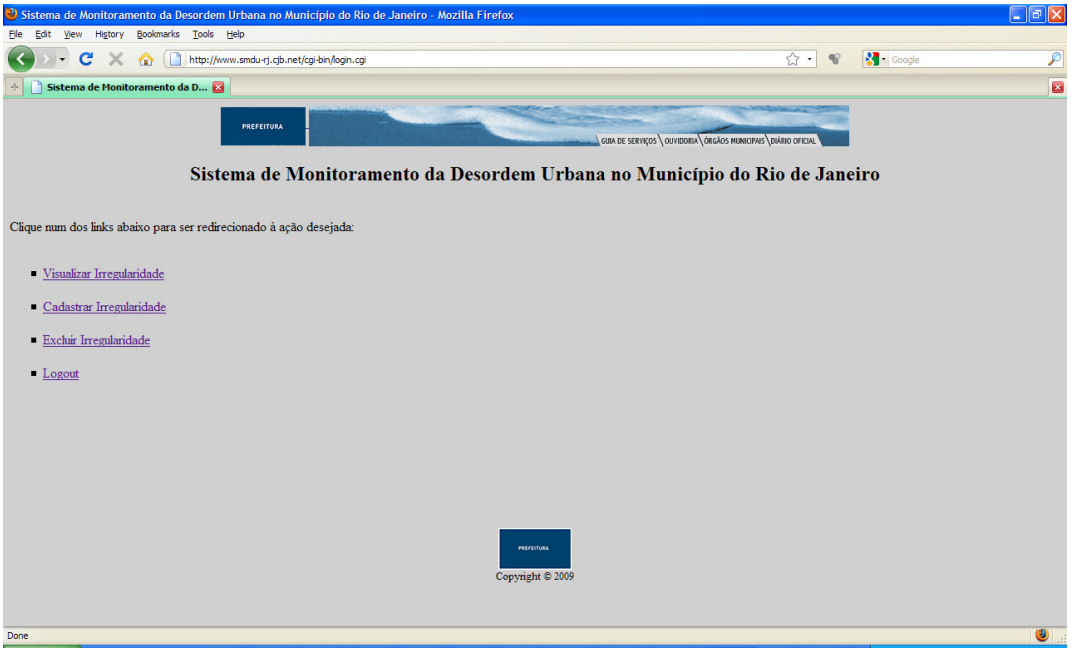


Figura 6.6 – Menu Inicial personalizado para Servidor.

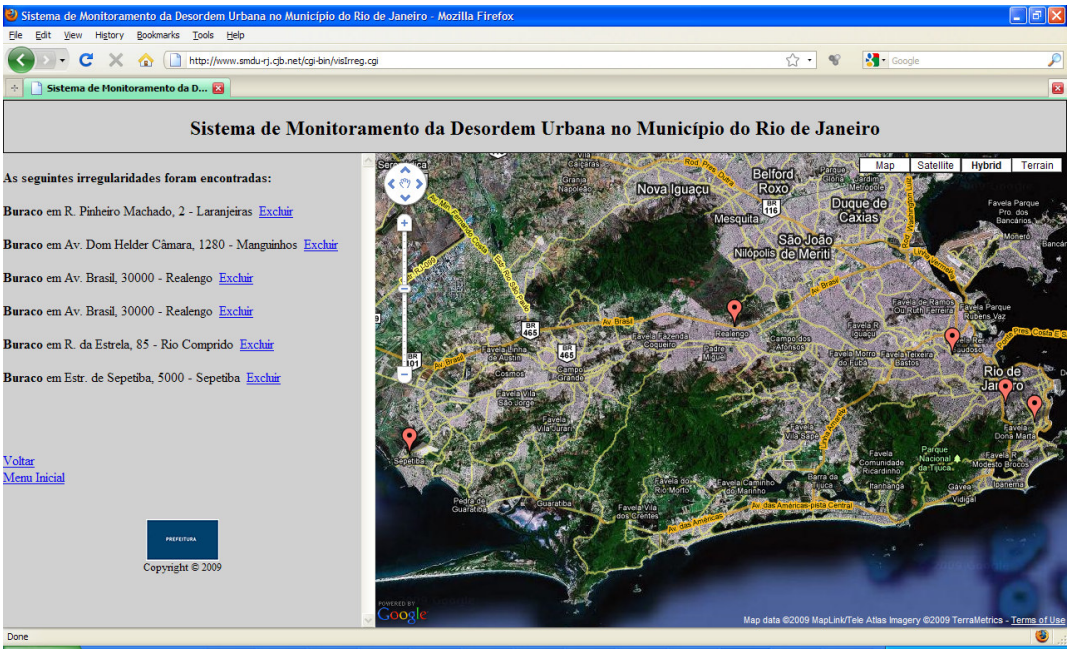


Figura 6.7 – Resultado da Busca para Excluir Irregularidades.

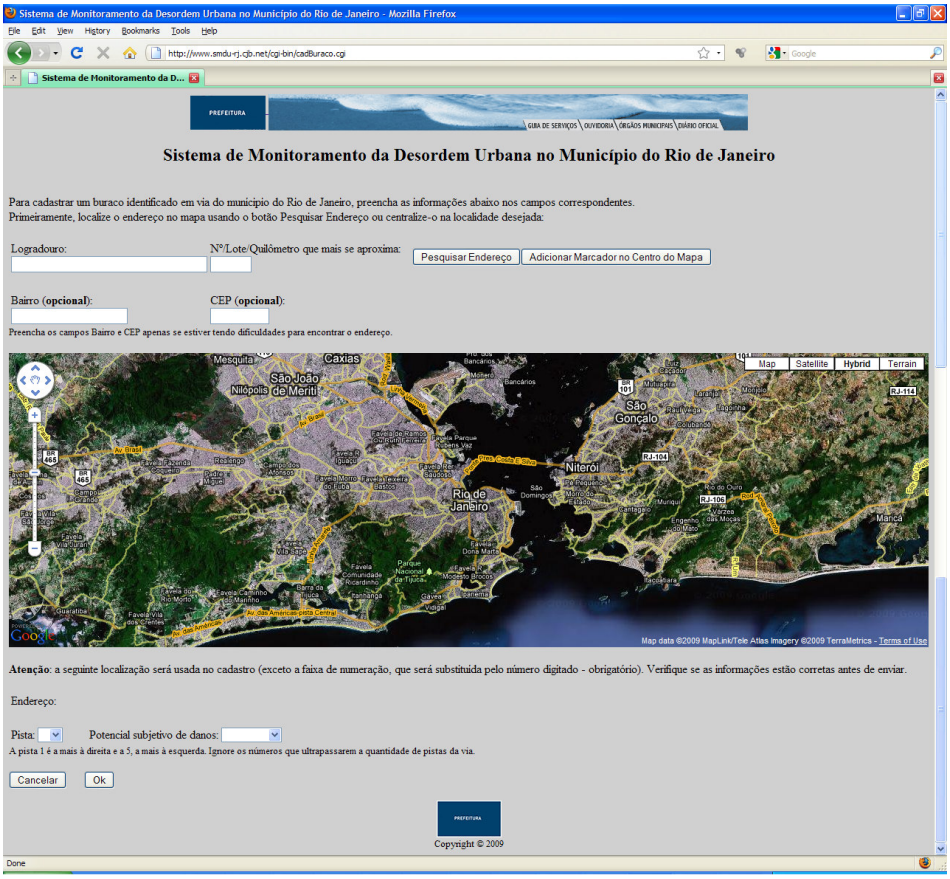


Figura 6.8 – Cadastrar Buraco.

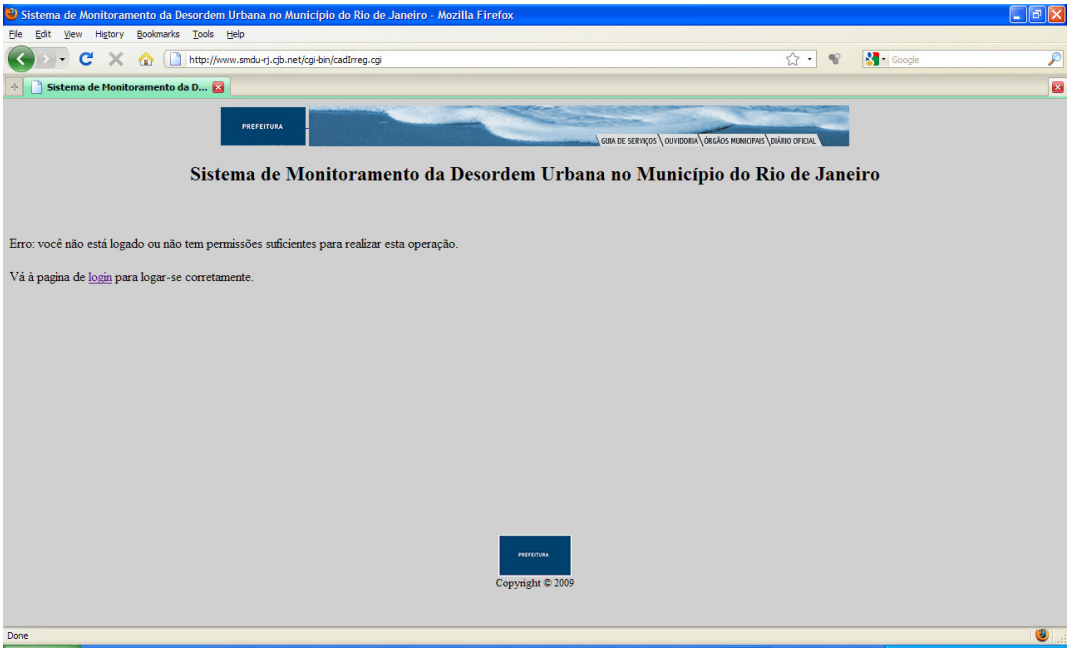


Figura 6.9 – Tentativa de Acesso sem estar logado ou sem privilégios suficientes.

