SMARTHOUSE – DESENVOLVIMENTO DE UMA INTERFACE WEB PARA AUTOMAÇÃO RESIDENCIAL

Thaian Andreia, Felipe Ferreira, Ramon Amorim, Luís Ricardo Sabino, Andrew Kretzer, Matheus Engleitner

Resumo: *Este documento apresenta, detalhadamente, o processo de desenvolvimento do projeto SmartHouse, uma interface Web de comunicação voltada para automação residencial desenvolvido para as disciplinas de Sistemas Embarcados I e Engenharia de Software II do curso de engenharia de Computação da Unisociesc.*
Serão apresentados neste artigo os requisitos, funcionalidades, tecnologias utilizadas, desenvolvimento, e implementação do sistema.

Palavras-chave: *SmartHouse, Automação, Residencial*

1 INTRODUÇÃO

Devido a atual situação econômica e ambiental do país, na qual temos um elevado custo mensal com tributações e a escassez de recursos naturais em diversas regiões, o consumo sustentável tornou-se um tema bastante abordado no cotidiano dos brasileiros.

Tendo como uma alternativa na redução dos gastos, a automatização residencial, que possibilita o controle de gastos e automatização de processos residenciais, proporciona ao seu usuário benefícios econômicos, conforto e agilidade em algumas tarefas do dia a dia, além de contribuir com questões ambientais, como a redução do consumo dos recursos naturais, através do controle e identificação de vazamentos de água ou equipamentos defeituosos na rede elétrica.

Pensando nos pontos citados anteriormente, fica a dúvida de como criar uma forma de resolver esses problemas. Será que um sistema de automatização residencial seria uma boa solução? Quais ferramentas esse sistema deveria ter e como seria a utilização dessas ferramentas pelo usuário que o adquirir?

A partir dessas perguntas, será desenvolvido o projeto SmartHouse, que une os benefícios da automatização residencial a um software de monitoramento interativo, com diversos recursos, que trará ao seu usuário uma forma simples de acompanhar seu consumo residencial remotamente, além de realizar diversas funções que trarão economia e conforto.

2 REQUISITOS DE DESENVOLVIMENTO

2.1 Tecnologias Utilizadas

Front-End: ASP.Net, HTML5, SQL, JQuery, Javascript e CSS.
Back-End: C# .Net
Armazenamento de Dados: SQL Server 2012.

Framework: .Net Framework 4.5, Bootstrap Admin LTE.
Transmissão de dados: protocolo de rede TPC/IP, comunicação via cabos ETHERNET, envio e recebimento de dados por variáveis de texto via WebService. Servidor web IIS, a aplicação rodará de forma local para garantir que funcione corretamente durante a apresentação, não havendo assim a necessidade de conexão via internet.
Plataforma Operacional: Sistema Operacional Microsoft Windows 7.
Plataforma de desenvolvimento: Microsoft Visual Studio 2013.

2.2 Especificação de Hardware Utilizado

Modelo: Acer Aspire E1-572-6
Processamento: Intel i5-4200U 1.6GHz
Armazenamento: HD SATA 500 GB HDD
Memória física: 4GB DDR3 L 800MHz

3 REQUISITOS DO SISTEMA

3.1 Requisitos Funcionais

RF1 Consultas
Descrição: O sistema deve fornecer ao usuário diversos indicadores dos dados que serão coletados, para que o usuário possa acompanhar os gastos residenciais. Esses indicadores serão gerados a partir dos dados de consumo que serão enviados para o sistema a partir dos microcontroladores, eles serão um resumo do consumo de água, energia elétrica, além dos dados coletados pelos demais sensores.
Prioridade: Alta

RF2 Interação
Descrição: O sistema deve fornecer ao usuário formas de interagir com os dispositivos físicos do projeto por meio de comandos do sistema, esses comandos serão componentes em tela que possibilitem a entrada de valores que serão enviados aos microcontroladores para realizar uma tarefa específica como ligar ou desligar equipamentos residenciais.
Prioridade: Essencial

RF3 Relatórios
Descrição: O sistema deverá realizar a emissão de relatórios de consumo a partir dos dados gravados na base.
Prioridade: Alta

3.2 Requisitos não funcionais

RNF1 Ambiente de desenvolvimento

Descrição: O ambiente de desenvolvimento deverá ser escolhido de acordo com as necessidades no sistema e conhecimento dos integrantes da equipe.
Prioridade: Alta

RNF2 Comunicação
Descrição: O sistema deverá se comunicar com os dispositivos Arduíno de forma que seja possível o envio e recebimento dos dados.
Prioridade: Alta

RNF3 Acessibilidade
Descrição: O sistema deve permitir que o usuário o acesse remotamente, via internet, realizando as consultas e interações de qualquer lugar
Prioridade: Alta

