9,0
(nove)

Fábio Alexandre Narita

SISTEMA DE SUPERVISÃO PARA AUTOMAÇÃO PREDIAL

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para graduação em Engenharia Mecânica, com ênfase em Automação e Sistemas.

São Paulo
2002

Fábio Alexandre Narita

SISTEMA DE SUPERVISÃO PARA AUTOMAÇÃO PREDIAL

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para graduação em Engenharia Mecânica, com ênfase em Automação e Sistemas.

Área de Concentração:
Engenharia de Automação e Sistemas

Orientador:
Prof. Dr. José Reinaldo Silva

São Paulo
2002

AGRADECIMENTOS

Ao meu orientador Prof. Dr. José Reinaldo Silva pelo aprendizado, apoio e compreensão.

Aos amigos Igor Neiva Camargo, Henrique Tatsuyuki Soejima e Raucer Curdulino pela ajuda e apoio ao meu trabalho.

A todos que direta ou indiretamente colaboraram na execução deste trabalho.

E a Deus pela minha família e pela vida.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

O desenvolvimento de um sistema de supervisão para automação predial tem como objetivo agregar novas funcionalidades ao sistema, além de auxiliar nas tarefas de controle da planta do edifício. Para tanto, através da implementação de um sistema de informação relacionado com os dados fornecidos pelos diversos elementos que compõem o prédio, pode-se extrair novas informações, bem como analisar e até prever comportamentos oriundos de eventos ocorridos na planta.

Tendo esse objetivo em mente, o presente trabalho teve como foco o estudo e a análise de um ambiente envolvendo a automação predial e o desenvolvimento de um sistema supervisório com a função de auxiliar nas tarefas de controle do prédio. Para isso, foi utilizada uma maquete como bancada de teste para que se pudesse modelar e analisar três tipos de sistemas: supervisão de segurança, incêndio e conforto térmico.

1 Introdução

O interesse pela automatização tem crescido ao longo dos anos, trazendo benefícios como redução de custos operacionais, otimização de processos, segurança e confiabilidade, entre outros. Entretanto, esse tema, muitas vezes visto em ambientes de manufatura, encontrou lugar também em prédios e residências, trazendo todas as vantagens acima citadas.

Hoje já é possível encontrar edifícios que possuem controle centralizado e automatizado da temperatura de seus ambientes, da segurança, de alarmes como o de incêndio, de luminosidade, entre outros. Isso acaba resultando em benefícios tanto econômicos, tais como utilização racional da energia e de elevadores, como na rapidez com que as ações são tomadas, como em casos de emergências de incêndio, por exemplo.

O trabalho desenvolvido como projeto de formatura versa sobre o tema da automação predial. Este projeto foi dividido em duas partes, sendo que cada uma será realizada paralelamente por grupos diferentes, ambos utilizando o conceito de sistemas de informação como suporte ao trabalho. O projeto desenvolvido por Marco Dyodi Takahashi e Patrick M. von Schaaffhausen teve como foco a implementação de um sistema de informação com a finalidade de interagir com a planta e seus elementos (CLP, sensores e atuadores). Complementarmente, o sistema desenvolvido neste trabalho objetivou a criação de um sistema de informação que pudesse auxiliar na tomada de decisões e fornecer novas informações baseadas na análise daquelas fornecidas pelo sistema por eles implementado.

Com isso, este trabalho vem a se juntar ao deles, auxiliando e complementando a tarefa de automatizar um edifício de forma integrada. Mais do que isso, o sistema de informação

ganha a função de supervisor, trabalhando em uma camada acima, agregando assim novas funcionalidades à tarefa de controlar todo o sistema.

Um exemplo disso pode ser visto na seguinte situação: suponha-se que um sensor de calor detecta uma elevação na temperatura de uma sala. Como proceder para se saber se há um foco de incêndio ou não? Um erro na determinação da situação atual pode levar a uma concepção errada sobre os eventos que ocorrem no sistema, causando transtornos como alarmes falsos ou falhas no acionamento de contra-medidas de incêndio, podendo resultar em acidentes e/ou inconvenientes.

O desenvolvimento do sistema acima proposto foi baseado na utilização das seguintes ferramentas:

- UML (Unified Modeling Language): utilizada como ferramenta de modelagem do sistema, auxiliando na análise de aspectos funcionais, relacionais e de projeto;
- JAVA: linguagem de programação utilizada para a implementação do sistema, sendo utilizada para acessar o banco de dados do sistema e possibilitar o desenvolvimento de interface gráfica com o usuário via internet;
- SQL (Structured Query Language): linguagem utilizada para realizar consultas e inserções no banco de dados do sistema;
- E no conceito de sistemas de informação introduzido por François Vernadet [Vernadet 96].

2 Objetivos

O objetivo deste trabalho é estudar e desenvolver um sistema de informação integrado com a planta de controle de um prédio, utilizando para isso um banco de dados. Essa integração permite auxiliar tanto no acompanhamento como na especificação de estratégias de controle de acordo com os eventos da planta, possibilitando também o armazenamento do histórico dos acontecimentos. Com isso, pretende-se chegar na integração dos diversos sistemas da planta (seja de supervisão ou de controle) para que se obtenha uma atuação mais rica e refinada.

Para tanto, a modelagem é baseada na análise dos aspectos funcional, informacional, organizacional e de integração do sistema como um todo. Com isso, objetiva-se um sistema em que haja cooperação de ações e definição de comportamentos, para que se tenha um controle mais próximo da necessidade real. Resulta então um modelo funcional que está em consonância com o modelo comportamental do sistema.

Prossegue-se então com o desenvolvimento de um sistema de informação agregado ao sistema de controle, de modo a se ter a integração dos diversos sistemas envolvidos na automatização da planta, conferindo assim unidade tanto funcional quanto comportamental. Assim, são obtidos os elementos necessários para o refinamento das decisões de controle, resultando numa união harmoniosa entre os sub-sistemas. Com isso, tem-se as ferramentas que auxiliam no fechamento da malha, fazendo com que o modelo de controle tenha melhor aderência à situação real (planta).

O sistema de informação tem como objetivo supervisionar e auxiliar a tarefa de controle, através da análise das informações providas por cada sistema ou objeto de controle. O sistema de informação é formado por três sistemas de supervisão:

1. Sistema de Supervisão de Incêndio;
2. Sistema de Supervisão de Temperatura;
3. Sistema de Supervisão de Presença.

Cada sistema de supervisão é responsável pelo monitoramento e pelo fluxo de informações entre os diferentes ambientes do prédio (salas, elevador e hall de entrada). Para simular o sistema será construída uma maquete em escala reduzida contendo todas as funcionalidades acima citadas.

Em resumo, pode-se listar como os objetivos desse trabalho os seguintes itens:

- Estudo do ambiente a ser automatizado (maquete), seus ambientes, sensores e atuadores e como cada um interage entre si;
- Estudo e modelagem do fluxo das informações oriundas dos diversos elementos presentes na maquete;
- Modelagem dos processos existentes e seqüências alternativas;
- Modelagem e desenvolvimento de um sistema de informação que armazene os dados gerados pelos elementos da maquete;
- Modelagem e desenvolvimento do sistema supervisório, com o intuito de auxiliar o sistema de informação que está diretamente relacionado com a maquete.

3 Descrição do Sistema em Estudo (Maquete)

O estudo, análise e modelagem do sistema de informação para automação predial serão baseados numa maquete com as seguintes características:

- Hall de entrada: com acesso ao elevador;
- Primeiro e Segundo andar: cada um com duas salas e acesso a elevador.

Abaixo segue o esquema da maquete utilizada como base de estudo para o projeto.

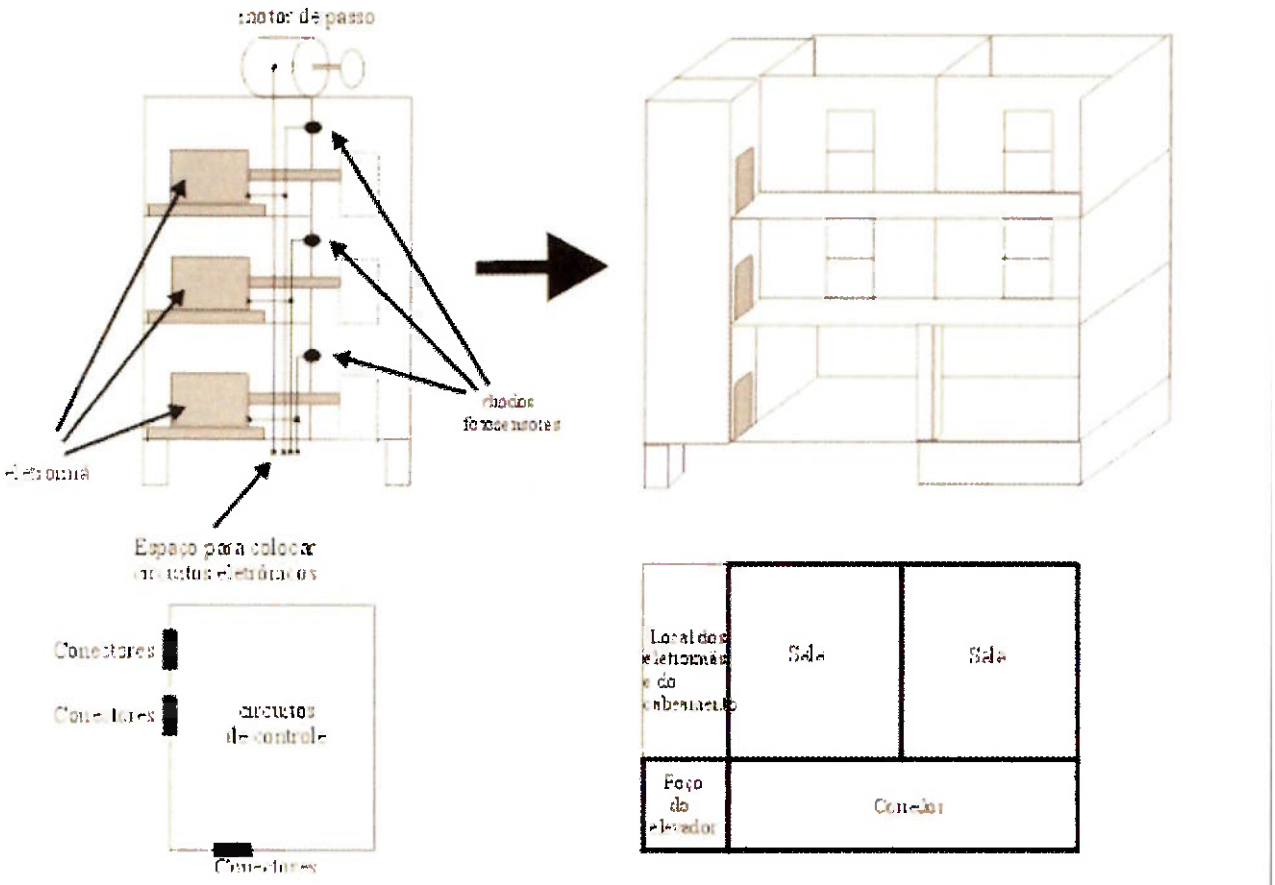


Figura 1 – Maquete e esquema do prédio e suas salas.

O hall de entrada possui uma webcam, utilizada para possibilitar a visualização do ambiente térreo da maquete por um usuário através da internet.

Cada sala da maquete possui os seguintes elementos:

- Um sensor de temperatura;
- Um sensor de presença;
- Um sensor de fumaça;
- Um ventilador, usado para controlar a temperatura.

O elevador possui somente um sensor de presença, que será utilizado para prover informações sobre a ocupação do mesmo, sendo importante para o sistema de segurança e de emergência.

4 O Sistema Supervisório

O objetivo do sistema supervisório é aproveitar as diversas informações colhidas pelo sistema de informação relacionada aos elementos da maquete, temperatura e presença, e relacioná-las, com a finalidade de, além de gerar novas informações, analisá-las para que se possa prever, por exemplo, padrões que resultem em um possível foco de incêndio.

Além disso, esse sistema consegue ter uma visão geral dos elementos que compõem a maquete, enquanto o sistema de informações agregado ao controle possui seu foco em cada ambiente, não tendo a capacidade de integrar e mesclar os dados dos demais elementos. Com isso, esse sistema pode inferir comportamentos que abrangem a dinâmica do edifício com um, podendo assim integrar as diversas partes do mesmo, resultando assim em um sistema com unidade tanto funcional como organizacional.

Num primeiro momento e de acordo com os elementos presentes na maquete e com as possíveis informações que podem ser extraídas da mesma, tem-se que o sistema supervisório foi dividido em três funções distintas:

- Sistema de supervisão de segurança;
- Sistema de supervisão de conforto térmico;
- Sistema de supervisão de incêndio.

4.1 O Sistema de Supervisão de Segurança

Uma das tarefas que se pode automatizar em um ambiente predial é a atividade de verificação de presença de pessoas. Esta pode estar relacionada com as permissões de permanência de indivíduos em um determinado recinto, podendo ser utilizado para controlar a entrada de pessoas e também para o controle do acendimento de luzes em um ambiente, de acordo com a presença ou não indivíduos.

Neste contexto, o sistema de supervisão de segurança desenvolvido tem como objetivo:

- Verificar a presença de pessoas nos ambientes que compõem a maquete;
- Verificar se a presença de indivíduos num determinado recinto é permitida ou não, baseando-se nos horários de permanência determinados para cada ambiente.

De posse dos dados fornecidos pelo sistema de controle do edifício, o sistema supervisório pode realizar o cruzamento da informação oriunda do sensor de presença no ambiente com o intervalo de horário estabelecido para permitir o acesso ao mesmo. Com isso, o sistema supervisório pode auxiliar o sistema de controle do edifício na tomada de decisão de acender ou não as luzes de um recinto quando for detectada a presença de uma pessoa ou acionar o alarme de invasão para aquela sala.