6.2 – Testes

Não foram realizados testes formais, porém o sistema foi informalmente demonstrado a cidadãos, os quais testaram os casos de uso e comprovaram seu correto funcionamento.

Conclusão

O software foi desenvolvido dentro das expectativas de prazo, custo e qualidade. Portanto, de acordo com os resultados demonstrados comprovou-se que as técnicas enunciadas pelas normas citadas e utilizadas para o desenvolvimento e acompanhamento do projeto são realmente muito úteis e confiáveis, como imaginado previamente. Elas ajudam a nortear tanto as atividades quanto o detalhamento técnico do mesmo, mitigando riscos e insatisfações das diferentes partes interessadas.

Falando sobre o software especificamente, ele cumpriu os requisitos acordados no documento de Especificação de Requisitos de Software. A funcionalidade de executar rotinas frequentes analisando as irregularidades foi realmente adiada, além da construção de outros diferentes módulos, como previsto no projeto. A estrutura de trabalho básica está montada e interligada, e sua expansão (adição de novos módulos e rotinas) se dará com o mínimo de esforço possível.

Isto se deve às características provenientes do conceito de sustentabilidade (do inglês *maintainability*, outra tradução seria *manutenibilidade*) que foi levado em consideração desde o início do projeto. As classes são bem separadas, de modo a permitir que a adição de novos módulos impacte o mínimo possível os já existentes. A biblioteca VBMCGI [9] contribuiu sobremaneira para atingir este objetivo, pois permite a programação orientada à web estritamente dividida em três camadas: *webdesign*, regras de negócio e banco de dados. Desta forma, o profissional de cada área não tem a necessidade (nem o direito) de intervir na camada do outro, garantindo um maior controle sobre o software e, consequentemente, mais sustentabilidade.

Além destes conceitos, o software atendeu a mais algumas demandais atuais: colaborativo (web 2.0), *open-source, web-based* e a aplicação de *mash-ups* com a *API* do Google Maps [4], que é a mais debatida e estudada nos dias de hoje.

Sobre a característica *open-source*, quase todas as ferramentas utilizadas são gratuitas e, as que não são (como o Visio, para a confecção do projeto) podem ser facilmente substituídas. A linguagem de programação C++ [5], em conjunto com o banco de dados *PostgreSQL* [7] e as bibliotecas gratuitas VBLib [8] e VBMCGI [9] atenderam plenamente aos requisitos especificados, demonstrando a possibilidade de serem usadas em aplicações comerciais sem maiores restrições. Este aspecto, em minha opinião,

deveria ser mais utilizado principalmente em países em desenvolvimento, pois reduz os custos dos *royalties* enviados, quase sempre, aos países centrais e desenvolvidos. Isto traz como consequência o aumento da disparidade entre os países pobres e ricos e da dependência daqueles por estes.

Sobre a API do Google [4], foi usada a última versão, 3. Como é uma versão nova, não existia muito suporte disponível, o que fez com que mais tempo fosse necessário para o entendimento de seu funcionamento. Além desta questão, parece que houve a intenção de deixar esta versão mais leve que a anterior, a 2. Isto também teve como consequência mais tempo de desenvolvimento, pois ações que antes possuíam métodos para sua realização agora tinham de ser implementadas manualmente, como cálculo do zoom do mapa e as fronteiras mostradas na tela.

Referências Bibliográficas

[1] "*O espaço público na Cidade do Rio de Janeiro: civilidade ou barbárie?"* - Câmara de Vereadores do Rio de Janeiro. http://www.aspasiacamargo.com.br/docs/RELATORIOFINALCPIDESORDEMURBANA.pdf, 2009 (Acesso em 16 de Maio de 2009).

[2] "Object Management Group - UML", http://www.uml.org/, 2009 (Acesso em 05 de Agosto de 2009).

[3] PRESSMAN, Roger S. "*Software engineering: a practitioner's approach*". McGraw-Hill, 2001.

[4] "Google Maps API – Google Code", http://code.google.com/apis/maps/, 2009 (Acesso em 05 de Agosto de 2009).

[5] STROUSTRUP, Bjarne. "*The C++ Programming Language: Special Edition*". Addison-Wesley Professional, 2000.

[6] MUSCIANO, Chuck e KENNEDY, Bill. *"HTML & XHTML: The Definitive Guide"*. O'Reilly Media, 2005.

[7] MATTHEW, Neil e STONES, Richard. *"Beginning Databases with PostgreSQL: From Novice to Professional"*. Apress, 2005.

[8] VILLAS-BOAS, Sergio B. "VBLib The general purpose Villas-Boas (sbVB) library in C++ cross-platform (Windows and Unix)", http://www.sbvb.com.br/cgi-bin/index.cgi?p=14, 2009 (Acesso em 05 de Agosto de 2009).

[9] VILLAS-BOAS, Sergio B. "VBMcgi home page", http://www.vbmcgi.org/, 2007 (Acesso em 05 de Agosto de 2009).

[10] DE SOUSA, Antônio C. G. "NORMA: PLANO PARA O GERENCIAMENTO DE PROJETO DE SOFTWARE - PGPS", http://www.del.ufrj.br/~ac/normapla.rtf, 2009 (Acesso em 05 de Agosto de 2009).

[11] DE SOUSA, Antônio C. G. "NORMA: ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE", http://www.del.ufrj.br/~ac/normaers.rtf, 2006 (Acesso em 05 de Agosto de 2009).

[12] DE SOUSA, Antônio C. G. "NORMA: DESCRIÇÃO DE PROJETO DE SOFTWARE", http://www.del.ufrj.br/~ac/normapla.rtf, 2007 (Acesso em 05 de Agosto de 2009).

[13] "CSSE Website", http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html, 2009 (Acesso em 05 de Agosto de 2009).

[14] “Extreme Programming: A Gentle Introduction.”, http://www.extremeprogramming.org/, 2006 (Acesso em 05 de Agosto de 2009).

[15] PRICE, Robert P. “CVS - Open Source Version Control”, http://www.nongnu.org/cvs/, 2006 (Acesso em 05 de Agosto de 2009).

Apêndice A

Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro

PLANO PARA O GERENCIAMENTO DE PROJETO DE SOFTWARE

VERSÃO 1.0

9 de Julho de 2009

Organização:

Romulo Vanzillotta Castello

_____

Antônio Cláudio Gómez de Sousa
Responsável pela aprovação do Plano

Prefácio

O escopo deste planejamento é definir regras claras para o projeto deste software, a fim de mitigar dúvidas e riscos e possibilitar um acompanhamento mais estreito do mesmo. O contexto é o Projeto Final no Departamento de Eletrônica e Computação da Universidade Federal do Rio de Janeiro. Os possíveis interessados são os componentes da banca de análise do Projeto (Prof. Antônio Cláudio Gómez de Sousa, orientador, e Prof. Sérgio Barbosa Villas Boas e Prof. Aloysio de Castro Pinto Pedroza, examinadores) e os futuros usuários, ou seja, cidadãos e servidores da administração do município do Rio de Janeiro.

Sumário

Lista de Figuras

Lista de Tabelas

1. Apresentação

1.1 Sumário do Projeto

1.1.1 Finalidades, Escopo e Objetivos

A finalidade deste projeto é servir como cadastro de todo tipo de irregularidades pertinentes à boa ordem do município do Rio de Janeiro, tanto para a prefeitura quando para os cidadãos, facilitando assim a tomada de decisão pela prefeitura e acompanhamento das não-conformidades existentes (buracos na rua, etc.). O escopo de trabalho do projeto é a criação de um sistema colaborativo *web-based*, onde os cidadãos e servidores possam cadastrar irregularidades de variados tipos. O objetivo do projeto é possibilitar o acesso a estas informações de forma segura e em tempo real.