RNF4 Segurança
Descrição: O sistema deve garantir que não haja interferência de terceiros nas informações coletadas, através de criptografia de dados.
Prioridade: Alta

RNF5 Interface de usuário
Descrição: A interface do sistema deverá ser de fácil entendimento ao usuário, mostrando dados detalhados, que informem de maneira coerente o consumo de um período.
Prioridade: Alta

RNF6 Responsividade
Descrição: O sistema deve possuir recursos responsivos, ou seja, se adaptar de acordo com a tela do dispositivo de acesso.
Prioridade: Alta

RNF7 Base de Dados
Descrição: O sistema deverá guardar as informações de consumo colhidas pelos microprocessadores.
Prioridade: Alta

RNF8 Modelos de Visualização
Descrição: O projeto do sistema deverá apresentar os três modelos PIM (Casos de Uso, Modelo Hierárquico de requisitos e Modelo) e os dois modelos PSM (Modelo de Bloco de e Modelo de Bloco Interno).
Prioridade: Alta

RNF9 Refinamento de consultas
Descrição: Os relatórios serão emitidos de acordo com os valores dos filtros que serão disponibilizados ao usuário para melhor refinamento das consultas.
Prioridade: Alta

RNF10 Rede

Descrição: Será necessária a criação de uma rede local que fará a conexão entre o servidor da aplicação e os microprocessadores.
Prioridade: Alta

4 METODOLOGIA

4.1 Ambiente de trabalho

Para o desenvolvimento do sistema, é necessário realizar algumas configurações para que seja possível a comunicação entre um projeto rodando numa máquina local e dispositivos físicos que enviarão e receber informações.

A comunicação é feita através do IIS (serviços de informação de internet), o projeto ASP.NET que será desenvolvido terá um site local no IIS, a configuração é feita através de um IP fixo atribuído à máquina em que o projeto está instalado e uma porta específica.

Deve-se desabilitar o firewall da rede local para que a comunicação entre os periféricos seja permitida, na plataforma de desenvolvimento, a URL de execução do projeto será a mesma do site criado no IIS.

Após realizar corretamente estas etapas, o projeto estará acessível em uma rede local e o desenvolvimento será iniciado.

4.2 Banco de Dados

Os dados do sistema serão armazenados em um banco de dados SQL Server, para isso, será utilizado o SQL Server Instalation Center, está ferramenta permite a criação de instâncias de banco de dados e instalação de programas e serviços disponíveis do SQL Server.

Para o projeto, além da criação da instância e serviços necessários para o funcionamento da mesma, será utilizado o SQL Server 2014 Management Studio para gerenciamento do banco de dados, tabelas e registros.

4.3 Projeto SmartHouse

Conforme os requisitos, o Web Site do projeto será desenvolvido em ASP.NET, nas linguagens ASPX para a camada visual e C# para a camada lógica.

O código fonte do projeto pode ser acessado em:
https://github.com/thaianandreia/SmartHouse.git

4.3.1 Criação:

No Visual Studio, será criado uma nova solução (Solution, arquivo .sln) em branco, nomeada Smarthouse, conforme especificado no blog oficial da Microsoft, as soluções são

estruturas para a organização de projetos no Visual Studio, no caso do sistema SmartHouse, a solução terá dois projetos, o projeto App, criado para a camada visual e o projeto Lib, onde estarão as bibliotecas, classes, funções e lógica do sistema.

A arquitetura do sistema será baseada no padrão MVC (Model View Controller), que, basicamente, um padrão moderno para aplicações Web, nele, as camadas separadas trazem maior organização ao projeto, facilitando a modularização e manutenção do código. Esta escolha foi feita devido ao grande número de necessidades que o sistema poderá abranger futuramente e que, com o uso dessa arquitetura, a implementação de novas funcionalidades será mais fácil e organizada.

4.3.2 Desenvolvimento do Web Site (Projeto App)

Inicialmente, será adicionado ao projeto o template Admin LTE, um Framework para front-end, baseado no Twitter Bootstrap 3, uma coleção de ferramentas de código aberto para a criação de websites e aplicações web. O template Admin LTE foi escolhido devido às suas funcionalidades de Dashboart e formulários que serão necessários à interface do projeto.

No projeto App será criada uma pasta (assets) para adicionar todos os arquivos do template.

O ASP.Net possibilita a criação de Master Pages, estas funcionam como um tipo de página mestra a todas as demais páginas, abstraindo o código que é comum a todas elas. No caso do projeto, na Master Page será linkado todos os arquivos .js (scripts jQuery e javascript) e .css (formatação e estilo dos componentes do template) que serão utilizados na aplicação, além da construção do modelo das páginas, como rodapé, menu principal e cabeçalho do sistema, deixando para os arquivos de interface somente a parte específica de cada um.