Além disso, abre-se a possibilidade de se estabelecer políticas diferenciadas para cada ambiente, tendo-se ainda a flexibilidade para alterá-la a qualquer momento. Pode-se também utilizar o histórico contido na base de dados coletado para inferir comportamentos de utilização dos ambientes. Com isso, o sistema supervisório pode assumir um papel mais pró-ativo no controle da presença e suas ações, determinando períodos em que as luzes devem permancer acesas ou verificando comportamentos anormais.

Para poder supervisionar a segurança de cada ambiente, o sistema, neste trabalho, irá necessitar das seguintes informações:

- Presença: relacionada com o sensor de presença que está em cada sala e no elevador;
- Horários permitidos de permanência para cada ambiente: são os horários em que a presença de pessoas é permitida num determinado ambiente. Este horário pode ser determinado pelo usuário da sala ou pelo administrador do sistema através dos controles presentes em cada ambiente, relacionados diretamente com o CLP ou via internet, acessando através de um "browser" a página de administração do prédio.

4.2 O Sistema de Supervisão de Conforto Térmico

Com a crescente utilização de equipamentos de condicionamento da temperatura nos ambientes que compõem um prédio, a administração e o uso adequado de tais sistemas são tarefas cada vez mais importantes e presentes no dia-a-dia do gerenciamento predial. Estas estão também relacionadas à sensação de conforto térmico que os usuários do prédio sentem ao entrarem em contato com os ambientes do mesmo, além de se relacionar com a questão da utilização racional de tais recursos, que estão diretamente ligados à economia de energia e dinheiro.

Tendo visto a sua importância, o sistema de conforto térmico desenvolvido neste trabalho baseou-se nesses aspectos práticos, normalmente encontrados na questão da automatização predial. Para atender tais requisitos, o sistema tem como objetivo:

- Analisar o comportamento da temperatura de cada ambiente;
- Auxiliar na decisão de quando ou não ligar o ventilador presente em cada sala, para a regulação ou manutenção da temperatura ambiente;
- Regular ou manter a temperatura definida pelo usuário ou administrador do ambiente.

Para tanto, o sistema precisa interagir com a maquete, através da aquisição e análise das seguintes informações:

- Temperatura atual da sala: dada pelo sensor de temperatura presente no recinto;
- Temperatura desejada: dada através dos controles presentes na sala ou através do acesso da página de gerenciamento do ambiente, via internet.

O resultado disso é a detecção de que a temperatura num determinado ambiente está dentro do desejado, configurando assim uma ação ou não do sistema de condicionamento de temperatura (nesse caso, o ventilador) para a regulação ou manutenção da temperatura. Caso a temperatura de um ambiente esteja acima do desejado para o mesmo, o sistema supervisório libera um comando para que o sistema de controle ligue o sistema de refrigeração do ambiente. Caso se tenha a situação contrária, o sistema supervisório avisa ao sistema de controle para parar de atuar para que a temperatura se reguralize e chegue ao desejado para o ambiente.

Nesse caso, o sistema de supervisão consegue ter acesso ao valor da temperatura em cada ambiente, podendo proporcionar ao sistema de controle da planta uma visão mais geral do comportamento térmico de um ambiente, andar ou até do edifício.

Através da análise histórica do comportamento das alterações térmicas de cada ambiente, o sistema supervisório pode utilizar essas informações para inferir com maior precisão o estado de um determinado ambiente ou tomar ações antecipadas de controle, visando tornar o processo de conforto térmico o mais autônomo possível.

4.3 O Sistema de Supervisão de Incêndio

Considerando-se um ambiente amplo e com inúmeras variáveis que um sistema predial pode apresentar, a automatização da detecção de incêndio pode facilitar em muito a tarefa de evitar e também acionar contra-medidas adequadas para tal situação. Com a utilização desse tipo de sistema, pode-se adquirir maior velocidade no diagnóstico de situações de início de incêndio, podendo representar assim uma grande vantagem para o gerenciamento do prédio, tendo em mente a segurança de seus ocupantes.

O sistema de supervisão de incêndio aqui abordado tem como foco aproveitar as informações que a planta disponibiliza para analisar e diagnosticar possíveis focos de incêndio. Com isso, pode-se então auxiliar o sistema de controle na tarefa de decidir se há realmente a configuração de uma real situação de incêndio, não confundindo com a situação em que há apenas alguém fumando um charuto.

Para executar essa tarefa de forma a se conseguir um diagnóstico mais preciso possível, o sistema se baseia nas seguintes informações disponibilizadas pelo sistema de controle:

- Temperatura atual da sala: dada pelo sensor de temperatura presente no recinto;
- Temperatura desejada: dada através dos controles presentes na sala ou através do acesso da página de gerenciamento do ambiente, via internet.
- Presença ou não de fumaça: informação fornecida pelo sensor de fumaça.

De posse dessas informações, relacionadas a cada ambiente em análise, o sistema as correlaciona, resultando assim num diagnóstico de uma possível presença ou não de um foco de incêndio. Neste trabalho, utilizou-se como critério para afirmar que há incêndio a presença de fumaça e a variação para mais de 5° C em relação à temperatura desejada.

Num passo seguinte, pode-se utilizar o histórico da temperatura para analisar a rampa de subida da temperatura em um ambiente. Com isso, utlizando-se em conjunto o sistema de conforto térmico, pode-se verificar a natureza dessa fonte de calor. Caso a atuação do sistema de condicionamento de ar não faça com que a temperatura do ambiente diminua e também haja a presença de fumaça, isso pode significar que haja um incêndio. Através da atuação conjunta desses dois sistemas através da coordenação do sistema de supervisão, pode-se auxiliar o sistema de controle a diagnosticar melhor cada situação e atuar de uma forma mais apropriada.

5 Modelagem do Sistema Supervisório

5.1 Considerações sobre a Modelagem do Sistema

A modelagem envolvida neste projeto visa representar um sistema de informação e supervisão que desempenhe as seguintes funções:

1. Segurança;
2. Controle de presença;
3. Monitoramento da temperatura;
4. Alarme de incêndio.

A modelagem de cada sistema requer que haja um fluxo de informações entre os elementos que o compõem, de modo a se obter cooperação mútua entra as ações e análise da situação do sistema. Isso evidencia a importância de se obter um modelo que combine a representação comportamental e funcional do sistema. Estes conceitos podem ser ilustrados da seguinte forma: para se controlar a iluminação e o controle de acesso de pessoas em um prédio, necessita-se de dois sistemas (comportamental), um para cada caso. Mas, funcionalmente, ambos poderiam ser implementados pelo mesmo sistema, através do monitoramento de sensores de presença. No primeiro caso, a constatação da ausência de pessoas em um ambiente faz com que o sistema apague as luzes desse aposento. Já no segundo caso, a leitura de presença de pessoa em um recinto pode apontar a quebra na segurança desse setor.

Além disso, a modelagem deve prever a interação entre o sistema e os usuários, uma vez que estes são o principal motivo de sua existência. Isso pode acabar introduzindo certas

particularidades em cada sistema, quebrando assim a unidade e a modularidade que inicialmente possa ter se pensado. Isso pode ser ilustrado na situação em que existem salas com diferentes controles de acesso, como por exemplo, devido a turnos diferentes de trabalho. Com isso, introduz-se o fator de tempo, podendo haver situações em que em uma sala a presença de alguém às 23:30 não configure uma situação de invasão, mas sim de trabalho.

Com isso, mostra-se a importância de se ter um sistema com um modelo que suporte e agregue todas essas funcionalidades e possibilidades, evitando redundâncias de informações e interpretações errôneas sobre o sistema. Descreve-se assim o papel do sistema de informação a ser desenvolvido neste projeto, com o intuito de fornecer e melhorar o desempenho na atuação e monitoramento da planta, que no caso, é um prédio. Além disso, há a introdução de novas funcionalidades, tais como um histórico dos acontecimentos na planta, sendo útil para uma futura análise dos eventos e suas causas, fornecendo elementos que podem ajudar no melhor entendimento do sistema e, consequentemente, na evolução do mesmo.

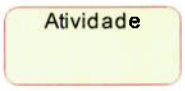
Posto a importância da correta modelagem de um sistema para o seu controle, o projeto segue com a pesquisa e aprendizado de ferramentas úteis para tal finalidade, tais como a utilização de design orientada a objetos, análise comportamental e funcional e ferramentas como UML.

5.2 Diagramas de Atividades

Para representar a dinâmica de um sistema, utilizou-se o diagrama de atividade. Os diagramas de atividade são cartas de fluxo que tem o papel de representar o fluxo de trabalho de um determinado sistema, exibindo tal fluxo da própria atividade e aquelas que serão realizadas em paralelo.

Os diagramas de atividade contêm os seguintes elementos:

1) Atividade:



Representa uma determinada atividade dentro do fluxo de trabalho do sistema.

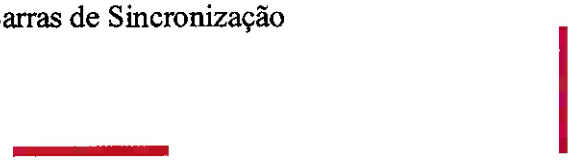
2) Transição:

Representa a orientação da passagem de fluxo de uma atividade para outra.

3) Decisão:

Representa um ponto de decisão em um fluxo de trabalho.

4) Barras de Sincronização



Barra de SincronizaçãoHorizontal Barra de Sincronização Vertical

Representam pontos de sincronização de atividades que ocorrem em paralelo.

5.3 Diagrama de Atividade do Sistema de Supervisão de Incêndio

Seu objetivo é a análise das informações providas pelos sensores de fumaça e temperatura com o intuito de definir e detectar uma situação de incêndio. Para tanto, o sistema primeiro verifica se há a presença de fumaça. Caso haja, então se verifica se a temperatura atual está 5 graus acima da temperatura de referência. Esse critério foi utilizado baseando-se no conhecimento de que o sistema de conforto térmico estará atuando para que a temperatura atual se aproxime da desejada. Caso haja um foco de incêndio, o sistema de conforto térmico não conseguirá chegar à temperatura esperada.

Segue-se então o diagrama de atividades que representa a lógica utilizada para determinar se há incêndio ou não em um determinado ambiente.

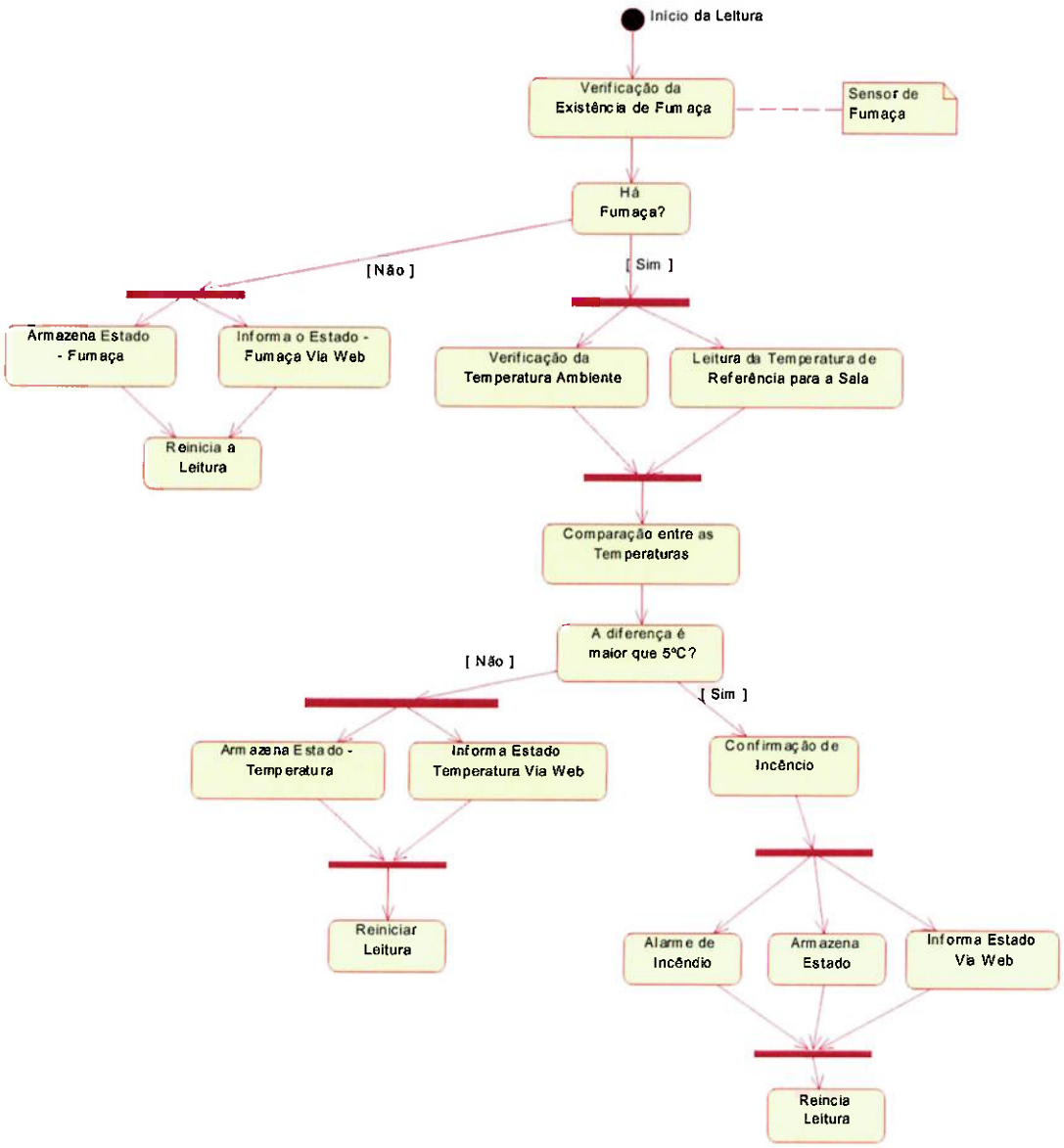


Figura 2 – Diagrama de Atividades do Sistema de Supervisão de Incêndio

5.4 Diagrama de Atividade do Sistema de Supervisão de Conforto Térmico

Seu objetivo é informar e auxiliar na adequação da temperatura nas salas do prédio. Este está diretamente relacionado com o sistema de condicionamento de ambientes e indiretamente ligado ao sistema de incêndio através do sistema de supervisão de incêndio.