1.1.2 Postulados e Restrições

A principal restrição é o prazo, definido pela próxima colação de grau. Outras restrições são: confiabilidade do sistema e interoperabilidade.

1.1.3 Liberações Parciais

Não estão previstos produtos parciais.

1.1.4 Sumário de Cronograma e Orçamento

Segue, abaixo, o cronograma macro do projeto:

Fase 1: Análise do problema
Fase 2: Desenho da solução
Fase 3: Instalação dos programas e configuração do ambiente
Fase 4: Desenvolvimento do módulo de identificação
Fase 5: Desenvolvimento dos módulos de inserção
Fase 6: Desenvolvimento das ferramentas de relatório
Fase 7: Integração com o website de mapas e hospedagem do software online

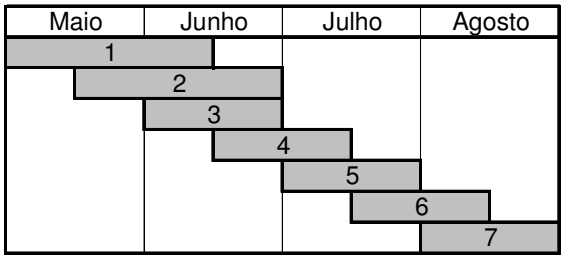


Figura 1.1 – Cronograma.

O detalhamento de cada item do cronograma é apresentado no item 5.2.2 deste documento.

O orçamento, com um analista/programador ao salário de R$ 6.500,00 em 4 meses e dedicando cerca de duas horas por dia (um quarto do expediente normal), ficará em R$ 6.500,00. Maiores detalhes no item 5.2.4.

1.2 Evolução do Plano

As mudanças no plano serão feitas ao longo do projeto, conforme forem sendo verificas sua necessidade. Os validadores serão avisados de cada nova versão.

2 Referências

Para o planejamento, será utilizada uma série de ferramentas e normas. Guiar-se-á pelas normas constantes da página online de Antônio de Sousa . Documentos que porventura se façam necessários serão mencionados posteriormente.

O ambiente de desenvolvimento é o Windows XP, linguagem de programação, C++ [5] e programa desenvolvedor, Visual Studio C++ 6.0 . Serão usadas as bibliotecas do professor Sérgio Barbosa Villas Bôas, principalmente as de âmbito geral e CGI. Banco de dados *open-source* PostgreSQL [7], para o armazenamento dos dados. Para a interface entre o *front-end* e o *back-end*, biblioteca SQLAPI.

Como o sistema é *web-based*, no ambiente de navegação estão inclusos um editor de HTML [6] (Bloco de Notas), e o navegador de Internet Firefox. Para a hospedagem das páginas e execução das CGIs, outro programa *open-source*, o Apache.

Procurou-se usar ao máximo esse tipo de programa para a redução dos custos do projeto.

Será usado o Microsoft Visio como ferramenta UML [2] para o desenho dos casos de uso, etc.

3 Definições

SMDU-RJ: Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro, nome do software;

UML: Unified Modeling Language, acronismo inglês para Linguagem de Modelagem Unificada. Ela é não-proprietária e de terceira geração, o que significa que é aberta, sem custos e auxilia a visualizar a comunicação entre os objetos como um todo;

CGI: Common Gateway Interface, ou interface de passagem comum. Mecanismo de acesso através dos provedores http para programação e registros de informações na Internet;

HTML: HyperText Markup Language ou linguagem HTML. Código para veicular textos e imagens na Internet.

4 Organização do Projeto

4.1 Interfaces Externas

A fronteira organizacional do projeto serão os usuários, cidadãos do município, que definirão os requisitos, meios mais amigáveis de uso, etc.

4.2 Estrutura Interna

Como o projeto contém apenas um membro, o Engenheiro de Software, sua estrutura organizacional é individual. Porém, constantemente será demonstrado a cidadãos com o intuito de avaliar sua usabilidade.

4.3 Papéis e Responsabilidades

O Engenheiro de Software responsável pela configuração do ambiente na máquina desenvolvedora, análise dos requisitos e desenvolvimento do sistema.

5 Processos de Gerenciamento

5.1 Partida no Projeto

5.1.1 Previsões

Para o estabelecimento dos parâmetros estatísticos relativos à parte técnica, como o número de linhas de código, Homem x mês, orçamento, etc, utilizou-se o software Cocomo [13].

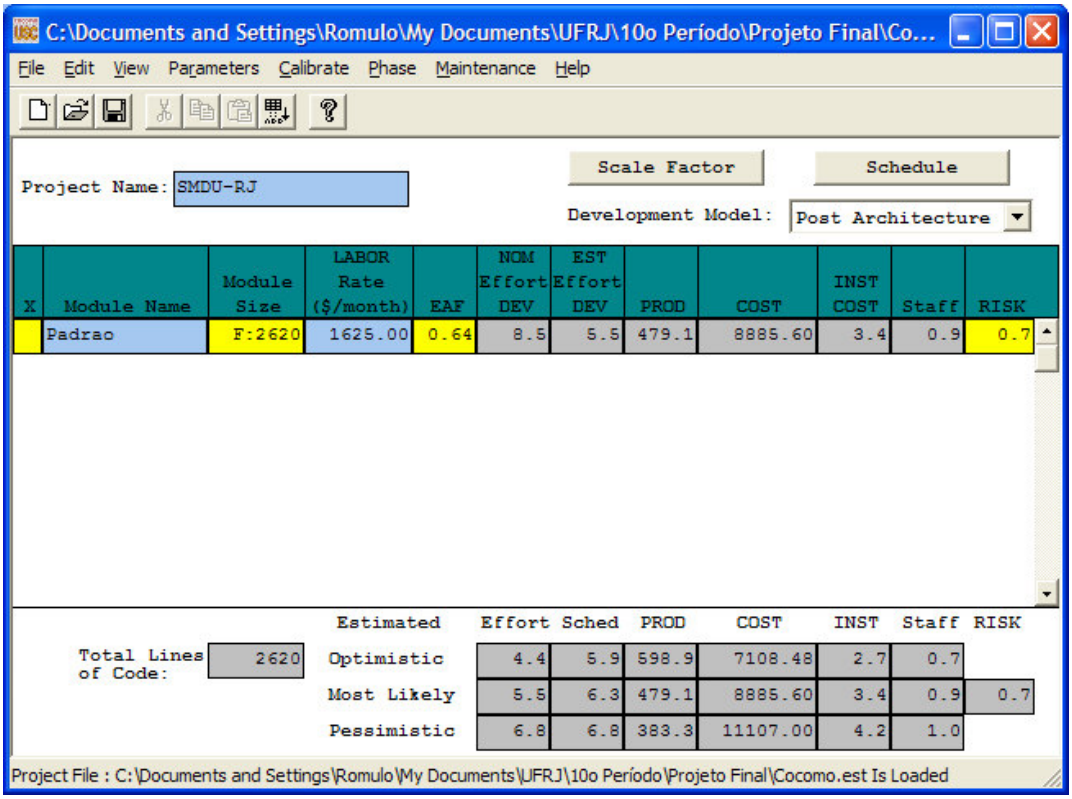


Figura 5.1 – Tela Principal Cocomo.

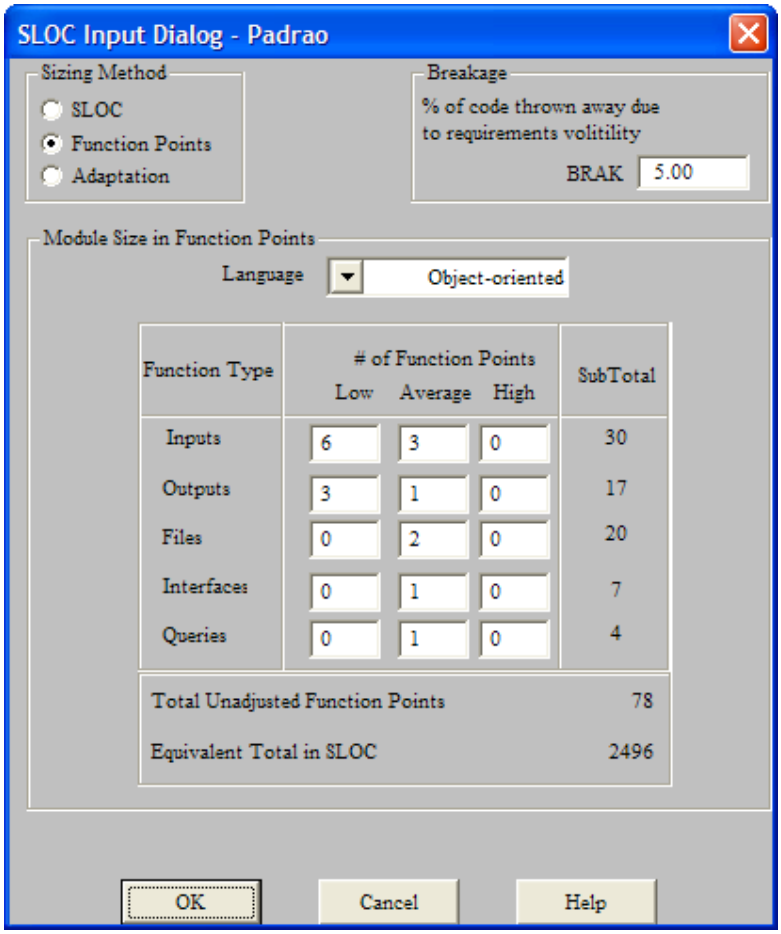


Figura 5.2 – Function Points.