O mesmo será feito no back-end do projeto App, adicionando qualquer função que será comum a todos os arquivos .cs no arquivo .cs da Master Page.

Quanto à organização, ambos os projetos serão modulados, ou seja, as funcionalidades do sistema serão agrupadas em pastas de acordo com sua função, cada grupo de funções também terá um namespace diferente, ou seja, para chama-las, será necessário ao desenvolvedor especificar a qual módulo ela pertence.

O projeto Lib será referenciado pelo projeto App, isso significa criar uma ponte, um link de acesso, para que a interface “converse” com a lógica do sistema.

4.3.3 Desenvolvimento do Web Site (Projeto App)

De acordo com o padrão MVC, o projeto Lib será responsável pela lógica do sistema e comunicação deste com o banco de dados, funcionando como uma camada entre o usuário e o servidor de dados.

A organização do projeto, além de modularizada, também contará com um arquivo principal que contém propriedades e funções comum a todos, como, por exemplo, conexão com banco de dados, dados de acesso de usuário e funções de log do sistema.

Este projeto possuíra referência ao .Net Framework e bibliotecas de sistema, comunicação com o servidor, linguagem e de controle de transações. Essas bibliotecas são

necessárias ao sistema, no sentido de desenvolvimento e controle entre interações com os dados do servidor.

4.3.4 Comunicação externa

Após criado o projeto, comunicação entre interface e servidor estabelecidas, será dado início ao desenvolvimento da comunicação entre os microcontroladores Arduíno. Para esta etapa será necessário configurar tanto o projeto App, quanto os microcontroladores.

No projeto, toda comunicação externa será realizada a partir de funções e eventos em jQuery nas próprias páginas do web site ou por arquivos .asmx, que utilizam o padrão WebService do ASP.Net, este padrão funciona como uma ponte entre aplicações, no caso a aplicação do projeto e as aplicações de cada microcontrolador. Os WebServices utilizam arquivos XML para fazer a comunicação, ou seja, ele recebe e envia um pacote XML contendo as informações necessárias, o destinatário deste pacote processará as informações e realizar as funções requisitadas. O XML é um padrão de arquivo fácil de manipular, por esse motivo foi escolhida esta tecnologia.

4.3.5 Configuração no microcontrolador:

Os microcontroladores deverão estar configurados para estabelecer a comunicação com o projeto, sendo assim possível receber e enviar pacotes de informações via rede local.

Abaixo estão listados os ip's de cada um dos hosts conectados em rede que farão parte do sistema de automação residencial:

192.168.0.200 – Servidor

192.168.0.225 – Microcontrolador de energia

192.168.0.250 – Microcontrolador de segurança

192.168.0.202 – Microcontrolador de hidráulica

Os exemplos e códigos de comunicação entre o servidor e os microcontroladores estão disponíveis no repositório do projeto.

4.3.6 Configuração no projeto:

(POST) Uma requisição do sistema funcionará da seguinte forma: A página web requisitará alguma informação de um microcontrolador específico através de uma função em jQuery, a função terá os parâmetros necessários e o endereço IP do microcontrolador de destino, está função enviará um pacote de informações formatadas para o endereço do microcontrolador.

O microcontrolador será responsável pelo processamento das informações e entrega do retorno para o sistema.

O sistema informará ao usuário se a requisição foi ou não realizada com sucesso.

```
$(".switch").bootstrapSwitch().on('switchChange.bootstrapSwitch', function (event, state) {
    var codigo = $("#" + event.currentTarget.id).data("codigo");
    if (codigo < 10)
        codigo = codigo[1];

    console.log(codigo);

    $.ajax({
        type: "POST",
        url: "http://<%= IP_Hidraulica %>/toggle/" + codigo,
        //data: dto,
        contentType: "application/jsonp; charset=utf-8",
        dataType: "jsonp"
    });
});
```

Imagem 1: Exemplo de método post para o microcontrolador

(GET) O recebimento de pacotes pelo sistema será enviado pelo microcontrolador ao sistema através de uma url direta ao webservice, passando os parâmetros e valores necessários. O sistema processará essas informações e enviar uma resposta ao microcontrolador.