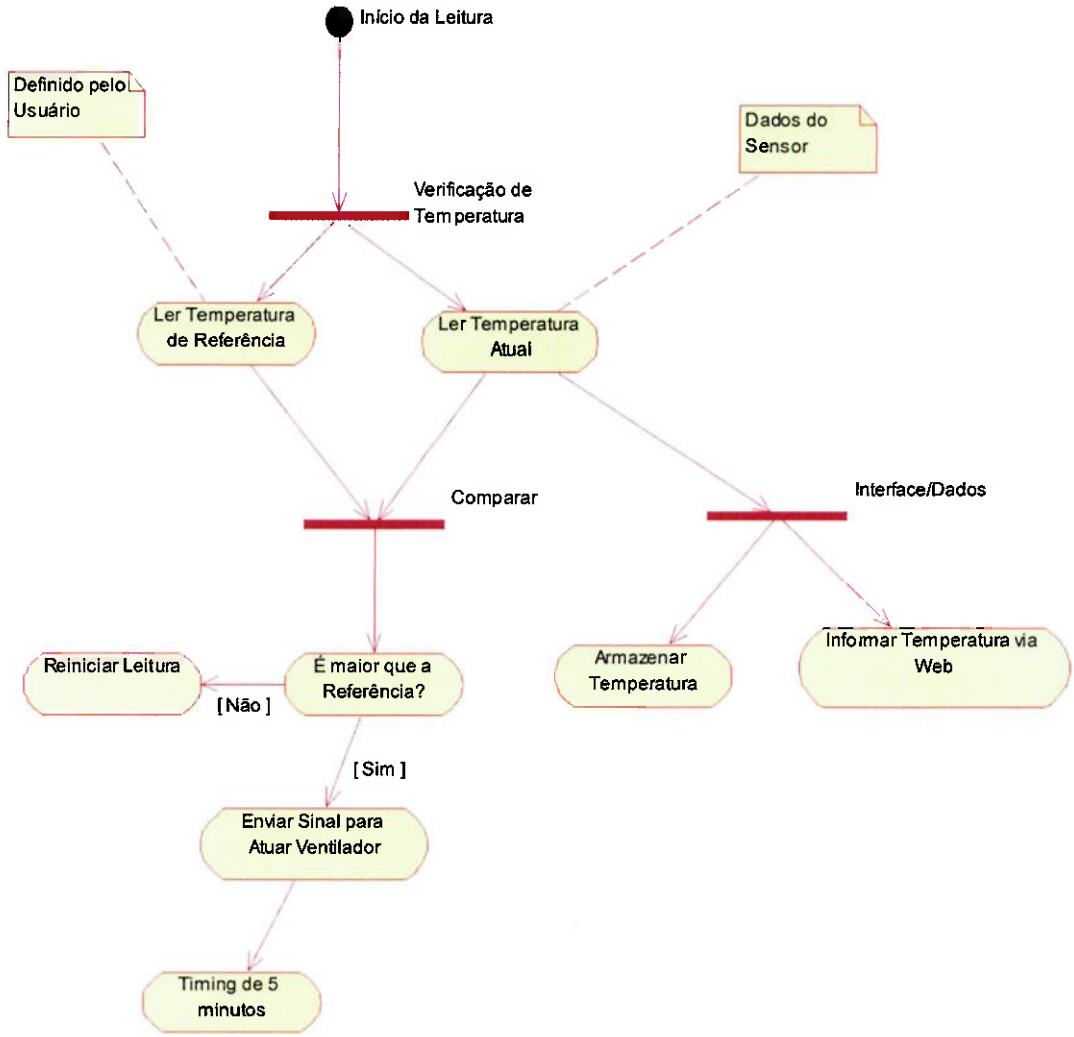


Figura 3 – Diagrama de Atividades do Sistema de Supervisão de Temperatura

A análise do conforto térmico é feita através da comparação da temperatura atual do ambiente, fornecida pelo seu respectivo sensor de temperatura, com a temperatura desejada da mesma. Com isso, o sistema retorna se a temperatura está abaixo, de acordo ou acima em relação à temperatura desejada.

5.5 Diagrama de Atividade do Sistema de Supervisão de Presença

Seu objetivo é verificar a presença de pessoas nos ambientes que fazem parte da estrutura predial. Este está diretamente relacionado com o controle de luz nas salas e de segurança no prédio. Indiretamente, o sistema de supervisão de incêndio poderia utilizá-lo em situações de emergência, como por exemplo em incêndio, de modo a traçar a melhor estratégia de evacuação e os melhores comandos para cada elemento do sistema. Um exemplo disso é o controle do elevador em uma situação de incêndio: caso haja ocupantes nele, o sistema central deverá informar ao sistema de controle do elevador que execute a estratégia de emergência para a evacuação de seus ocupantes, que é a parada no andar mais próximo e longe da região com problemas, sendo que em seguida deve-se travar o elevador, parando o seu funcionamento.

A verificação da segurança em um determinado ambiente é feita através da comparação entre o horário permitido de presença e o horário em que a presença de pessoas é detectada. Segue-se então o diagrama de atividades relacionada com a dinâmica do sistema de supervisão de segurança.

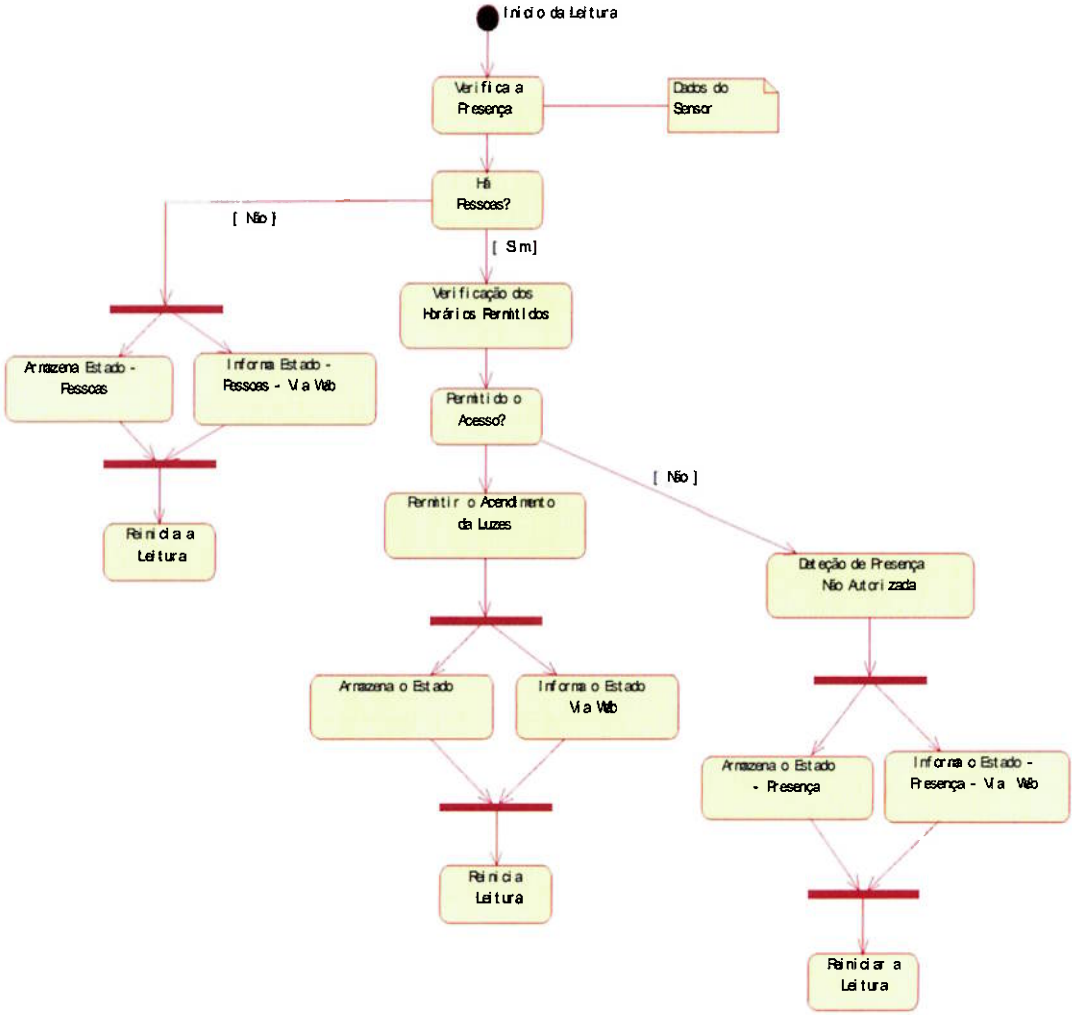


Figura 4 – Diagrama de Atividades do Sistema de Supervisão de Presença

5.6 Modelagem dos Objetos do Sistema

Baseando-se na teoria de modelagem por objetos, pode-se identificar três grandes classes que descrevem o sistema a ser desenvolvido. São elas:

- Super Classe *"Ambient"*: esta classe é utilizada para gerar e descrever os ambientes existentes na maquete, resultando assim, neste caso, em três classes:
  - Classe *"Hall"*: esta classe representa o hall de entrada da maquete (piso térreo);
  - Classe *"Elevator"*: esta classe representa o elevador existente na maquete;
  - Classe *"Room"*: esta classe representa as salas existentes na maquete.
- Classe *"DBInterface"*: classe que descreve a interação dos demais objetos com o banco de dados da maquete;
- Classe *"Supervisor"*: classe que representa o sistema supervisório da maquete, a qual contém os métodos relacionados com a supervisão de segurança, conforto térmico e incêndio.

5.6.1 Super Classe "Ambient"

Esta classe representa os diversos ambientes possíveis na maquete. Seus atributos são:

- *"EnvID"(Environment Identification)*: número que identifica univocamente um ambiente da maquete. Os números adotados foram:
  - 1: para representar o hall de entrada;
  - 2: para representar o elevador;
  - 11: para representar a sala 1 do primeiro andar;

- 12: para representar a sala 2 do primeiro andar;
- 21: para representar a sala 1 do segundo andar;
- 22: para representar a sala 2 do segundo andar.

- *"Location"*: campo que descreve a localização do ambiente na maquete: sala, elevador ou hall.

Os métodos dessa classe são:

- *"GetID"*: retorna o número de identificação do objeto;
- *"GetLocation"*: retorna a localização do ambiente na maquete.

Derivando-se da super classe *"Ambient"*, tem-se duas classes, as quais herdam os métodos e os atributos desta primeira:

- Classe *"CRoom"*: classe que representa as salas existentes na maquete. Seus atributos são herdados da super classe *"Ambient"*. Seus métodos são:
  - *"GetLastTemp"*: retorna a temperatura atual da respectiva sala;
  - *"GetRefferenceTemp"*: retorna a temperatura desejada (de referência) da respectiva sala;
  - *"GetLastSmoke"*: retorna se há ou não presença de fumaça na respectiva sala;
  - *"GetLastPresence"*: retorna se há ou não pessoas na respectiva sala;
  - *"GetInitialHour"*: retorna a hora inicial que é permitida a presença na respectiva sala;
  - *"GetFinalHour"*: retorna a hora final que é permitida a presença na respectiva sala.

- Classe *"CElevator"*: classe que representa o elevador existente na maquete. Seus atributos são herdados da super classe *"Ambient"*. Seus métodos são:
    - *"GetLastPresence"*: retorna se há ou não pessoas no elevador;
    - *"GetInitialHour"*: retorna a hora inicial que é permitida a presença no elevador;
    - *"GetFinalHour"*: retorna a hora final que é permitida a presença no elevador.

5.6.2 Classe "DBInterface"

Esta classe representa a interação entre os diversos objetos do sistema com o banco de dados da maquete. Sua função é fazer a conecção com a base de dados e os objetos que necessitam das informações presentes nele para realizar suas atividades.Esta interação é representada pelo fluxo dos seguintes dados:

- Temperatura atual do ambiente;
- Temperatura desejada (de referência) para o ambiente;
- Presença ou não de fumaça no ambiente;
- Presença ou não de pessoas no ambiente;
- Horários permitidos de presença no ambiente;
- Presença ou não de incêndio no ambiente;
- Presença ou não de intrusos no ambiente;
- Diferença ou não da temperatura atual com a desejada.

Para realizar a interface entre o banco de dados e os objetos, são utilizados os seguintes métodos:

- "TimePresenceSensor": retorna o horário em que a leitura da presença foi feita no ambiente;
- *"SmokeSensor"*: retorna 1 se o sensor de fumaça estiver funcionando e 0 caso contrário. A verificação é baseada na consistência entre a data da leitura do sensor e a data em que a consulta é feita, com a finalidade de se ter certeza que o dado presente no banco de dados é o mais atual possível;
- *"PresenceSensor"*: retorna 1 se o sensor de fumaça estiver funcionando e 0 caso contrário. A verificação é baseada na consistência entre a data da leitura do sensor e a data em que a consulta é feita, com a finalidade de se ter certeza que o dado presente no banco de dados é o mais atual possível;
- *"TempSensor"*: retorna 1 se o sensor de fumaça estiver funcionando e 0 caso contrário. A verificação é baseada na consistência entre a data da leitura do sensor e a data em que a consulta é feita, com a finalidade de se ter certeza que o dado presente no banco de dados é o mais atual possível;
- *"GetAmbientLastTemperature"*: retorna a última leitura do sensor de temperatura do ambiente;
- *"GetAmbientLastPresence"*: retorna a última leitura do sensor de presença do ambiente;
- *"GetAmbientLastSmoke"*: retorna a última leitura do sensor de fumaça do ambiente;
- *"GetAmbientReferenceTemperature"*: retorna a temperatura de referênicia ou desejada para o ambiente;
- *"GetAmbientInicialTimePermission"*: retorna o horário inicial de presença permitida para o ambiente;

- *"GetAmbientFinalTimePermission"*: retorna o horário final de presença permitida para o ambiente;
- *"SetAmbientFireResult"*: insere no banco de dados o resultado da análise do sistema de supervisão de incêndio;
- *"SetAmbientSecurityResult"*: insere no banco de dados o resultado da análise do sistema de supervisão de segurança;
- *"SetAmbientComfortResult"*: insere no banco de dados o resultado da análise do sistema de supervisão de conforto térmico;
- *"Simulator"*: objeto usado para simular o sistema Supervisório enquanto não há a integração com a bancada de teste. Retorna 1 se for para simular, 0 para parar o programa e 33 caso não haja comando no momento da leitura do BD, significando que não se deve simular e nem parar o programa. O comando para simulação é dado através de uma página em HTML.