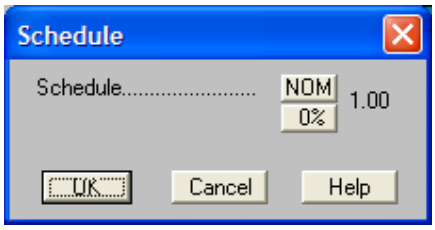


Figura 5.3 – Schedule.

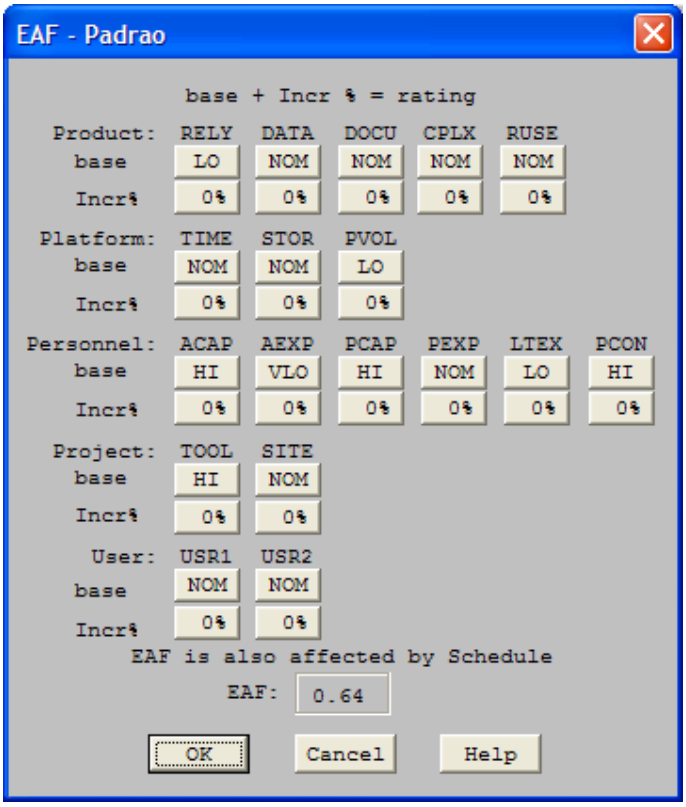


Figura 5.4 – EAF.

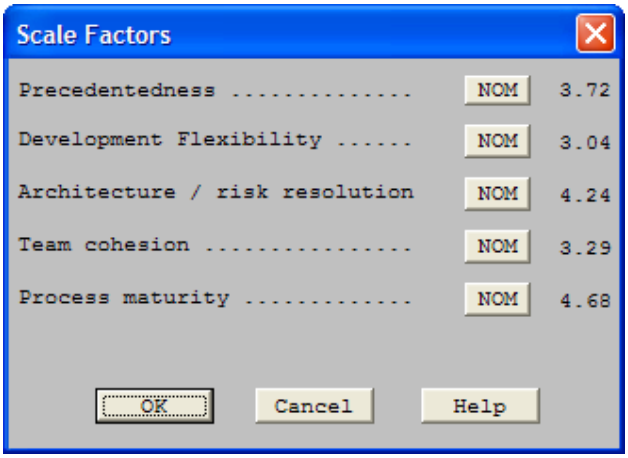


Figura 5.5 – Scale Factors.

5.1.2 Equipe

O projeto será desenvolvido exclusivamente pelo Engenheiro de Software, incluindo: seu gerenciamento, validação do andamento do projeto conforme o planejado, confecção do controle de riscos, análise da qualidade de implementação do sistema, integração dos interesses dos futuros usuários com o desenvolvimento do software. Além disto, será responsável também pela codificação na linguagem de programação, implementando a lógica das funcionalidades de todo o sistema.

5.1.3 Plano para a Aquisição de Recursos

Não se aplica a este projeto.

5.1.4 Plano de Treinamento da Equipe

Não se aplica a este projeto.

5.2 Plano de Trabalho

5.2.1 Atividades

Como a equipe responsável pela implementação do projeto é individual e o prazo para implementação também é pequeno, o risco do projeto é consideravelmente alto. Vale lembrar que o desenvolvimento do mesmo é plausível devido à experiência prévia em desenvolvimento de projetos desta natureza

5.2.2 Prazos

Segue o cronograma macro:

Fase 1: Análise do problema
Fase 2: Desenho da solução
Fase 3: Instalação dos programas e configuração do ambiente
Fase 4: Desenvolvimento do módulo de identificação
Fase 5: Desenvolvimento dos módulos de inserção
Fase 6: Desenvolvimento das ferramentas de relatório

Fase 7: Integração com o website de mapas e hospedagem do software online

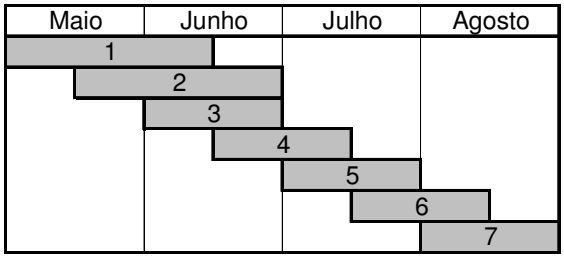


Figura 5.6 – Cronograma.

Na fase 1, análise do problema, será definido como será o nosso projeto do ponto de vista de software, incluindo a análise dos casos de uso, diagrama de classes e atividades, para que depois seja possível escolher as ferramentas mais adequadas para implementação do sistema. O produto desta fase será a Especificação de Requisitos do Software.

Para a fase 2, desenho da solução, serão identificadas todas as classes necessárias para o projeto. O resultado desta fase será a Descrição do Projeto de Software.

Na fase 3, será dado início à implementação da solução com a configuração do ambiente de desenvolvimento necessário na máquina correspondente, com a instalação de todos os programas e interfaces.

As fases 4, 5 e 6 representam uma divisão dos trabalhos. Isto poderá ser mudado de acordo com a finalização do desenho da solução.

Por último, a fase 7 representa a finalização do projeto, que inclui a integração com o website de mapas e a hospedagem do software online. Isto também pode sofrer alterações, já que a integração com o website de mapas pode se fazer necessária durante o desenvolvimento das fases anteriores.

5.2.3 Alocação de Recursos

Como este projeto prevê apenas a alocação de recursos humanos, este item pode ser verificado no item 5.1.2.

5.2.4 Alocação de Orçamento

O orçamento calculado para a realização desse projeto está ligado diretamente à mão-de-obra, ou seja, um analista/programador, uma vez que todos os softwares utilizados para o desenvolvimento do produto são freeware. Como o responsável tem experiência na implementação de sistemas como este, seu custo foi calculado em R$ 6.500,00 ao mês, o que dá um custo total para implementação de R$ 8.125,00 (prazo de para implementação de 4 meses mais um mês de contingência), dedicando-se cerca de duas horas por dia (um quarto do expediente normal). Adicionando uma contingência de 10% neste valor para quaisquer imprevistos, o valor total fica em R$ 8.937,50.

5.3 Plano de Trabalho

5.3.1 Controle de Requisitos

Especificamente para este projeto, mudanças de requisito na visualização dos dados na tela não irão impactar tanto na implementação do projeto, exceto o armazenamento dos dados. Ou seja, quais informações serão preenchidas, qual tipo destas informações e o que seria reportado de relevante para a consulta destas novas informações. Logo, qualquer mudança de requisito poderá ser facilmente implementada sem que haja uma perda considerável do que já foi feito, e consequentemente poderá haver uma readaptação com o novo escopo. Uma mudança de requisito que represente de fato uma mudança de escopo acarretará em um novo prazo e consequentemente um novo custo.

5.3.2 Controle de Prazos

Como foi descrito no item anterior, uma mudança de requisito não impactará tanto na implementação do projeto, exceto se houver a inclusão de novos dados. Caso isto venha a acontecer, será necessário fazer as devidas alterações no cronograma, levando em consideração a viabilidade das possíveis mudanças e como ficaria o novo planejamento sem que houvesse um impacto no prazo. Após as mudanças no cronograma, seria feita uma nova análise do caminho crítico dando as diretrizes das prioridades, a fim de cumprir o planejado.

5.3.3 Controle de Orçamento

Como o prazo de implementação é invariável e os custos estão diretamente ligados ao custo de mão-de-obra, o orçamento é, por consequência, também invariável.