Por todos os componentes de hardware estarem ligados em uma rede local, faz-se necessário desabilitar o firewall, para que o mesmo não bloqueie o envio e recebimento de pacotes na rede, futuramente, está previsto que o sistema seja acessível via internet, nesse caso, o projeto será adaptado para atender os requisitos de comunicação e segurança.

```
[WebMethod]
[ScriptMethod(UseHttpGet = true)]
public void Gravar(decimal temperatura, decimal umidade, int fogo, int gases, int presenca)
{
    ctxTemperatura.Inserir(temperatura);
    ctxUmidade.Inserir(umidade);
    SmartHouse.SEG.PresencaVO senderP = new SmartHouse.SEG.PresencaVO();
    senderP.CodigoComodo = 1;
    senderP.IndicadorPresenca = presenca == 1;
    ctxPresenca.Inserir(senderP);
}
```

Imagem 2: Exemplo de método do WebService que recebe valores de um microcontrolador

Note que, para receber as informações, é necessário adicionar a propriedade `[ScriptMethod(UseHttpGet = true)]`, ele permite o recebimento de informações vindas de hosts que não sejam o próprio servidor, sem essa instrução, a comunicação não pode ser estabelecida.

5 PROCESSOS

5.1 Segurança

Para o método Segurança, o sistema contempla as seguintes funcionalidades:

- Gráfico em tempo real para temperatura e umidade
- Indicador de presença, para quando houver movimentação em determinado cômodo
- Indicador de foco de incêndio
- Indicador de vazamento de gases

O modo de envio desses dados se dá por acesso direto do microprocessador ao WebService, ou seja, não é necessário que o sistema dê um Post no microcontrolador, o sistema dará um Post no próprio WebService, a fim de buscar, dinamicamente, os dados que popularão gráficos e indicadores de tempo real.

5.2 Hidráulica

Para o método Hidráulica, o sistema contempla as seguintes funcionalidades:

- Gráfico em tempo real de consumo de água
- Indicador de vazamento e/ou funcionamento de cada válvula
- Switch (tomada virtual) para ligar ou desligar uma válvula

5.2 Elétrica

Para o método Elétrica, o sistema comtempla as seguintes funcionalidades:

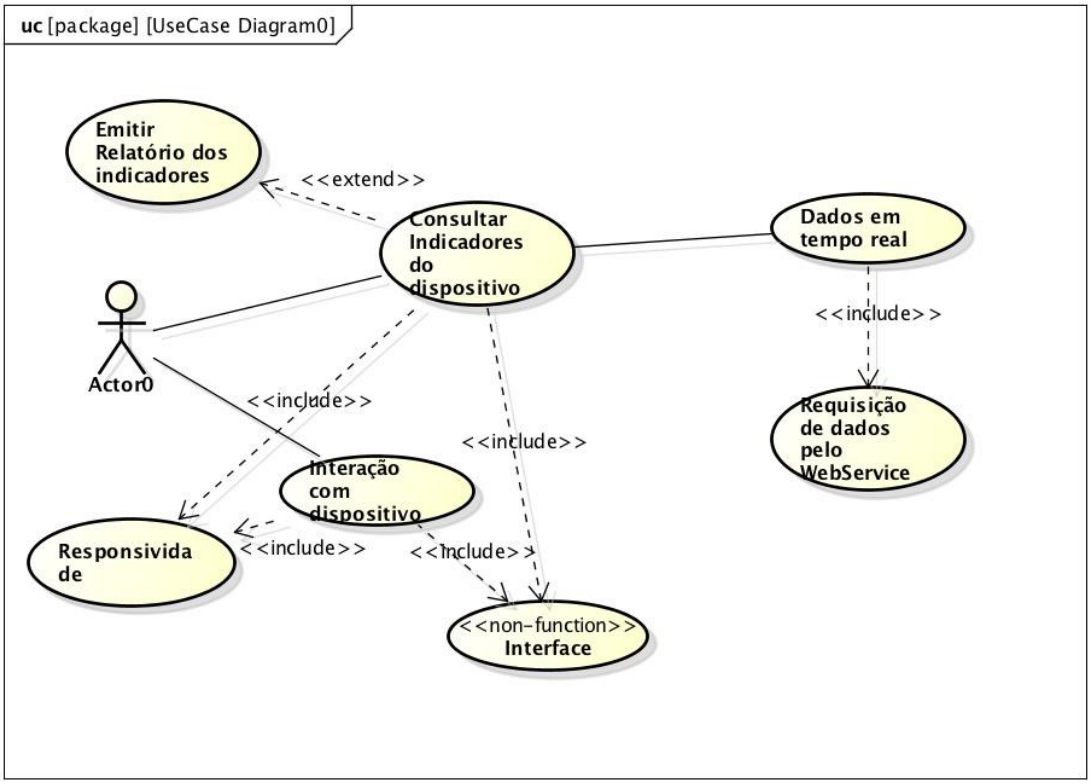
- Gráfico em tempo real de consumo de energia
- Switch (tomada virtual) para ligar ou desligar tomadas

Nos módulos de Hidráulica e Elétrica, a comunicação será feita através de requisições (Posts) do sistema aos microcontroladores, estes receberão a requisição e farão as tarefas conforme programados, após a execução das tarefas, o microcontrolador enviará um retorno ao sistema, que poderá ser um status de sucesso (switchies) ou um valor que será enviado ao WebService e salvo na base de dados (gráficos de tempo real e indicadores), para posteriormente serem utilizados na exibição de dados.