5.6.3 Classe "Supervisor"

Esta classe implementa o sistema de supervisório, realizando as tarefas de supervisão de segurança, de conforto térmico e de incêndio. Para tanto, os métodos implementados são:

- *"run_security"*: supervisão de segurança: retorna 1 caso haja presença não permitida e 0 caso contrário;
- *"run_comfort"*: supervisão de conforto térmico: retorna -1 caso a temperatura atual esteja abaixo da deseja, 0 caso a temperatura atual esteja de acordo ou 1 caso a temperatura atual esteja acima da desejada;
- *"run_fire"*: supervisão de incêndio: retorna 1 caso haja incêndio e 0 caso contrário.

6 Modelagem do Fluxo de Informações

Como visto anteriormente, a maquete gera inúmeras informações através de seus diversos elementos. Toda a informação gerada é centralizada em um único banco de dados, o qual é responsável por relacionar o sistema supervisório com o sistema de informação da maquete. Além disso, o banco de dados fornece informações para o servidor de internet, para que as interações com o usuário (consulta e determinação de parâmetros) através de um "browser" possam ser realizadas. Abaixo segue-se um esquema geral do fluxo de informações entre os diversos sistemas que interagem com a maquete.

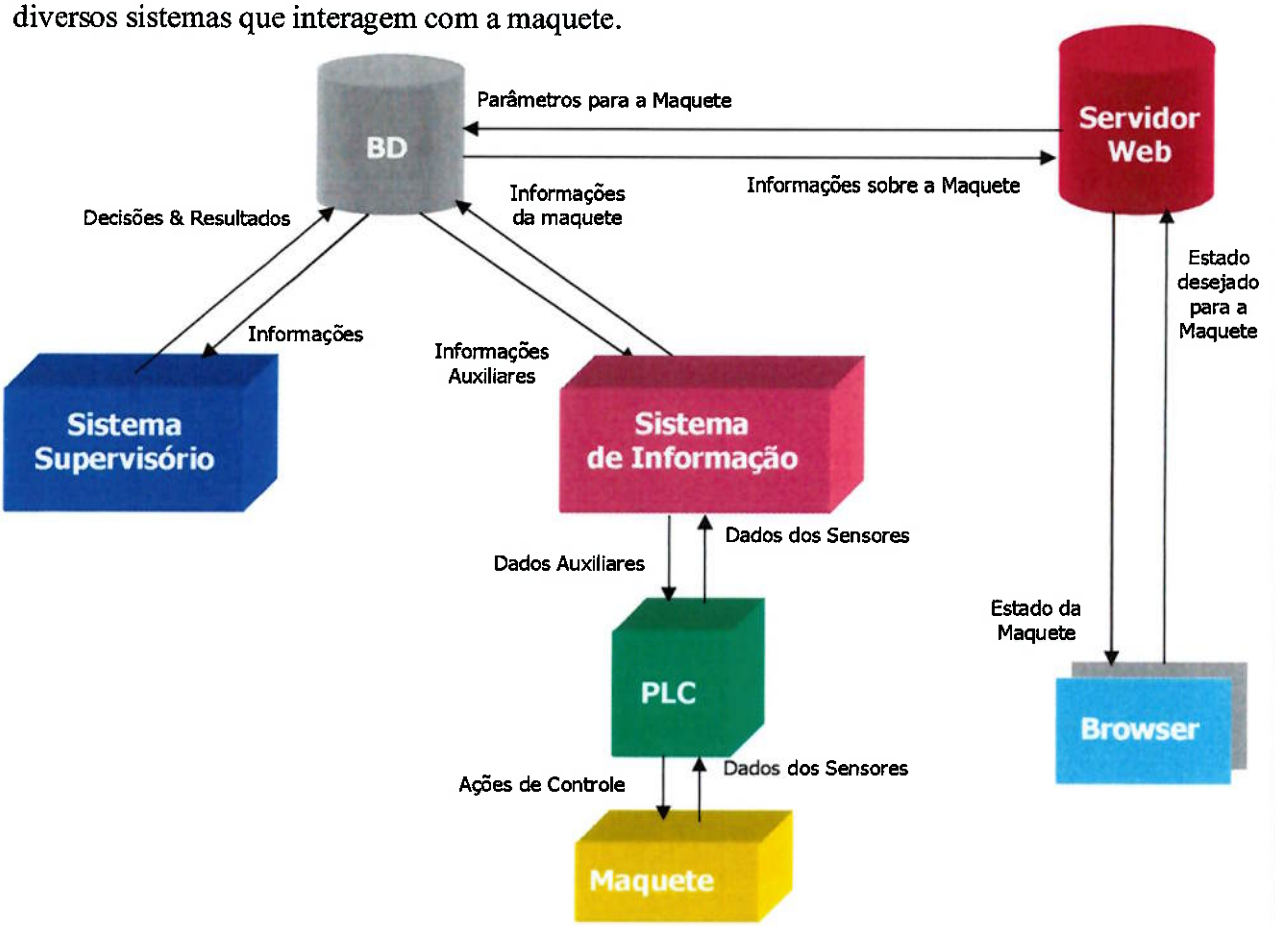


Figura 5 – Fluxograma das informações entre os sistemas que interagem com a maquete.

6.1 Fluxo de Informações

As informações com que o sistema interage dependem tanto da sua origem (sensor, ambiente), quanto de quem a requisita (ambiente, sistema de supervisão). Além disso, cada informação está também atrelada à hora e dia em que esta foi coletada, refletindo a característica do sistema de supervisão de ter que tomar decisões quase que instantâneas, uma vez que se está analisando eventos discretos no tempo e que podem acontecer a qualquer momento.

Tendo em mente isso, abaixo segue-se então a descrição do fluxo de informações entre os diferentes objetos presentes no sistema:

1. Sala:
   - Informações próprias:
     - Número de identificação da sala (*"EnvID"*);
     - Localização da sala (*"Location"*).
   - Informações captadas:
     - Temperatura de referência: é a temperatura que um usuário, por exemplo, determina como ideal para esta sala;
     - Temperatura atual: é a leitura fornecida pelo sensor de temperatura presente nesta sala;
     - Horário de presença permitida: horário definido pelo setor de segurança, por exemplo, no qual a presença de pessoas na sala é permitida;
     - Presença: é a leitura fornecida pelo sensor de presença da sala indicando a presença ou não de pessoas;

- Fumaça: é a leitura fornecida pelo sensor de fumaça da sala indicando a presença ou não de fumaça.

2. Elevador:
   - Informações próprias:
     - Número de identificação da sala (*"EnvID"*);
     - Localização do elevador (*"Location"*).
   - Informações captadas:
     - Presença: é a leitura fornecida pelo sensor de presença do elevador indicando a presença ou não de pessoas;
     - Horário de presença permitida: horário definido pelo setor de segurança, por exemplo, no qual a presença de pessoas na sala é permitida.
3. Hall:
   - Informações próprias:
     - Número de identificação da sala (*"EnvID"*);
     - Localização do hall (*"Location"*).
4. Sistemas de Supervisão de Segurança:
   - Informações necessárias:
     - Horário de presença permitida: horário definido pelo setor de segurança, por exemplo, no qual a presença de pessoas no ambiente é permitida;
     - Presença: é a leitura fornecida pelo sensor de presença do ambiente indicando a presença ou não de pessoas.
   - Informações enviadas:

- Resultados: são sinais resultados da supervisão feita pelo sistema. Envia 1 para o caso de presença não autorizada e 0 caso contrário.

5. Sistemas de Supervisão de Conforto Térmico:
   - Informações necessárias:
     - Temperatura de referência: é a temperatura que um usuário, por exemplo, determina como ideal para o ambiente;
     - Temperatura atual: é a leitura fornecida pelo sensor de temperatura presente no ambiente.
   - Informações enviadas:
     - Resultados: são sinais resultados da supervisão feita pelo sistema. Envia -1 para temperatura abaixo do ideal, 0 quando estiver de acordo e 1 quando estiver acima.
6. Sistemas de Supervisão de Incêndio:
   - Informações necessárias:
     - Temperatura de referência: é a temperatura que um usuário, por exemplo, determina como ideal para esta sala;
     - Temperatura atual: é a leitura fornecida pelo sensor de temperatura presente nesta sala;
     - Fumaça: é a leitura fornecida pelo sensor de fumaça da sala indicando a presença ou não de fumaça.
   - Informações enviadas:
     - Resultados: são sinais resultados da supervisão feita pelo sistema. Envia 1 para incêndio e 0 caso contrário.

7. Interface com banco de dados:
   - Informações enviadas/recebidas:
     - Temperatura de referência;
     - Temperatura atual;
     - Horário de presença permitida;
     - Presença;
     - Fumaça;
     - Resultados dos sistemas de supervisão.
8. Simulador:
   - Informações enviadas:
     - Temperatura de referência/desejada para cada sala;
     - Horário de presença permitida para cada ambiente;
     - Temperatura atual de cada sala;
     - Presença e a hora em que a mesma acontece em cada ambiente;
     - Fumaça e a hora em que a mesma acontece em cada ambiente.
   - Informações recebidas:
     - Alarme de incêndio de cada sala;
     - Alarme de segurança/invasão para cada ambiente;
     - Indicação do conforto térmico de cada sala;
     - Valores dos sensores de fumaça, temperatura e presença de cada ambiente;
     - Status (funcionando ou não) dos sensores de fumaça, temperatura e presença de cada ambiente.

Segue abaixo o esquema mostrando o sentido do fluxo de informações para cada objeto do sistema:

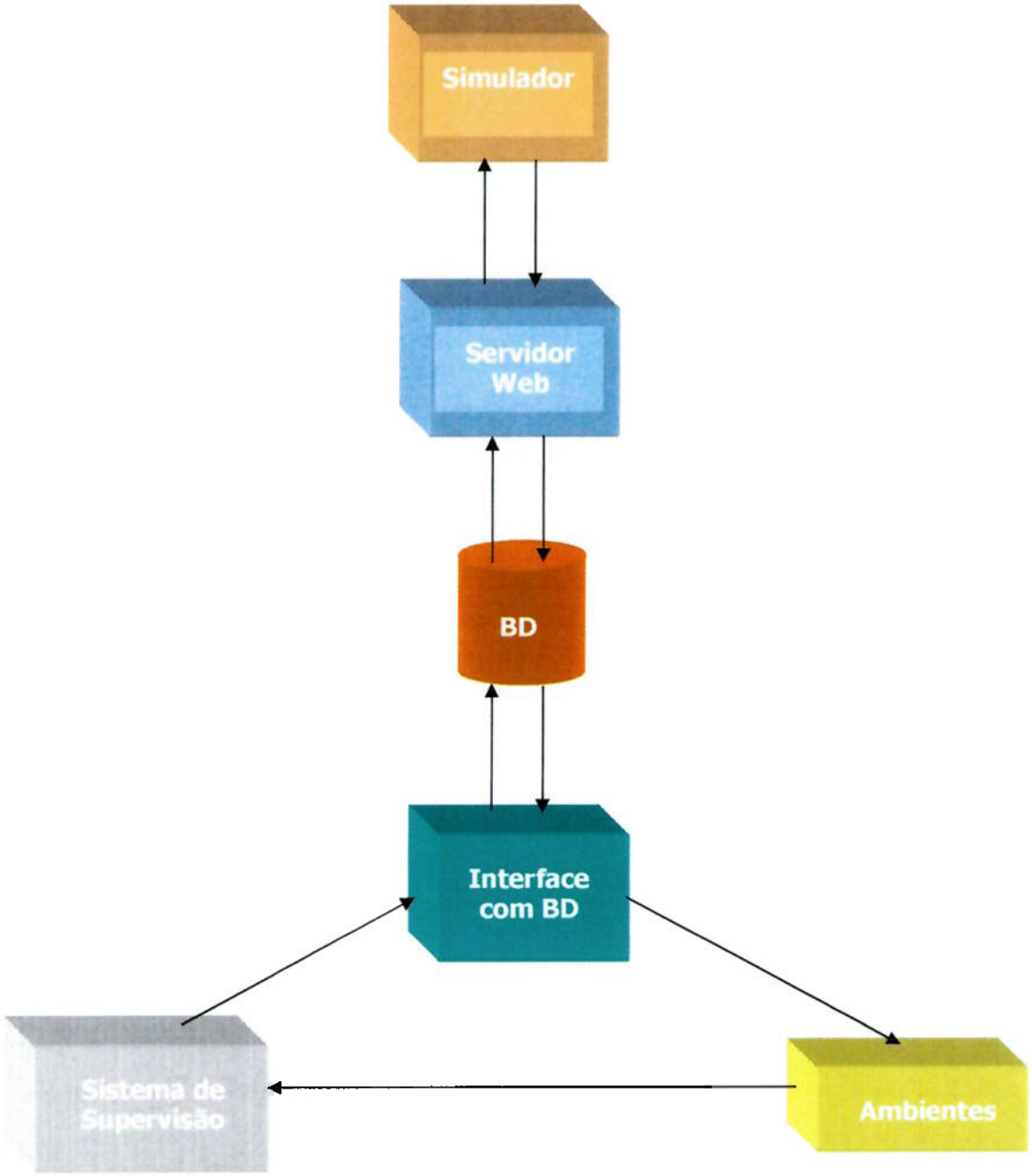


Figura 6 – Sentido do fluxo das informações trocadas entre os objetos.