5.3.4 Controle de Qualidade

O controle de qualidade se baseará no que foi previsto no projeto. Assim, se todos os processos forem executados dentro do previsto, ou seja, se todas as atividades definidas no cronograma e na especificação dos requisitos forem realizadas como foi previsto inicialmente, será considerado de boa qualidade. A qualidade do produto será medida através da proposta do mesmo, ou seja, se ele responde a todas as especificações para as quais foi designado e se se mantém dentro da interface proposta, o produto será de boa qualidade. Serão feitas revisões técnicas sobre os artefatos gerados durante o desenvolvimento do sistema.

5.3.5 Plano de Relatórios

Caso haja alguma mudança de escopo devido a especificações inicialmente avaliadas como inviáveis de se implementarem, será realizado um relatório descrevendo e justificando a não realização da especificação, assim como uma nova proposta de alteração dos requisitos. Este relatório será enviado ao responsável pela aprovação deste documento. Confirmada a alteração pelo responsável da aprovação, será gerado um novo documento de planejamento com as mudanças aprovadas, atualizando, assim, a proposta de planejamento.

Os bugs encontrados no projeto a partir dos testes informais serão relatados por email também ao validador, informando em que local e que tipo de erro ocorreu. Após a correção do erro, uma resposta ao email inicial será enviada, notificando sua correção e como o mesmo foi solucionado.

5.4 Plano de Gerenciamento de Riscos

Os riscos selecionados são apresentados abaixo.

Tabela 5.1 – Gerenciamento de Riscos.

5.5 Plano de Encerramento

A partir do planejamento, espera-se cumprir os prazos e metas estabelecidos, entregando os protótipos, documentos e o produto nas datas previstas, satisfazendo as características desejadas pelo usuário e alcançando a qualidade e satisfação esperadas pelo mesmo.

6 Processos Técnicos

6.1 Modelo dos Processos

A primeira etapa será a análise do problema, seguida pelo desenho da solução. Como início da implementação da solução, será feita a configuração dos ambientes na máquina desenvolvedora.

Posteriormente, será iniciada a codificação do desenvolvimento, dividida nas fases 4, 5 e 6. A última fase, 7, finalizará o projeto com a integração ao website de mapas e a hospedagem online do software. O modelo de ciclo de vida será em espiral [3] e os marcos e paralelismos de cada fase podem ser encontrados no cronograma detalhado, item 5.2.2. A cada volta do espiral será produzida uma versão do software.

6.2 Métodos, Ferramentas e Técnicas

Os conceitos da metodologia de desenvolvimento Extreme Programming (XP) [14] serão aprendidos e aplicados à medida do possível, e a linguagem de programação será C++ [5]. A ferramenta será o editor de HTML Bloco de Notas. Ambiente Windows.

6.3 Infra-estrutura

Ambiente de desenvolvimento Visual Studio. Bibliotecas SQLAPI, para a interface entre o *front-end* e o *back-end*, VBLib [8] e VBMCgi [9] , respectivamente para âmbito geral e programação de CGIs.

Normas constantes da página *online* do professor Antônio Cláudio Gómez de Sousa, http://www.del.ufrj.br/~ac/eel873conteudo.htm#normas . Documentos que porventura se façam necessários serão mencionados posteriormente.

Os recursos básicos necessários ao projeto, como hardware, internet, eletricidade, já se encontram nos local físico de desenvolvimento.

6.4 Plano para a Aceitação do Produto

Como a interface com o usuário final é web-based, acredita-se que não haverá maiores problemas na aceitação do produto por parte dos mesmos.

Como forma de divulgação, porém, serão realizadas apresentações com *datashows* para os grupos de usuários *in loco*.

7 Planos para os processos de Suporte

7.1 Gerenciamento de Configuração

Será usado o programa CVS [15] para identificar os itens de software, o estado e a liberação de versões e processos para controlar as mudanças em requisitos.

7.2 Plano de Verificação e de Validação

O plano de verificação inclui revisão de marcos, revisão do progresso e modelagem.

Já o plano de validação inclui técnicas para testar, demonstrar, analisar e inspecionar. Será mantido um canal estreito com os usuários para o cumprimento destas etapas.

7.3 Documentação

A documentação será realizada previamente à codificação do sistema, e atualizada à medida que o mesmo for desenvolvido.

7.4 Plano para Assegurar a Qualidade

O plano para assegurar a qualidade prevê a verificação e validação, já citadas anteriormente, além de revisões e auditorias constantes ao longo do processo ao fim de cada fase.

7.5 Revisões e Auditorias

As revisões e auditorias acontecerão ao longo do processo e pela verificação de concordância com os requisitos estipulados, estreitamente com o validador e com os futuros usuários.

7.6 Plano para a Resolução de Problemas

Se o desenvolvimento começar a destoar muito dos prazos estabelecidos no programa Microsoft Project, identificar possíveis oportunidades para a alteração dos requisitos (diminuição).

7.7 Gerenciamento de Sub-contratações

Não estão previstas sub-contratações para este projeto.

7.8 Plano de Aperfeiçoamento

Para fins de aperfeiçoamento do projeto, 20% do tempo serão dedicados a esta tarefa. Ou seja, em 80% do tempo, o programador/analista desenvolverá suas atividades corriqueiras, e nos outros 20%, pensará em como melhorá-las.

Para o aperfeiçoamento, serão realizadas periodicamente reuniões com o validador e futuros usuários para a troca de idéias e conhecimentos acerca do assunto do projeto.

Apêndice B

Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro

ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE

VERSÃO 1.0

19 de Julho de 2009

Organização:

Romulo Vanzillotta Castello

Antônio Cláudio Gómez de Sousa
Responsável pela aprovação do Plano

Sumário

1 Introdução

1.1 Finalidade

A finalidade desta especificação é definir claramente os requisitos do software em questão para a minimização das insatisfações por parte do usuário e erros de implementação. Destina-se aos analistas e programadores responsáveis pelo sistema.

1.2 Escopo

O Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro – SMDU-RJ – tem como objetivo geral propor um sistema capaz de servir como base de dados das irregularidades existentes na cidade, tanto aos cidadãos como aos funcionários da prefeitura, de forma prática e usável. Aos funcionários, se prestaria a guiar seu trabalho de ações corretivas e preventivas, e aos cidadãos se prestaria como um alerta enquanto as situações não fossem resolvidas, além de um canal direto com a prefeitura. Desta forma, pretende-se conter e eliminar a crescente desordem presente na cidade do Rio de Janeiro. O escopo de trabalho do projeto é a criação de um ambiente via Internet, onde os usuários possam cadastrar dados referentes aos problemas percebidos. Ou seja, possibilitar o registro e acesso a estas informações de forma confiável e online.

1.3 Definições, Acrônimos e Abreviaturas

SMDU-RJ: Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro, nome do software;

UML: Unified Modeling Language, acronismo inglês para Linguagem de Modelagem Unificada. Ela é não-proprietária e de terceira geração, o que significa que é aberta, sem custos e auxilia a visualizar a comunicação entre os objetos como um todo;

CGI: Common Gateway Interface, ou interface de passagem comum. Mecanismo de acesso através dos provedores http para programação e registros de informações na Internet;

HTML: HyperText Markup Language ou linguagem HTML. Código para veicular textos e imagens na Internet.

1.4 Referências

Os documentos referenciados são:

- NORMA: ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE [11];
- PLANO PARA O GERENCIAMENTO DE PROJETO DE SOFTWARE - Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro versão 1.0. Data: 09/07/2009; responsável: Romulo Vanzillotta Castello.

1.5 Resumo

Como prosseguimento desta especificação, será dada uma descrição geral do produto. Além disso, serão listados os requisitos específicos, que incluem, principalmente, as interfaces externas: dos usuários, de hardware, de software e de comunicação. Também fazem parte dos requisitos específicos: os funcionais, de desempenho, as restrições do projeto e atributos.

2 Descrição Geral

2.1 Perspectiva do Produto

O produto é dependente apenas de um navegador de Internet para sua utilização, o qual é gratuito e já está instalado em praticamente todos os computadores. Deste modo, o sistema pode ser acessado em qualquer local do município com apenas uma conexão de Internet e um computador.

2.2 Funções do Produto

A função do produto é dar aos seus usuários a oportunidade de registrar e acessar diversos tipos de irregularidades existentes no município do Rio de Janeiro, incluindo cidadãos e servidores da administração municipal.

2.3 Características do Usuário

Os usuários do produto incluem, portanto, os munícipes e servidores públicos daquela cidade. Supõe-se uma faixa etária de 20 a 65 anos, e, apesar de ser acessível a todos, espera-se que o usuário tenha um grau de cultura de médio a elevado devido à necessária noção de coletividade e praticidade para usar um sistema de ouvidoria online, eventualmente tendo um grau baixo de cultura. Seu uso será intermitente pelos cidadãos e permanente pelos servidores que trabalham com a avaliação das reclamações. Portanto, a amigabilidade é um requisito indispensável para convencer o cidadão a usar o sistema, preterindo outros canais mais custosos, arcaicos e com menor publicidade dos problemas.