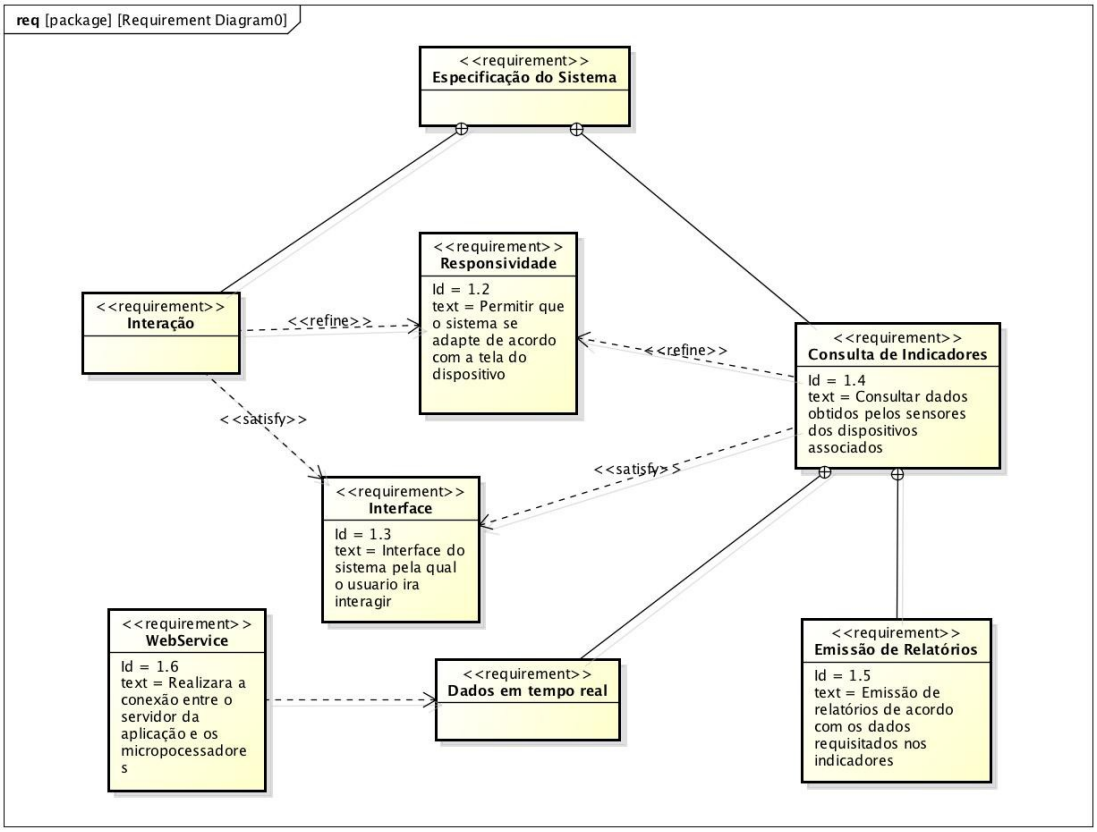
Além dos controles de tempo real, o sistema gerará gráficos, extratos, previsões de consumo, médias e diversos outros mecanismos de controle de gastos e monitoramento para o usuário através de telas de consultas com filtros de datas e/ou valores.