6.2 Banco de Dados

Para armazenar todas as informações geradas pela maquete e pelos diversos sistemas, utilizou-se o banco de dados Microsoft Access. Neste, foram implementados as seguintes tabelas:

1. *"Environment"*: tabela que guarda as informações dos ambientes existentes na maquete. Possui os seguintes campos:
    a. *"EnvID"*: guarda o número de identificação único do ambiente (Ex. elevador = 2);
    b. *"Type"*: guarda a descrição do tipo de ambiente (Ex. 1 = hall);
    c. *"Floor"*: guarda o andar em que o ambiente se localiza(Ex. hall = andar térreo);
    d. *"Location"*: guarda a descrição da localização do ambiente (Ex. sala 21 = primeira sala do segundo andar).

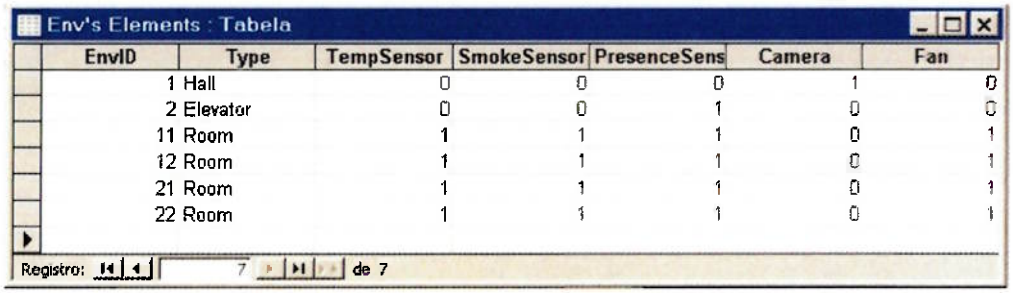
Environment : Tabela

EnvID	Type	Floor	Location
1	Hall	Térreo	
2	Elevator	Mobile	
11	Room	First	Room_1
12	Room	First	Room_2
21	Room	Second	Room_1
22	Room	Second	Room_2

Registro: 7 de 7

Figura 7 – Tabela Environment.

2. *"Env's Elements"*: tabela que guarda os elementos presentes em cada ambiente. Possui os seguintes campos:
    a. *"EnvID"*: guarda o número de identificação único do ambiente;
    b. *"Type"*: guarda a descrição do tipo de ambiente;
    c. *"TempSensor"*: indica a presença ou não do sensor de temperatura no ambiente;
    d. *"SmokeSensor"*: indica a presença ou não do sensor de fumaça no ambiente;
    e. *"PresenceSensor"*: indica a presença ou não do sensor de presença no ambiente;
    f. *"Camera"*: indica a presença ou não de uma câmera;
    g. *"Fan"*: indica a presença ou não de um ventilador.


Env's Elements : Tabela

EnvID	Type	TempSensor	SmokeSensor	PresenceSens	Camera	Fan
1	Hall	0	0	0	1	0
2	Elevator	0	0	1	0	0
11	Room	1	1	1	0	1
12	Room	1	1	1	0	1
21	Room	1	1	1	0	1
22	Room	1	1	1	0	1

Registro: 7 de 7

Figura 8 – Tabela Env's Elements.

3. *"PresenceSensor"*: tabela que guarda os valores referentes ao sensor de presença. Para tanto, possui os seguintes campos:
    a. *"EnvID"*: guarda o número de identificação único do ambiente;
    b. *"Presence"*: guarda a leitura referente ao sensor de presença (1 para presença de pessoas e 0 caso contrário);
    c. *"Date"*: guarda a data em que o dado foi inserido no BD;

d. *"Time"*: guarda a hora em que o dado foi inserido no BD.

PresenceSensor : Tabela

EnvID	Presence	Date	Time
2	1	28/11/02	21:30:05
11	1	28/11/02	15:47:59
12	0	28/11/02	19:16:04
21	0	28/11/02	19:16:04
22	1	28/11/02	12:29:59

Registro: 6 de 6

Figura 9 – Tabela PresenceSensor.

4. *"SmokeSensor"*: tabela que guarda os valores referentes ao sensor de fumaça. Para tanto, possui os seguintes campos:

   a. *"EnvID"*: guarda o número de identificação único do ambiente;

   b. *"Smoke"*: guarda a leitura referente ao sensor de fumaça (1 para presença de fumaça e 0 caso contrário);

   c. *"Date"*: guarda a data em que o dado foi inserido no BD;

   d. *"Time"*: guarda a hora em que o dado foi inserido no BD

SmokeSensor : Tabela

EnvID	Smoke	Date	Time
11	0	28/11/02	19:16:03
12	0	28/11/02	19:16:04
21	0	28/11/02	19:16:04
22	0	28/11/02	19:16:04

Registro: 5 de 5

. Figura 10 – Tabela Smoke Sensor.

5. *"TempSensor"*: tabela que guarda os valores referentes ao sensor de temperatura. Para tanto, possui os seguintes campos:
    a. *"EnvID"*: guarda o número de identificação único do ambiente;
    b. *"Smoke"*: guarda a leitura referente ao sensor de temperatura ;
    c. *"Date"*: guarda a data em que o dado foi inserido no BD;
    d. *"Time"*: guarda a hora em que o dado foi inserido no BD.

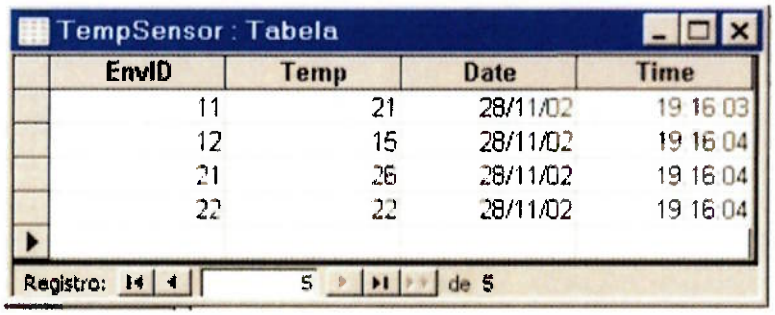


Figura 11 – Tabela TempSensor.

6. *"UserInterface"*: tabela que guarda os valores que o usuário determina como parâmetro para um ambiente. Para tanto, possui os seguintes campos:
    a. *"EnvID"*: guarda o número de identificação único do ambiente;
    b. *"RefTemp"*: guarda a temperatura desejada pelo usuário para um determinado ambiente;
    c. *"InicialPermTime"*: guarda o horário inicial de permanência em um determinado ambiente;
    d. *"EndPermTime"*: guarda o horário final de permanência em um determinado ambiente;
    e. *"Date"*: guarda a data em que o dado foi inserido no BD;

f. *"TTime"*: guarda a hora em que o dado foi inserido no BD.

g. *"DDate"*: guarda a data em que o dado foi inserido no BD.

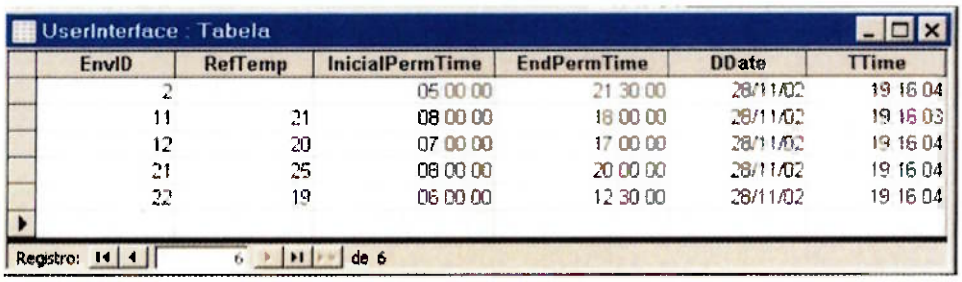


Figura 12 – Tabela UserInterface.

7. *"Results"*: tabela que guarda os resultados relacionados aos sistemas de supervisão de segurança, conforto térmico e de incêndio. Seus respectivos campos são:

   a. *"EnvID"*: guarda o número de identificação único do ambiente;

   b. *"FireAlarm"*: guarda o resultado da análise do sistema de incêndio (1 para presença de incêndio, 0 caso contrário e 33 para indicar mau funcionamento do sensor de temperatura e/ou sensor de fumaça);

   c. *"IntrusionAlarm"*: guarda o resultado da análise do sistema de segurança (1 para presença de intrusos, 0 caso contrário e 33 para indicar mau funcionamento do sensor de presença);

   d. *"TempDifference"*: guarda o resultado da análise do sistema de conforto térmico (-1 para temperatura abaixo da deseja, 0 caso a temperatura esteja de acordo, 1 caso esteja acima e 33 para indicar mau funcionamento do sensor de temperatura);

   e. *"DDate"*: guarda a data em que o dado foi inserido no BD;

   f. *"TTime"*: guarda a hora em que o dado foi inserido no BD.

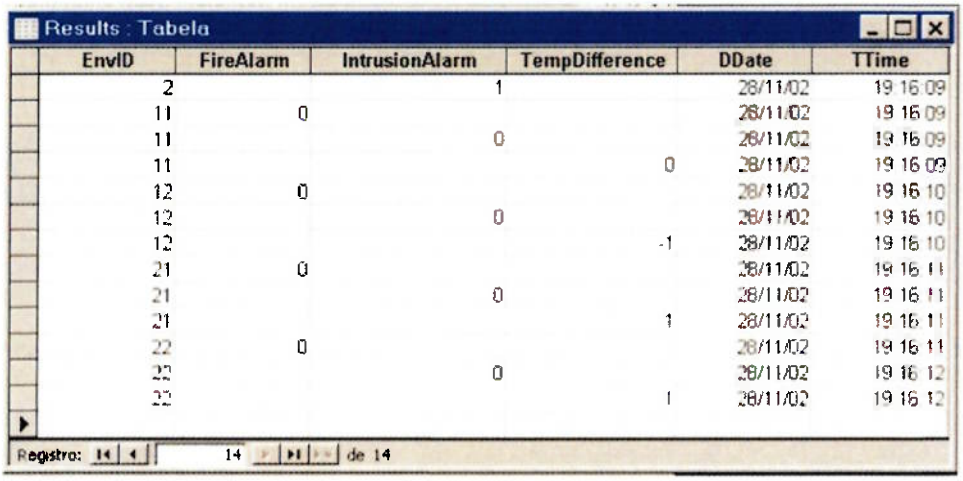


Figura 13 – Tabela Results.

8. *"Simulator"*: a tabela que guarda a ordem de simular ou parar o programa enviada pelo Web Browser (interface em HTML). Seus respectivos campos são:
   a. *"Simulate"*: guarda o número que identifica a ordem de simulação: '1' para simular e '0' para parar o programa;
   b. *"Date"*: guarda a data em que o dado foi inserido no BD;
   c. *"Time"*: guarda a hora em que o dado foi inserido no BD.

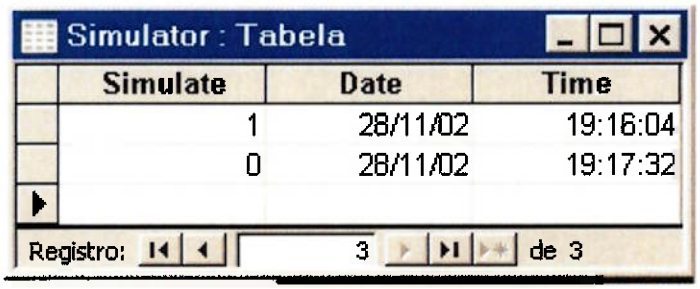


Figura 14 – Tabela Simulator.

Segue abaixo o relacionamento entre as tabelas:

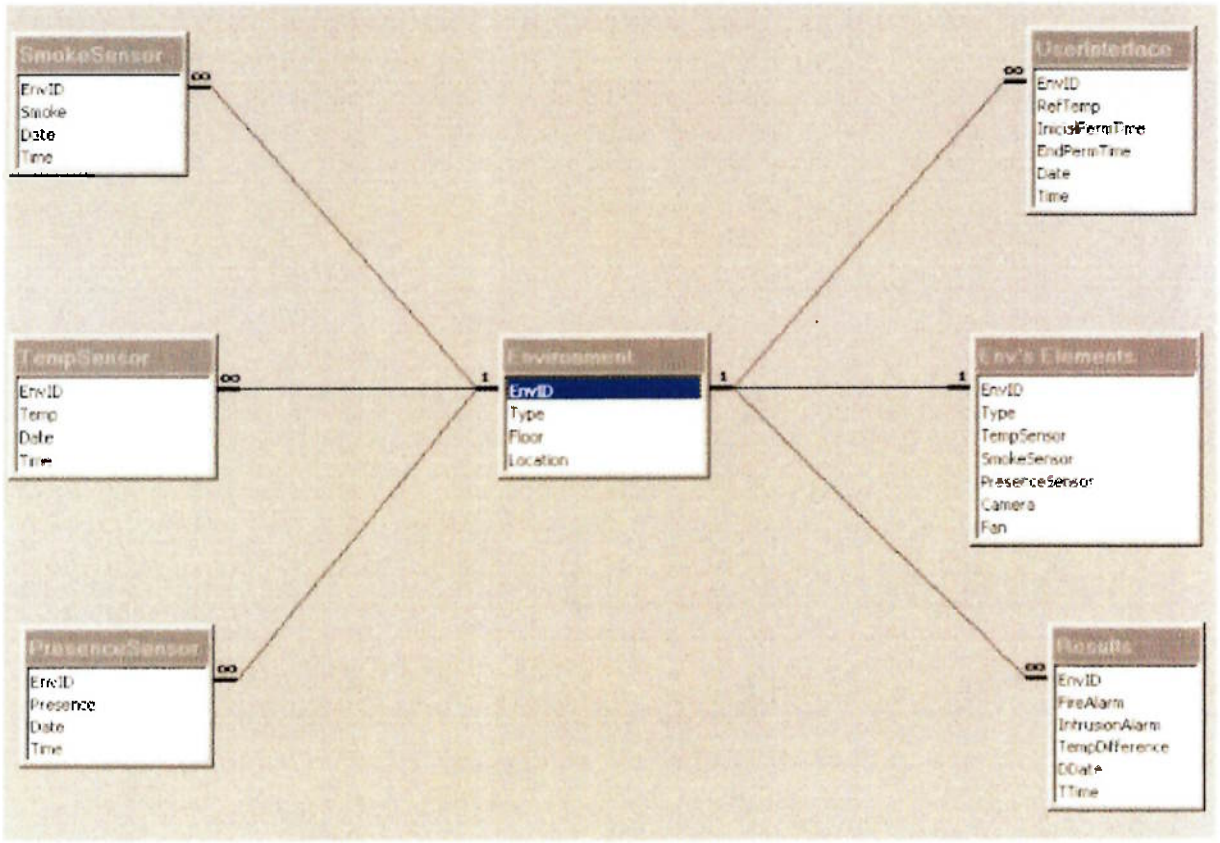


Figura 15 – Relacionamento entre as tabelas do sistema.