2.4 Restrições

A principal restrição é a segurança do sistema, que só pode ter novos itens cadastrados por pessoas acreditadas, sejam cidadãos ou servidores. Deverá ser feita imediatamente uma avaliação estrita quando do cadastramento de um novo usuário, realizando-se uma interface com o banco de dados da prefeitura e checando itens como o número de inscrição do IPTU, CPF, endereço, etc. Do contrário, poderia haver a ocorrência de itens que não correspondem à realidade, com o simples intuito de brincadeiras, trotes, ou mesmo criar situações

fantasiosas para a desmoralização da administração pública, da cidade propriamente dita, de algum bairro específico ou do entorno da casa de alguma pessoa.

2.5 Pressupostos e Dependências

O principal pressuposto corresponde ao que já foi descrito acima, o requisito de alta segurança do sistema. A dependência seria da disponibilidade dos dados da prefeitura para a verificação dos dados do cidadão na ocasião do cadastro de um novo usuário.

2.6 Postergar Requisitos

O requisito que pode ser postergardo é o de geração de relatórios frequentes para informar os servidores de tendências irregulares num mesmo local, as quais requerem mudanças. Como exemplos, pode-se citar a alta incidência de buracos numa mesma via, a qual requer a troca do piso asfáltico, ou uma série de invasões numa mesma área e num curto espaço de tempo, denotando a criação (ou expansão) de uma nova favela.

Outro requisito é a questão dos módulos: existe uma quantidade muito grande de tipos de irregularidades, cada um implementado por um módulo diferente. À medida que futuras versões forem lançadas, novos módulos podem ser adicionados ao software.

3 Requisitos Específicos

3.1 Interfaces Externas

3.1.1 Interfaces dos Usuários

A interface com o usuário será *web-based*, como dito anteriormente. O acesso será simples a ponto de não requerer uma versão específica de navegador, sendo bastante uma versão razoavelmente atual (menos de 5 anos de lançamento).

3.1.2 Interfaces de Hardware

Não haverá qualquer interface com o hardware.

3.1.3 Interfaces de Software

Este software deverá ser integrado ao banco de dados da prefeitura que contém informações sobre os munícipes. Haverá interface também com o navegador de Internet.

3.1.4 Interfaces de Comunicação

O software usará a Internet como rede de comunicação principal, e posteriormente a rede interna da prefeitura para verificação dos dados cadastrais.

3.2 Requisitos Funcionais

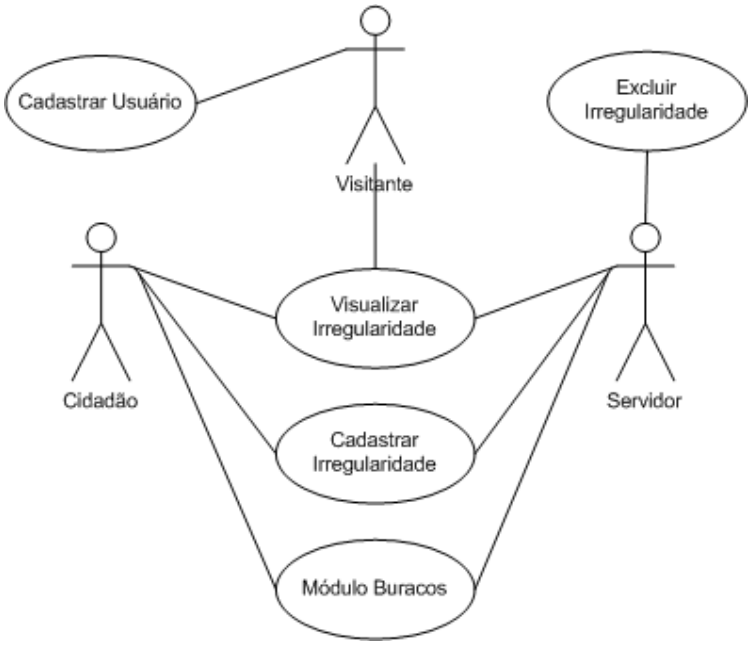
Este item diz respeito à análise orientada a objeto, utilizada neste projeto.

- Diagrama de Casos de Uso

  A figura abaixo representa o diagrama de casos de uso. Existem 3 tipos de atores no sistema e 5 eventos principais.

  Atores: Visitante, Cidadão e Servidor

  Eventos: Visualizar Irregularidade, Cadastrar Usuário, Cadastrar Irregularidade, Módulo Buracos, Excluir Irregularidade



❖ Especificação dos Casos de Uso

Apresentar-se-á a seguir o detalhamento de cada um dos casos de uso do sistema, ou seja, uma descrição de cada ação praticada pelos atores e da resposta do sistema a esta.

✓ Visualizar Irregularidade

Atores: Visitante, Cidadão e Servidor;

Caso de uso relacionado: Nenhum

Pré-condições: Nenhuma;

Pós-condições: Mostrar a irregularidade no mapa e sua descrição

Fluxo Principal

(1) O sistema apresenta a tela de bem-vindo e os campos de busca para a visualização de irregularidades
(2) O ator entra com as informações que deseja buscar
(3) O sistema apresenta os resultados, já localizados no mapa, e aguarda o ator clicar em algum
(4) Se o ator clicar em algum, só ele aparece no mapa, e sua descrição aparece ao lado esquerdo da tela

- ✓ Cadastrar Usuário

Ator: Visitante;

Caso de uso relacionado: Visualizar Irregularidade;

Pré-condições: Nenhuma;

Pós-condições: Novo usuário cadastrado no sistema;

Fluxo principal:

(1) O ator clica no link de cadastramento na tela de boas vindas ou após visualizar irregularidades
(2) O sistema apresenta a tela de cadastro de usuário para o ator
(3) O ator entra com suas informações: tipo de usuário, nome, endereço, telefone, e-mail, CPF, IPTU ou departamento (dependendo do ator), nome de usuário e senha inicial
(4) O sistema comunica o sucesso na adição do usuário

- ✓ Cadastrar Irregularidade

Atores: Cidadão e Servidor

Caso de uso relacionado: Cadastrar Usuário

Pré-condições: Estar identificado no sistema;

Pós-condições: Redirecionamento à página de inserção de dados do módulo correspondente

Fluxo principal

(1) O ator clica no link de cadastrar irregularidade na tela de sucesso de identificação de usuário ou de cadastramento de usuário
(2) O sistema apresenta a lista de módulos implementados
(3) O ator seleciona o módulo desejado e é redirecionado à tela de inserção de dados correspondente

- ✓ Módulo Buraco

Atores: Cidadão e Servidor

Caso de uso relacionado: Cadastrar Irregularidade

Pré-condições: Estar identificado no sistema;

Pós-condições: Nova irregularidade cadastrada no sistema

Fluxo principal

(1) O sistema apresenta a tela de inserção de dados correspondente ao módulo previamente selecionado (Buraco)
(2) O ator preenche os dados referentes ao buraco: endereço que mais se aproxima da localização do mesmo, seu potencial subjetivo de danos e a pista
(3) O sistema comunica o sucesso da inserção da nova entrada