6 MODELAGEM PIM

6.1 Casos de Uso

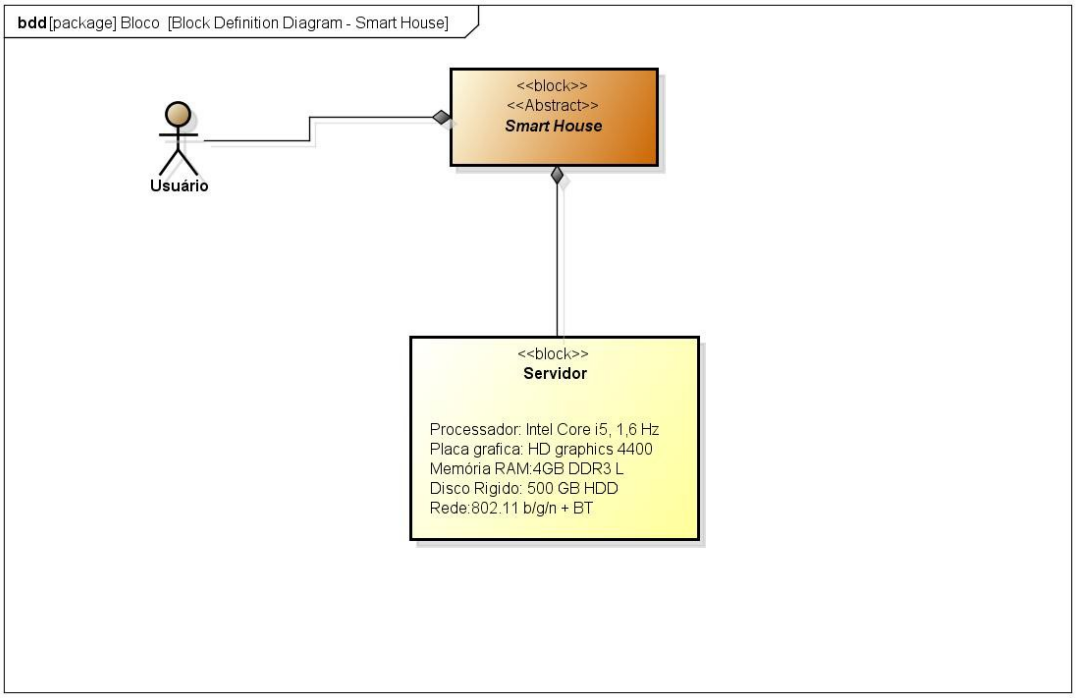


6.2 Diagrama de Requisitos

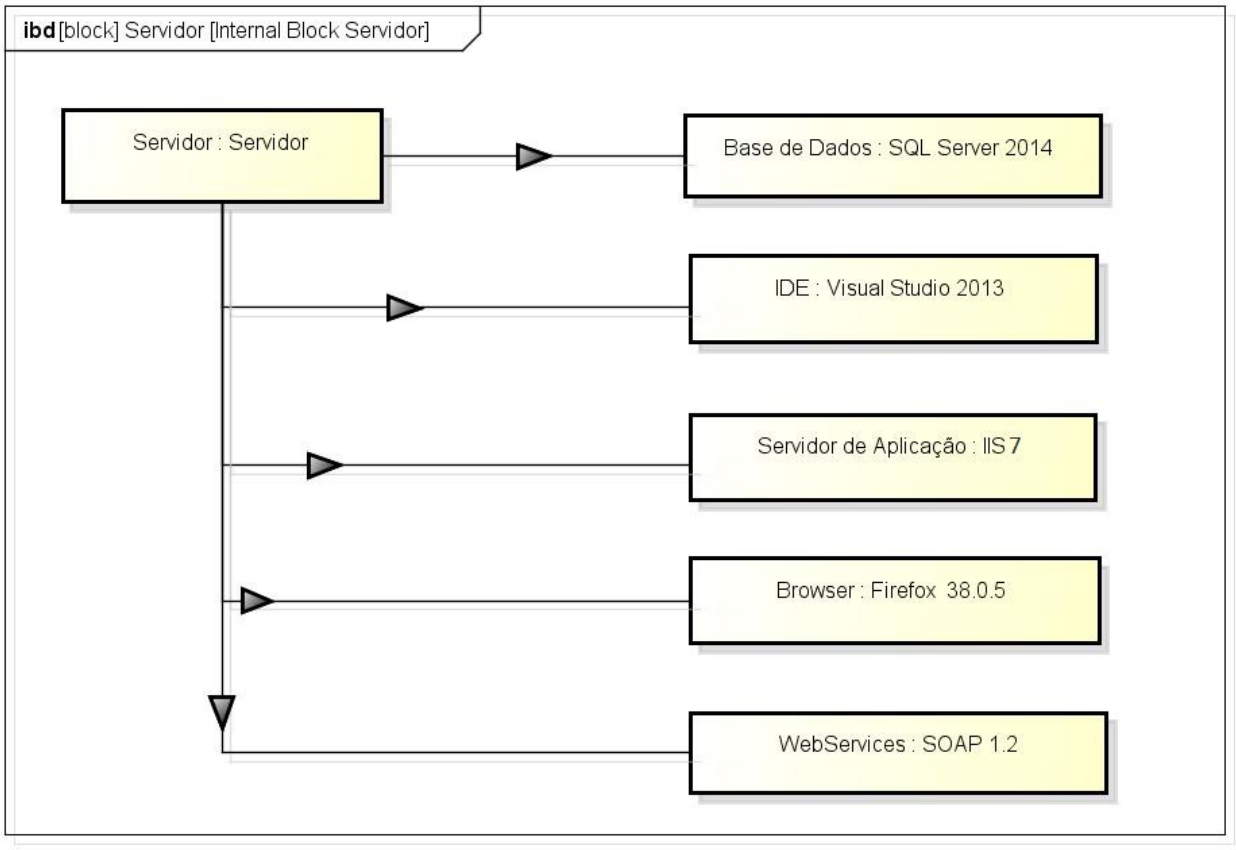


7 MODELAGEM PSM

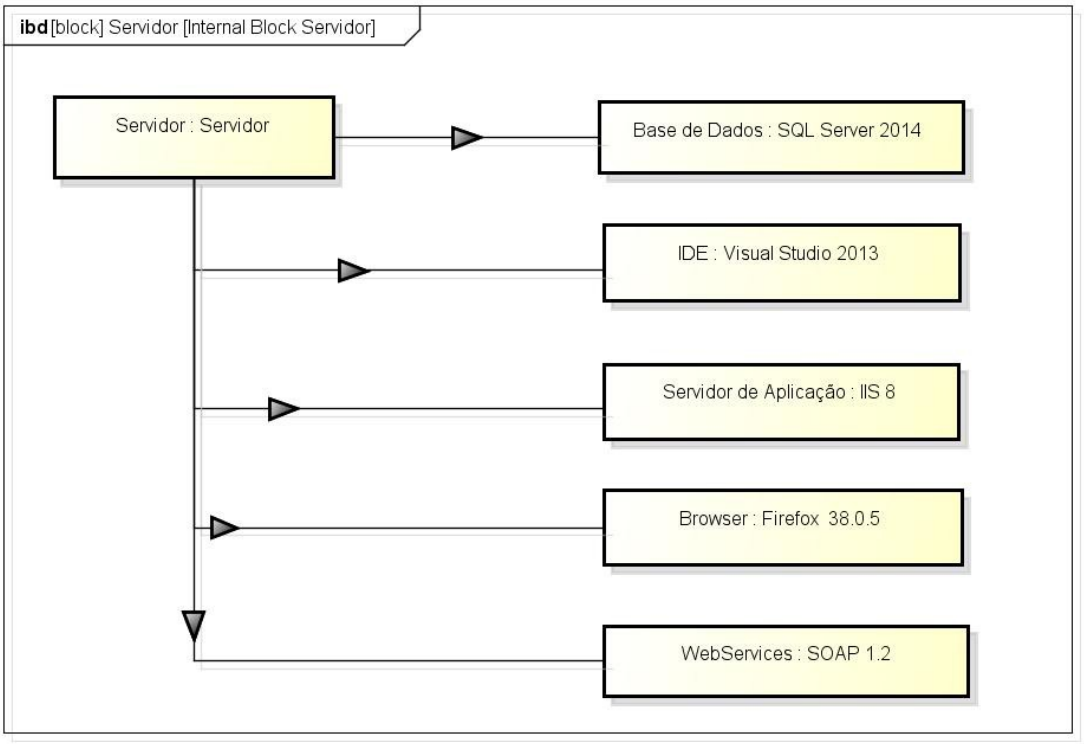
7.1 Diagrama de Bloco



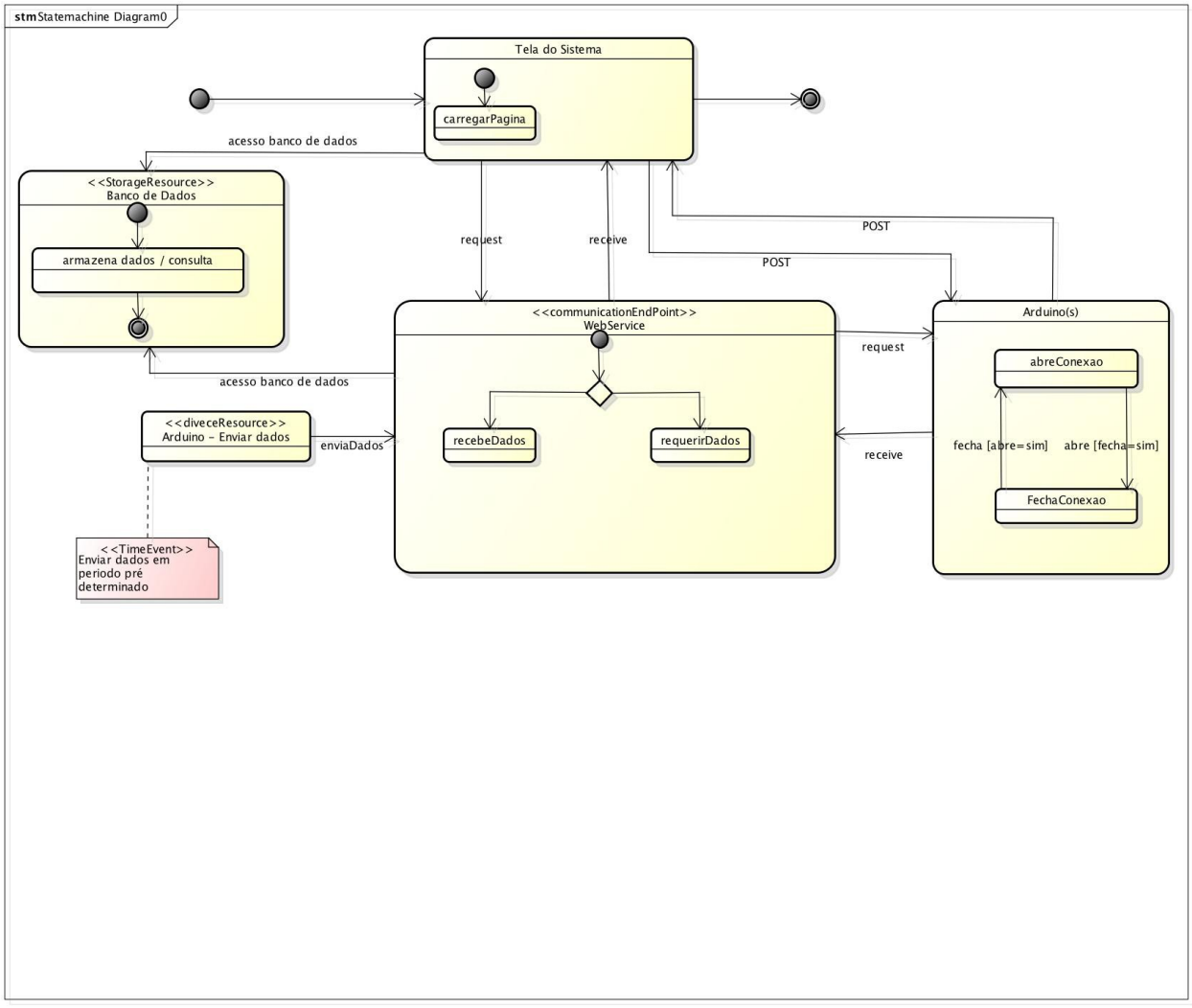
7.2 Bloco de Servidor Interno



7.2 Bloco de Servidor Interno



7.3 Modelo de Máquina de Estados



8 CONCLUSÃO

Em sua versão inicial, o sistema SmartHouse possui diversas funcionalidades que poderão ser modificadas ou aprimoradas e ainda incrementar o seu funcionamento de forma simples e segura, devido ao uso do padrão MVC, em testes, ele mostrou-se eficiente no que se diz respeito ao monitoramento e interação, sendo estes de implementação fácil, desde que seguido o padrão de desenvolvimento criado pelos seus programadores.