7 O Programa

O programa desenvolvido foi baseado na linguagem JAVA da SUN, sendo usado o ambiente JBUILDER 5, versão Personal, da Borland. Como banco de dados, foi utilizado o Microsoft Access, sendo a linguagem SQL usada para fazer tanto o acesso como as alterações no mesmo.

Para implementar o sistema de supervisão da maquete, foram criadas as classes de acordo com a modelagem dos objetos necessários, visto no item 5.6. A partir destes objetos, o programa principal foi desenvolvido baseando-se no princípio de que o sistema supervisório com um todo estaria atuando em um ambiente de cada vez. Ou seja, as supervisões de incêndio, segurança e conforto térmico são realizadas primeiro, por exemplo, na sala 11 para depois ser feita a verificação na sala 12.

Uma outra possibilidade de se realizar a supervisão da maquete seria tornar a tarefa de verificação de incêndio, segurança e conforto térmico independentes entre si, ou seja, desenvolver sistemas que fossem executados em paralelo. Este caso poderia refletir uma situação em que há vários servidores rodando em paralelo para executar essas diferentes tarefas, o que poderia resultar numa maior velocidade e segurança para o processo como um todo. Entretanto, uma implementação desse tipo foge do escopo do trabalho, pois tornaria muito complicado o seu desenvolvimento.

A listagem do programa desenvolvido se encontra no Anexo A. A seguir, tem-se um diagrama representando o fluxo do programa principal.

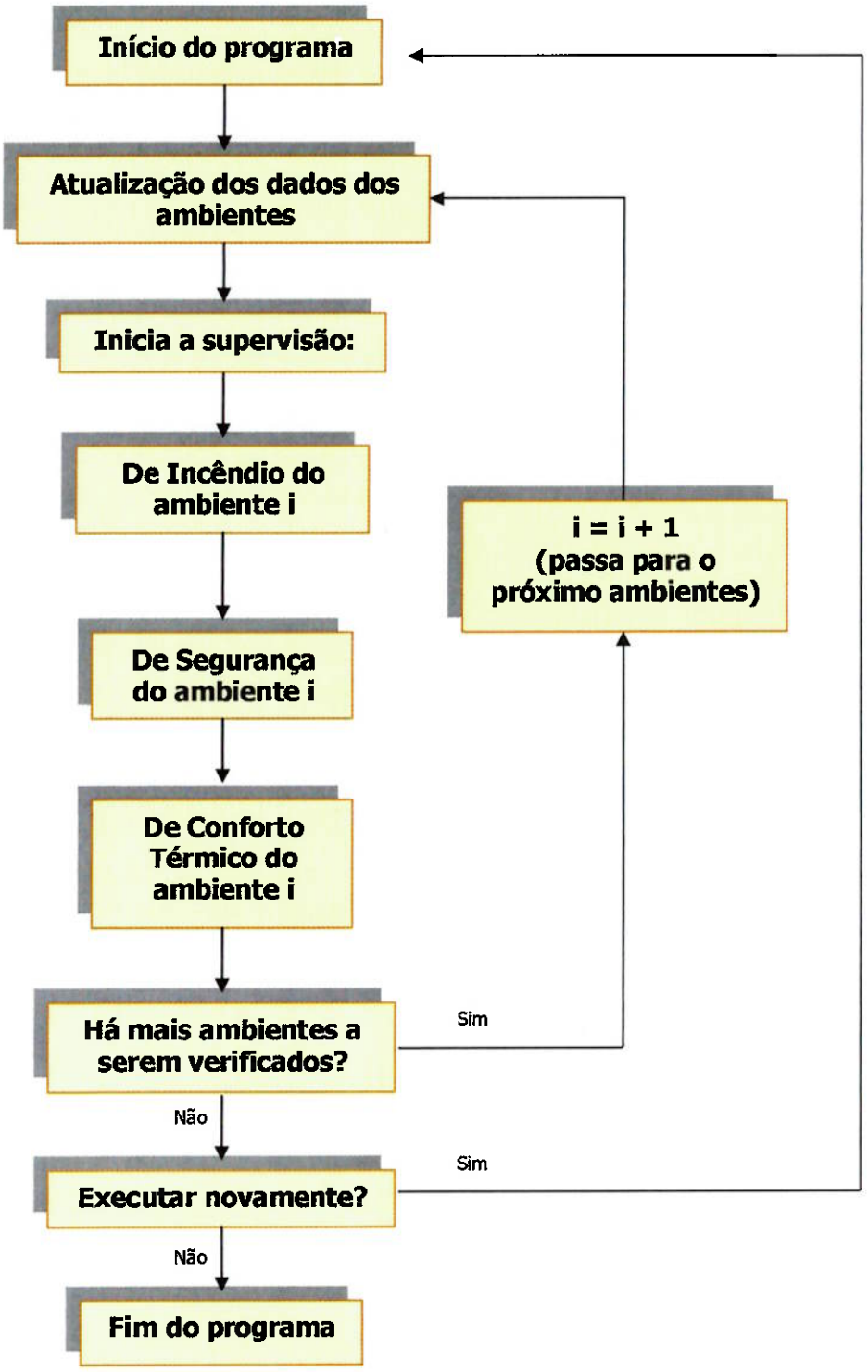


Figura 16 – Fluxograma do programa principal.

8 Simulação e Apresentação de Resultados

Para que se pudesse simular e testar o sistema de supervisão, foi desenvolvida uma interface para simular o programa. A inserção dos dados utilizados na simulação, bem como a apresentação dos resultados do programa fazem uso de uma interface gráfica em HTML, visualiazada e manipulada através de um Web Browser.

Para tanto, foi utilizada a tecnologia de *Servlets Java* para simular a interação do sistema supervisório através da Internet. Utilizou-se o servidor contido no kit de desenvolvimento de servlets em Java (JSDK 1.2). É a partir dele que se processa toda a interação, inserção de dados e recebimento dos resultados para as entradas dos dados da simulação (estados dos ambientes). O banco de dados é usado como interface de troca de dados entre o servidor da Web e o programa Supervisório.

O simulador poderá ser adaptado no momento em que se integrar o sistema supervisório à bancada de teste. Nesse momento, poderá se visualizar os resultados da supervisão da maquete, podendo acessar o status de cada ambiente (segurança, incêndio, conforto térmico, sensores).

A seguir tem-se um fluxo representativo das etapas realizadas durante a simulação, bem como a interação entre a interface gráfica (Web Browser), o servidor web, o banco de dados e o sistema de supervisão. O programa de simulação e o programa de supervisão são executados em "ambientes" diferentes (JBUILDER e WebServer). Isso possibilita que esses dois sistemas possam estar em locais diferentes, conferindo maior versatilidade no acompanhamento da supervisão predial.

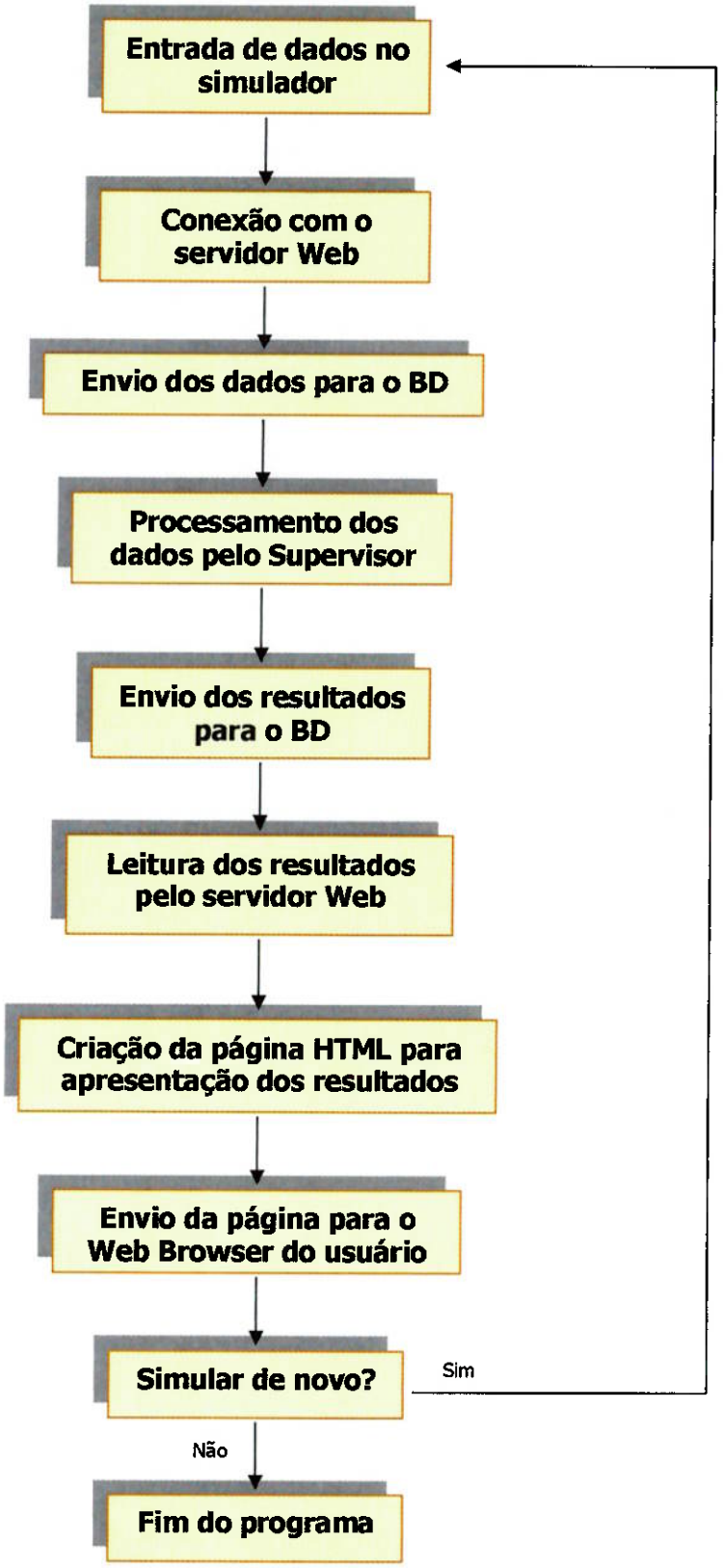


Figura 17 – Fluxograma do programa de simulação.

8.1 Interface de Entrada de Dados

Através de um Web Browser, inserem-se os dados para se simular uma supervisão de um edifício. Para tanto, os dados necessários para se realizar essa tarefa são:

- Para configuração dos ambientes:
  - *Temperatura desejada*: insere-se aqui a temperatura que se deseja para um determinado ambiente;
  - *Horário Permitido de Entrada*: é o horário mínimo permitido para a presença em um determinado ambiente;
  - *Horário Permitido de Saída*: é o horário máximo permitido para a presença em um determinado ambiente
- Para Simulação da Supervisão:
  - *Temperatura atual*: insere-se aqui a temperatura atual do ambiente, simulando-se assim a leitura que um sensor de temperatura estaria fazendo no mesmo;
  - *Presença*: insere-se aqui a informação de que se há ou não pessoas em um ambiente, simulando-se assim a leitura que um sensor de presença estaria fazendo no mesmo. Além disso, escolhe-se o horário em que foi feita a leitura do sensor de presença: a mesma no momento da simulação ou escolhe-se uma outra;
  - *Fumaça*: insere-se aqui a informação de que se há ou não fumaça em um ambiente, simulando-se assim a leitura que um sensor de fumaça estaria fazendo no mesmo. Além disso, escolhe-se o horário em que foi feita a leitura do sensor de fumaça: a mesma no momento da simulação ou escolhe-se uma outra.