- ✓ Excluir Irregularidade

  Atores: Servidor

  Caso de uso relacionado: Nenhum

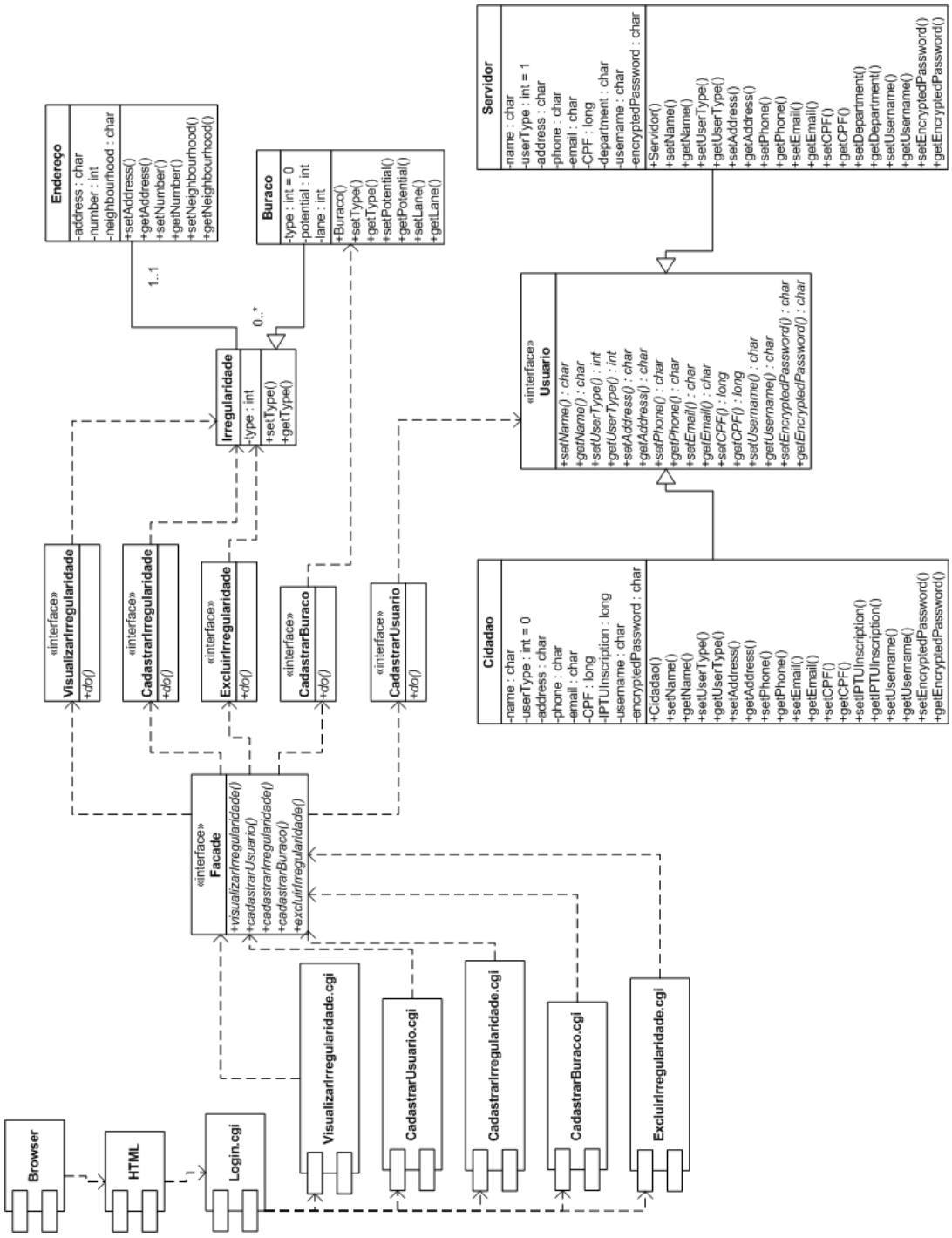
  Pré-condições: Estar identificado no sistema como servidor;

  Pós-condições: Exclusão de uma irregularidade considerada inválida ou solucionada

  Fluxo principal

  (1) O ator servidor clica no link de excluir irregularidade na tela de sucesso de identificação de usuário
  (2) O sistema apresenta uma tela de busca com campos de endereço e tipo de irregularidade
  (3) Se a busca retornar resultados, o ator servidor exclui algum deles

❖ Diagrama de Classes

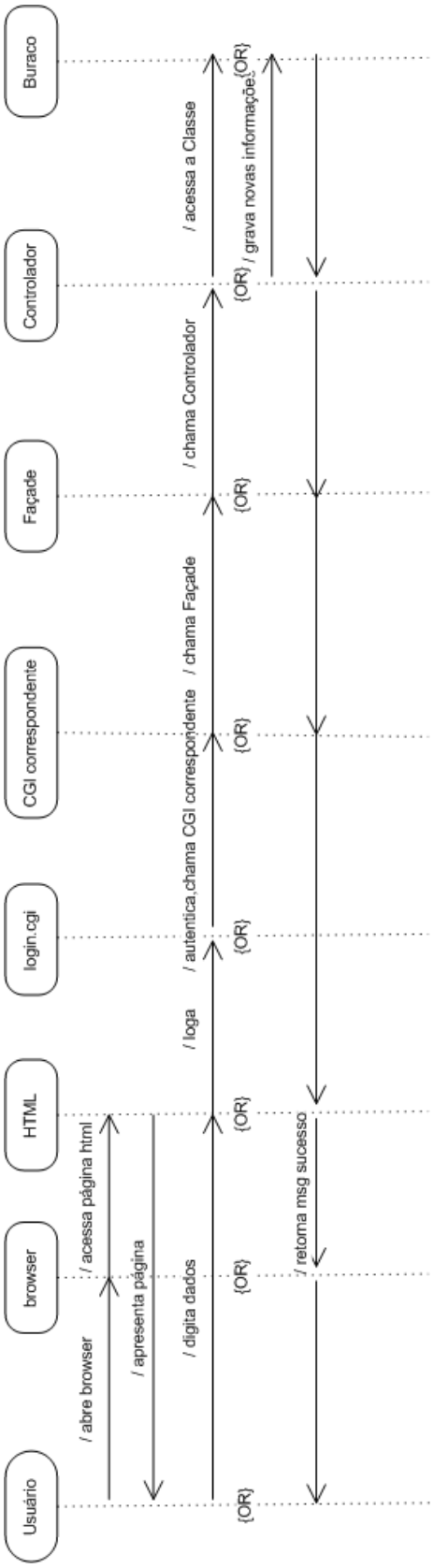


A classe abstrata Usuário é especializada pelas classes Cidadão e Servidor. A classe Cidadão apenas possui o número de inscrição no IPTU como especificidade, e a Servidor possui o departamento do funcionário.

Já a classe abstrata Irregularidade é especializada pela classe Buraco. Esta última possui as especificidades de cada irregularidade, no caso de buracos. Para a adição de novos módulos, novas classes especializarão a abstrata Irregularidade com as características especiais de cada tipo.

A classe Endereço faz parte de Irregularidade e foi representada separada devido a sua importância no contexto. Um endereço pode ter uma ou várias irregularidades (nunca nenhuma, pois, neste caso, não há necessidade do endereço na base de dados do sistema, apenas na base do Google Maps), porém uma irregularidade só pode ter um endereço atribuído.

❖ Diagrama de Seqüência para o Caso de Uso Módulo Buraco



Potencial Subjetivo de Danos	Grau de 1 a 3 (pequeno, médio e grande) que o usuário considera como a intensidade dos possíveis danos causados pelo buraco existente. Esta é uma percepção pessoal e, portanto, subjetiva.
Pista	Número de 1 a 5 que representa em qual pista da via se encontra o buraco. A pista 1 é a mais à direita e a 5, a mais à esquerda. Como nem todas as vias possuem esta quantidade de pistas, o usuário deve ignorar os números que ultrapassarem a quantidade de pistas da via em questão.
Façade	Interface que faz a ligação entre as CGIs e as classes de controle. A etimologia da palavra indica que ela é de origem da mesma palavra francesa *façade* (só se diferenciando na pronúncia: fə-'säd x fasad, respectivamente), a qual, por sua vez, proveio do italiano *facciata*, de *faccia* – face, e por último, do latim coloquial *facia*. Data aproxidamente de 1681, e significa fachada. Fonte: Merriam-Webster Collegiate® Dictionary

3.3 Requisitos de Desempenho

O sistema deve ter um tempo de resposta razoável, que gira em torno de 5 segundos numa conexão de banda larga. O número de usuários simultâneos pode ser um problema no futuro, quando o sistema se tornar conhecido e usado pela maioria das pessoas, mas por enquanto não é um requisito relevante.

3.4 Restrições de Projeto

Não há restrições relacionadas a software, hardware ou normas, pelo contrário, o projeto foi idealizado à luz da máxima flexibilidade possível. O software utilizado é, em grande maioria, open-source.

3.5 Atributos

Os atributos de qualidade referentes ao projeto incluem: segurança, amigabilidade, interoperabilidade, devido à possibilidade de agregar módulos ao desenvolvimento já existente que cubram novas necessidades.

Apêndice C

Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro

DESCRIÇÃO DO PROJETO DE SOFTWARE

VERSÃO 1.0

22 de Julho de 2009

Organização:

Romulo Vanzillotta Castello

Antônio Cláudio Gómez de Sousa
Responsável pela aprovação do Plano

Sumário

1 Introdução

1.1 Finalidade

O projeto para o Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro – SMDU-RJ – tem um ciclo de vida em espiral, ou seja, o software passa por todo o processo até haver uma avaliação do usuário e, dependendo do resultado, o ciclo se reinicia [3]. É dirigido aos analistas e programadores responsáveis pelo sistema.

1.2 Escopo

O Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro – SMDU-RJ – tem como objetivo geral propor um sistema capaz de servir como base de dados das irregularidades existentes na cidade, tanto aos cidadãos como aos funcionários da prefeitura, de forma prática e usável. Aos funcionários, se prestaria a guiar seu trabalho de ações corretivas e preventivas, e aos cidadãos se prestaria como um alerta enquanto as situações não fossem resolvidas, além de um canal direto com a prefeitura. Desta forma, pretende-se conter e eliminar a crescente desordem presente na cidade do Rio de Janeiro. O escopo de trabalho do projeto é a criação de um ambiente via Internet, onde os usuários possam cadastrar dados referentes aos problemas percebidos. Ou seja, possibilitar o registro e acesso a estas informações de forma confiável e on-line.