Além das funcionalidades e uso do sistema, a conclusão desse projeto trouxe aos seus envolvidos um grande aprendizado, tanto nas tecnologias exploradas, estudadas e aplicadas quanto na organização de um grupo de desenvolvimento no cumprimento satisfatório das metas de um projeto.

9 REFERÊNCIAS

MICROSOFT. Solution(.Sln) File.
Disponível em: <https://msdn.microsoft.com/en-us/library/bb165951%28v=vs.140%29.aspx>
Acesso em: 27 abr. 2015.

MICROSOFT. Model-View-Controller.
Disponível em: <https://msdn.microsoft.com/en-us/library/ff649643.aspx>
Acesso em: 07 abr. 2015.

BOOTSTRAP. Bootstrap 3.3.4 release. 16 mar. 2015.
Disponível em: <http://blog.getbootstrap.com/2015/03/16/bootstrap-3-3-4-released/>
Acesso em: 25 mai. 2015.

MEILING, S., STEINBACH, T., DUGE, M., SCHMIDT, T.C. Consumer-oriented integration of smart homes and smart grids: A case for multicast-enabled Home Gateways?, Berlin, 9 set. 2013
Disponível em: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6698009&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6698009>
Acesso em: 3 mai. 2015.

MICROSOFT. Master Pages no ASP.NET 2.0
Disponível em: <https://msdn.microsoft.com/pt-br/library/cc580600.aspx>
Acesso em: 3 mai. 2015

MICROSOFT. Using ASP.NET Web Services
Disponível em: <https://msdn.microsoft.com/en-us/library/t745kdsh%28v=vs.90%29.aspx>
Acesso em: 9 jun. 2015

O arquivo fonte do projeto SmartHouse pode ser acessado em: https://github.com/thaianandreia/SmartHouse