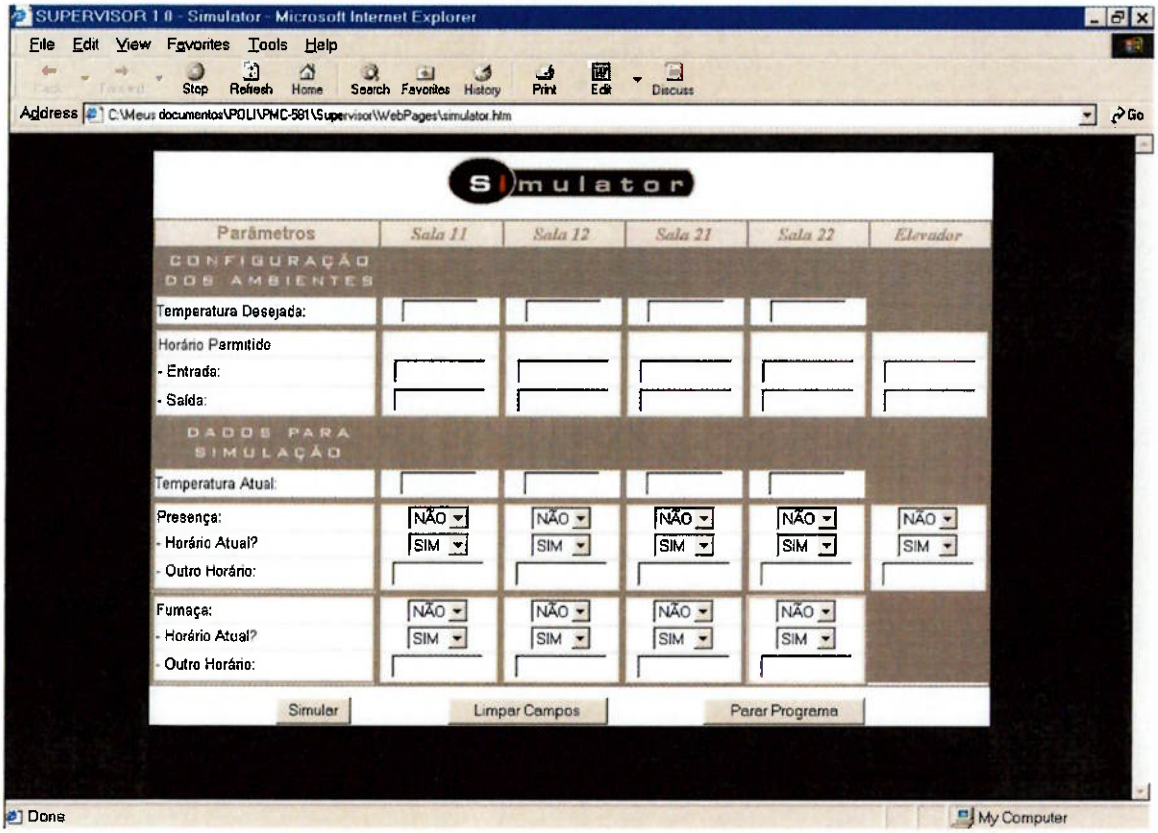


Figura 18 - Tela de entrada de dados do Simulator.

8.2 Apresentação dos Resultados

Uma vez que os dados para a simulação foram inseridos na interface do *Simulator*, o *Servlet em Java* se encarrega de transferi-los para o banco de dados. Com isso, o sistema de supervisão pode recuperá-los para analisar e processar os resultados da simulação, correspondendo para cada status de um ambiente uma ação de controle, além de apresentar o estado do mesmo.

Os resultados possíveis na simulação do sistema de supervisão são:

- *Incêndio*: apresentação de status de alarme de incêndio, de normalidade (ok) ou de erro no sistema, causado pelo mau funcionamento dos sensores de fumaça e/ou temperatura;
- *Segurança*: apresentação de status de alarme de invasão (presença não permtida), de normalidade (ok) ou de erro no sistema, causado pelo mau funcionamento do sensor de presença;
- *Conforto Térmico*: apresentação de status da temperatura no ambiente, se está abaixo do ideal, se está na ideal (ok), se está acima do esperado ou erro no sistema, causado pelo mau funcionamento do sensor de temperatura;
- *Sensor de Fumaça*: apresenta a leitura do sensor em um dado momento, indicando se há ou não fumaça no ambiente ou se o sensor não está funcionando (defeito);
- *Sensor de Temperatura*: apresenta o valor da temperatura atual no ambiente;
- *Sensor de Presença*: mostra o status de um ambiente - se está ocupada ou vazia. Apresenta uma mensagem de erro caso for detectado que o sensor de presença não está funcionando;
- *Status dos sensores*: apresenta se os sensores estão ou não funcionando (ok ou com defeito).

Para exemplificar o funcionamento do programa para simulação, seguem-se algumas figuras com os dados utilizados e seus conseqüentes resultados.

- Exemplo 1:

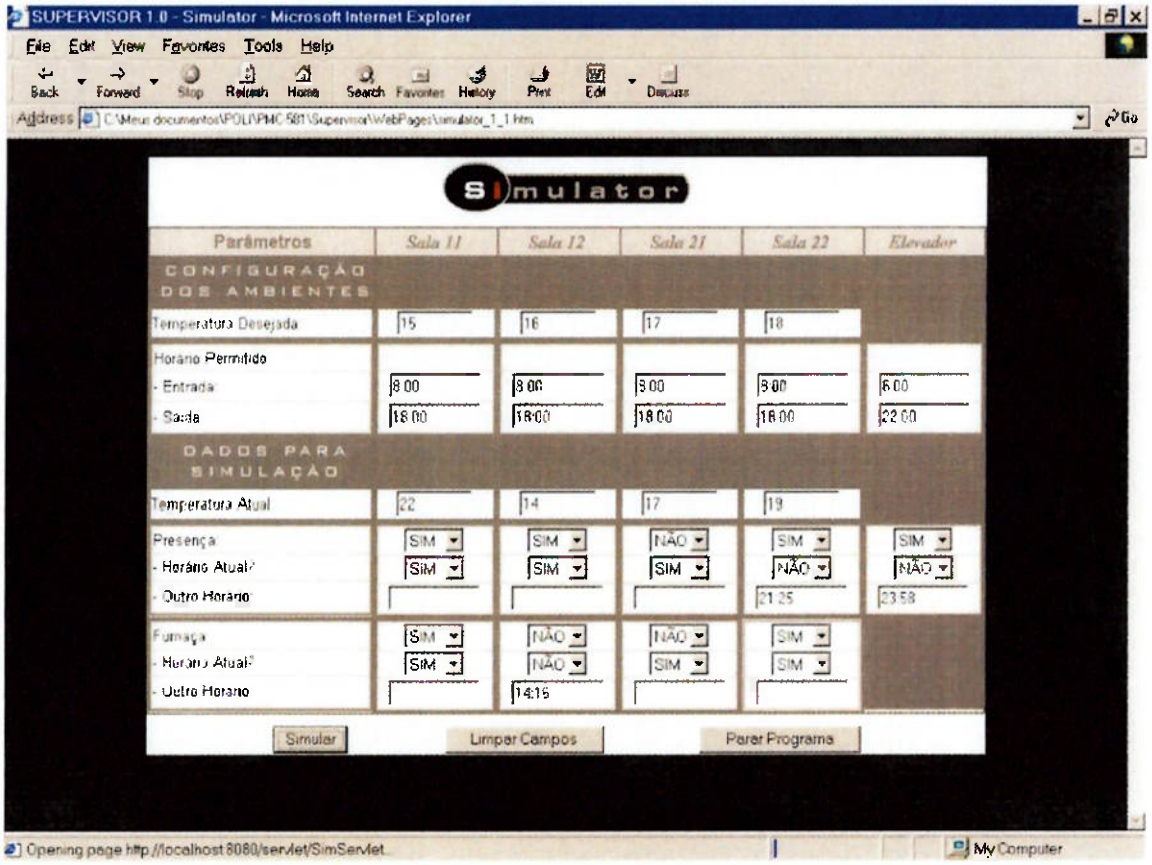


Figura 19 - Entrada de dados.

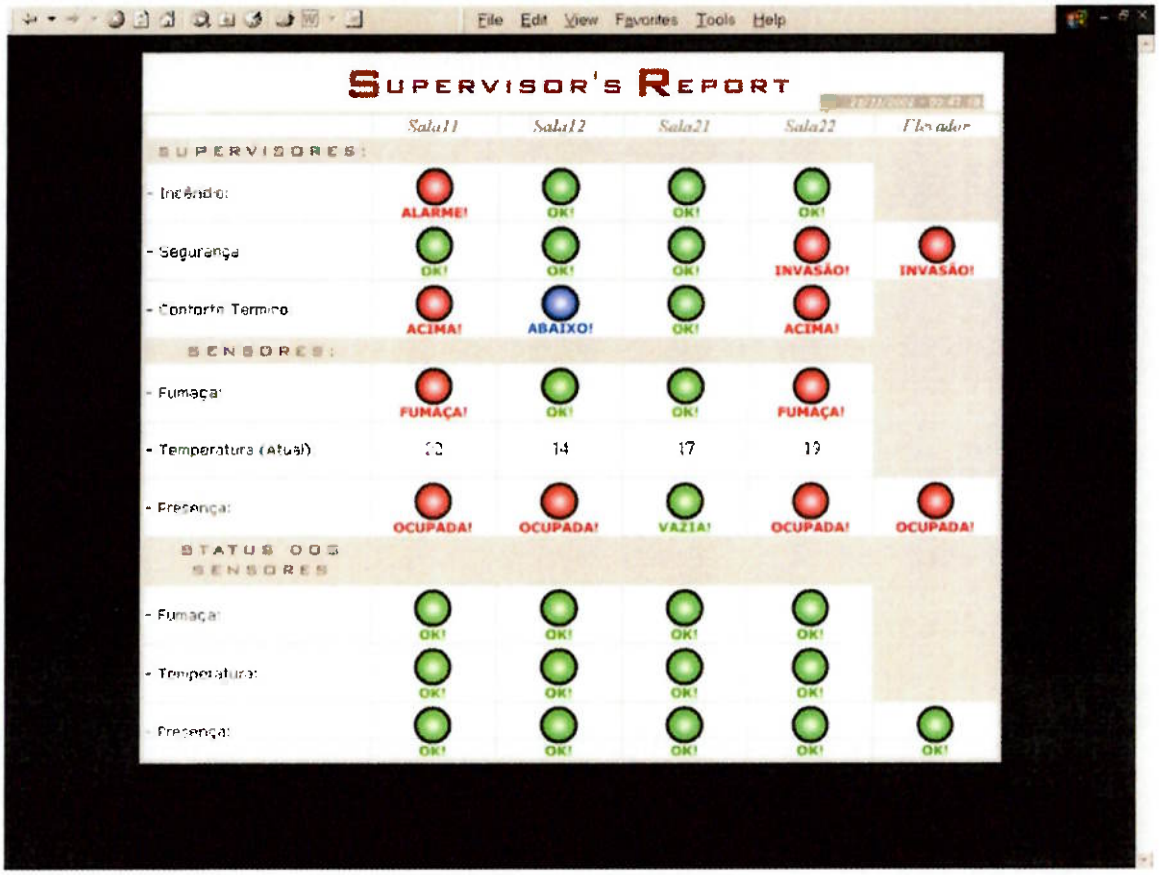


Figura 20 - Resultados.

- Exemplo 2:

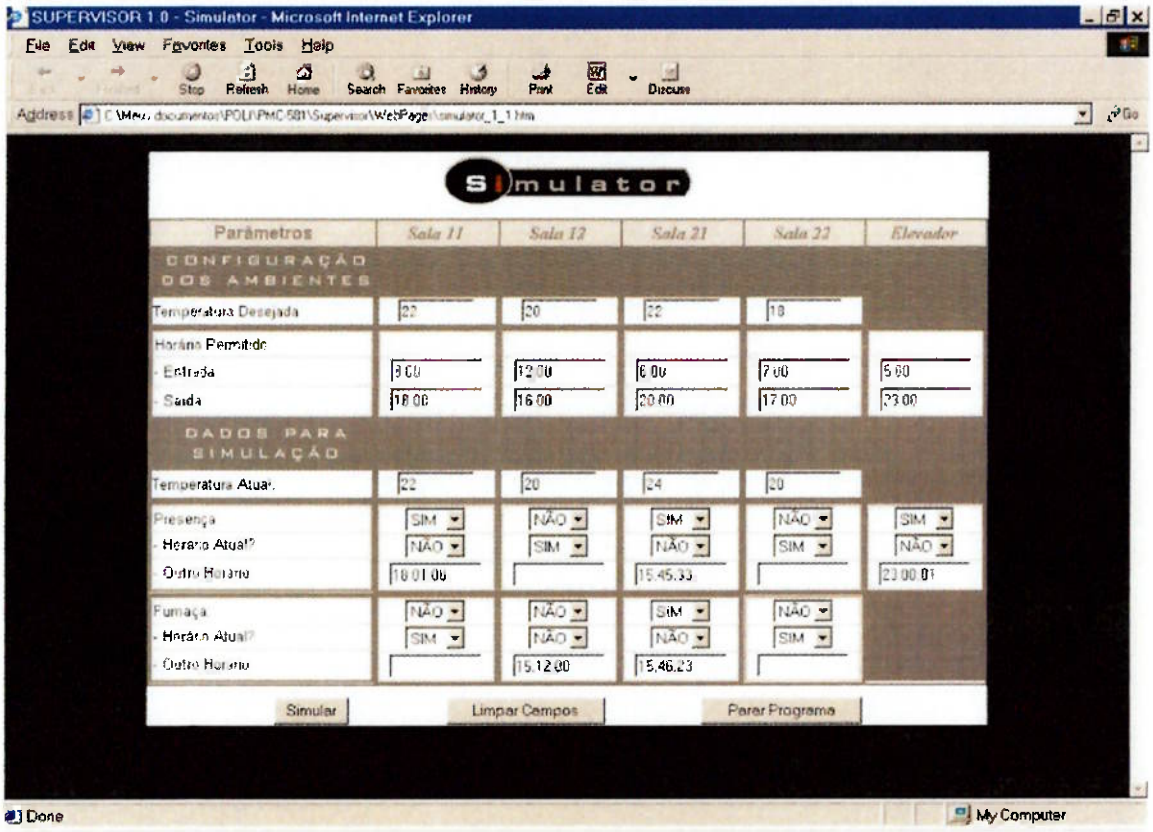


Figura 21 - Entrada de dados.