1.3 Definições e Acronismos

SMDU-RJ: Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro, nome do software;

UML: Unified Modeling Language, acronismo inglês para Linguagem de Modelagem Unificada. Ela é não-proprietária e de terceira geração, o que significa que é aberta, sem custos e auxilia a visualizar a comunicação entre os objetos como um todo;

CGI: Common Gateway Interface, ou interface de passagem comum. Mecanismo de acesso através dos provedores http para programação e registros de informações na Internet;

HTML: HyperText Markup Language ou linguagem HTML. Código para veicular textos e imagens na Internet.

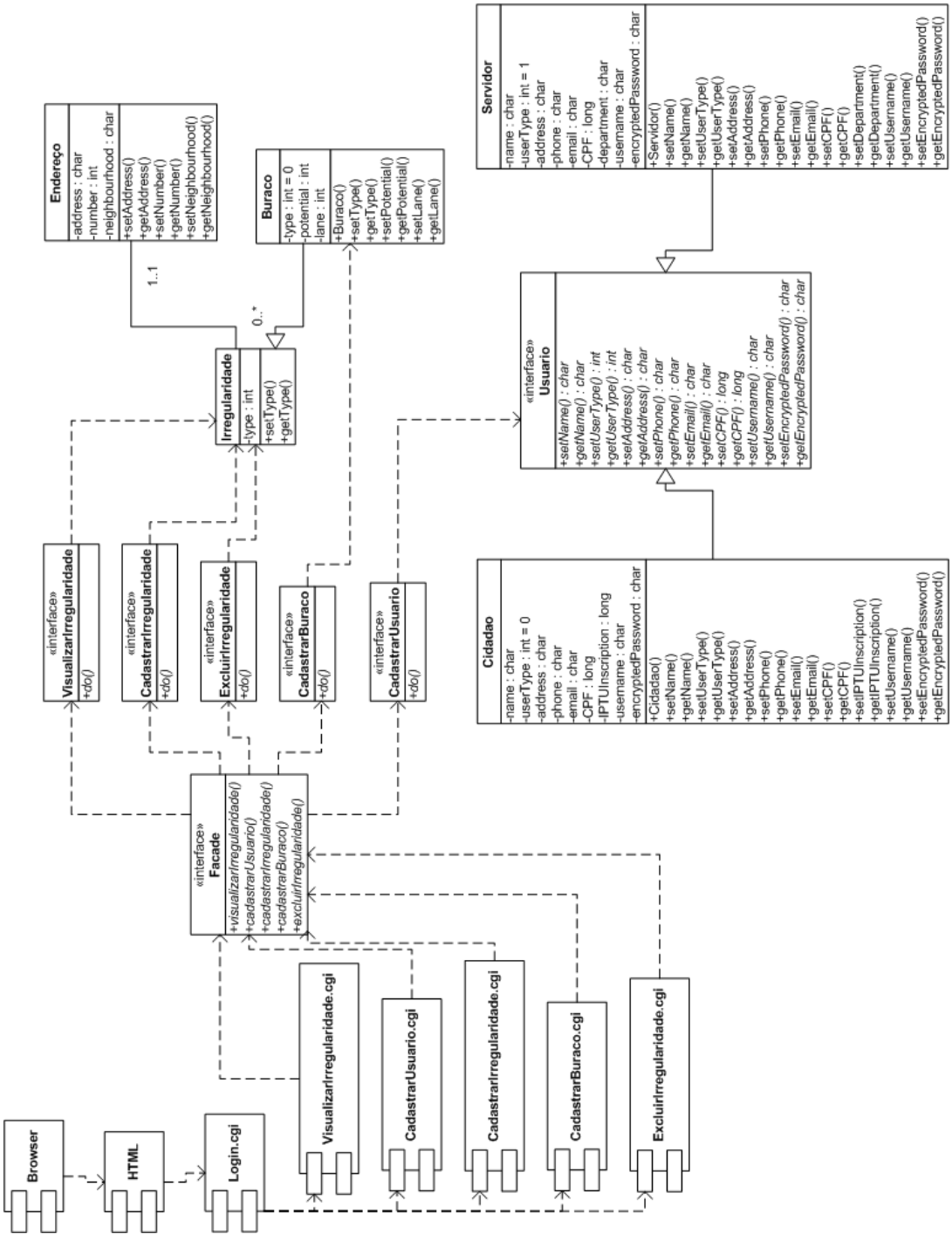
2 Referências

Os documentos referenciados são:

- NORMA: DESCRIÇÃO DE PROJETO DE SOFTWARE [12] ;
- PLANO PARA O GERENCIAMENTO DE PROJETO DE SOFTWARE - Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro versão 1.0. Data: 09/07/2009; responsável: Romulo Vanzillotta Castello;
- ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE - Sistema de Monitoramento da Desordem Urbana no Município do Rio de Janeiro versão 1.0. Data: 19/07/2009; responsável: Romulo Vanzillotta Castello;

3 Decomposição

3.1 Decomposição em Módulos



3.2 Decomposição em Processos Concorrentes

Os clientes concorrerão pelo servidor através de threads.

3.3 Decomposição de Dados

A decomposição de dados pode ser visualizada no diagrama de classes.

4 Descrição das Dependências

O Diagrama de classes já contempla as dependências entre os módulos.

5 Descrição das Interfaces

5.1 Interfaces dos Módulos

5.1.1 Interfaces do Façade

visualizarIrregularidade;
cadastrarUsuario;
cadastrarIrregularidade;
cadastrarBuraco;
excluirIrregularidade.

5.1.2 Interfaces dos controladores (VisualizarIrregularidade, CadastrarUsuario, CadastrarIrregularidade, CadastrarBuraco, ExcluirIrregularidade)

Os controladores contém a lógica, isto é, as regras de negócio do sistema. Eles são chamados pela CGI para executar as ações propriamente ditas. Mais detalhes no item 6.1.2, Detalhamento dos controladores.

5.1.3 Interfaces do módulo Buraco

Buraco (construtor);
setType;
getType;
setPotential;
getPotential;

setLane;

getLane.

5.1.4 Interfaces dos módulos Cidadão e Servidor

Cidadao (construtor);

setName;

getName;

setUserType;

getUserType;

setAddress;

getAddress;

setPhone;

getPhone;

setEmail;

getEmail;

setCPF;

getCPF;

setIPTUInscription;

getIPTUInscription;

setUsername;

getUsername;

setEncryptedPassword;

getEncryptedPassword.

O módulo Servidor apenas se diferencia pelo construtor e pelos métodos setDepartment e getDepartment, os quais substituem as interfaces setIPTUInscription e getIPTUInscription do Cidadão.

5.2 Interfaces entre Processos

Os processos não se interfaceiam, já que foram desenhados de forma que ficassem bem divididos entre si.

6 Projeto Detalhado

6.1 Projeto Detalhado dos Módulos

6.1.1 Detalhamento do Façade

O Façade faz a separação de camadas. Ele é chamado pelas CGIs e seus métodos somente chamam os métodos "do" dos controladores, para que estes, por sua vez, possam acessar os métodos das classes internas. Isto garante a segurança do sistema. Exemplo:

CadastrarUsuario(name, address, phone, email, CPF, IPTUInscription, department, username, encryptedPassword)

CadastrarUsuario.do(name, address, phone, email, CPF, IPTUInscription, department, username, encryptedPassword)

6.1.2 Detalhamento dos controladores

Atuam como mais um elemento da separação de camadas. Seus métodos "do" chamam os métodos das classes das entidades e passam os parâmetros, aumentando a segurança do sistema. Exemplo:

CadastrarUsuario.do(name, address, phone, email, CPF, IPTUInscription, department, username, encryptedPassword)

CadastrarUsuario.setName(name);
CadastrarUsuario.setAddress(address);
CadastrarUsuario.setPhone(phone);
CadastrarUsuario.setEmail(email);
CadastrarUsuario.setCPF(CPF);
CadastrarUsuario.setIPTUInscription(IPTUInscription);
CadastrarUsuario.setDepartment(department);
CadastrarUsuario.setUsername(username);
CadastrarUsuario.setEncryptedPassword(encryptedPassword).

6.1.3 Detalhamento do módulo Buraco

Potencial: potencial subjetivo de danos, grau de 1 a 3 (pequeno, médio e grande) que o usuário considera como a intensidade dos possíveis danos causados pelo buraco existente;

Pista: número de 1 a 5 que representa em qual pista da via se encontra o buraco;

O algoritmo dos métodos é simples e comum a todos, do tipo:

```
setName(name)
   this.name = name;

name getName()
   return this.name;
```

Só variam os tipos de dados entrados.

❖ Diagrama de Seqüência para o Caso de Uso Módulo Buraco