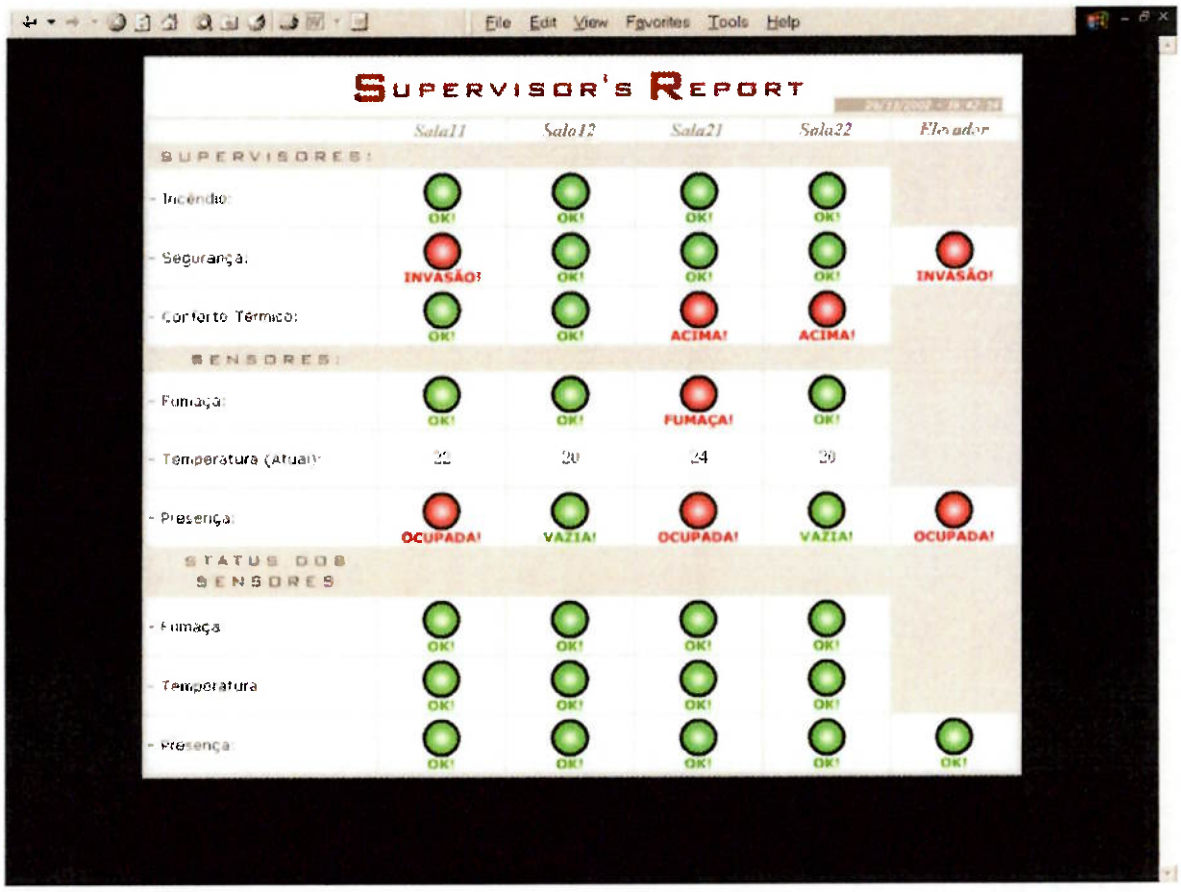


Figura 22 - Resultados.

- Exemplo 3:

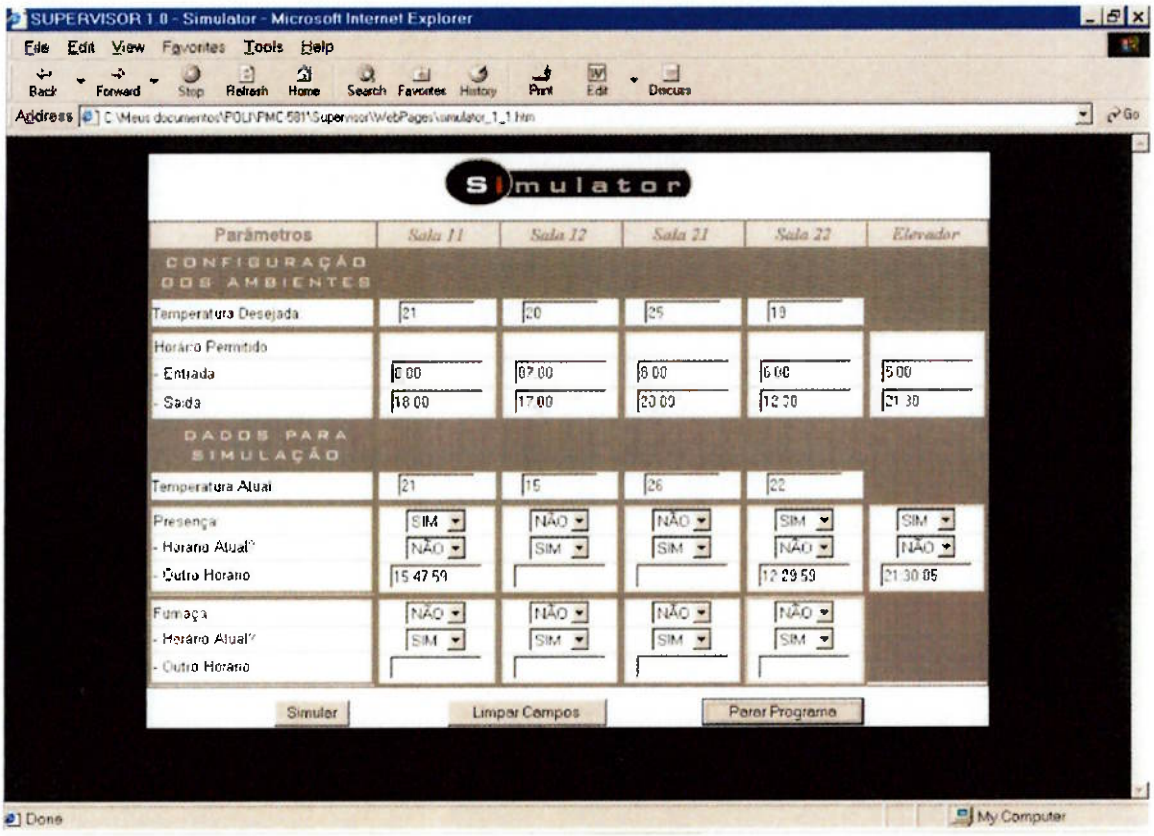


Figura 23 - Entrada de dados.

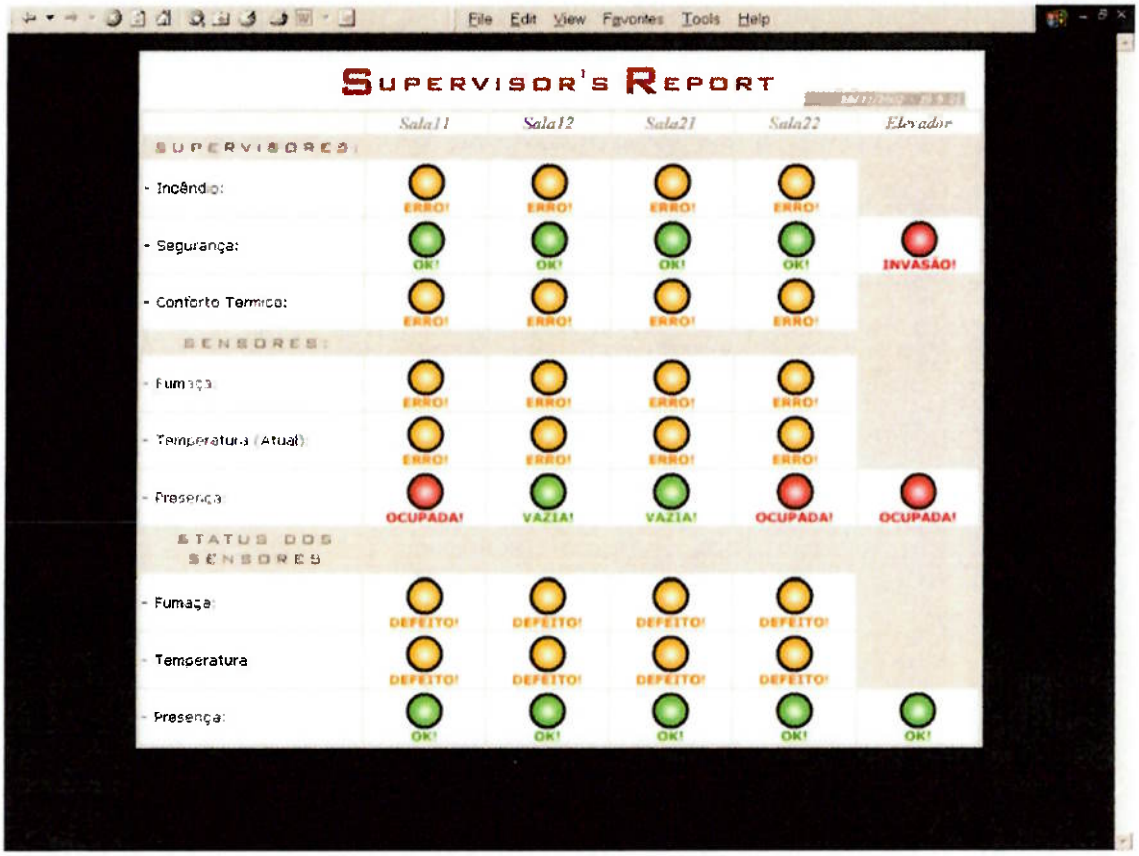
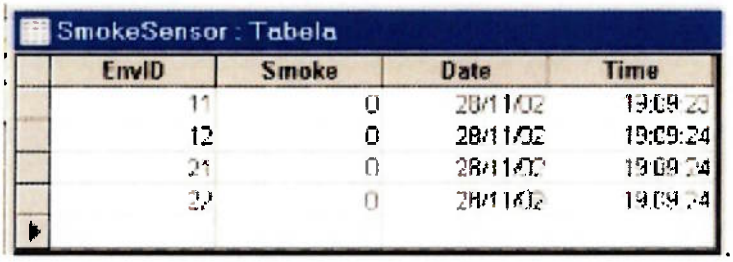


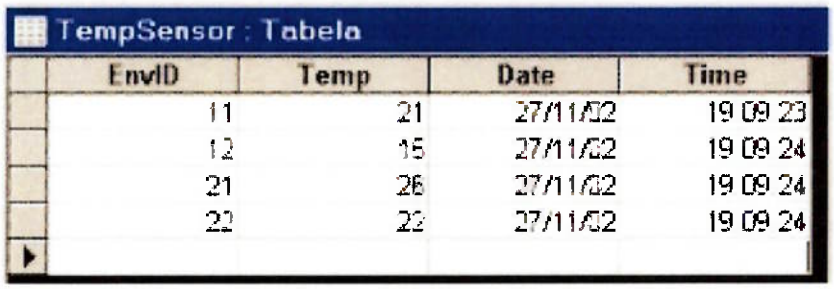
Figura 24 - Resultados



SmokeSensor : Tabela

EnvID	Smoke	Date	Time
11	0	28/11/02	19:09:23
12	0	28/11/02	19:09:24
21	0	28/11/02	19:09:24
22	0	28/11/02	19:09:24

Figura 25 - Status do sensor de fumaça nesse exemplo.



TempSensor : Tabela

EnvID	Temp	Date	Time
11	21	27/11/02	19 09 23
12	15	27/11/02	19 09 24
21	26	27/11/02	19 09 24
22	22	27/11/02	19 09 24

Figura 26 - Status do sensor de temperatura nesse exemplo.

- *Exemplo 4:*

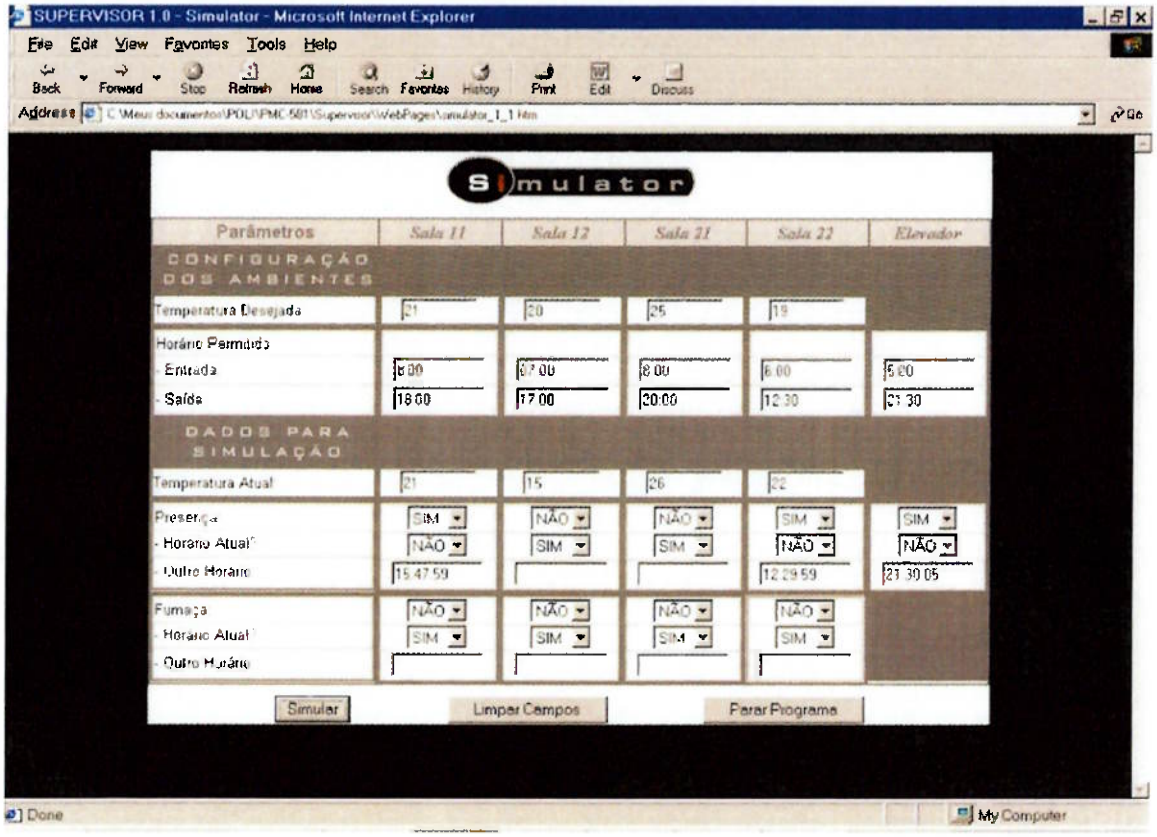


Figura 27 - Entrada de dados.

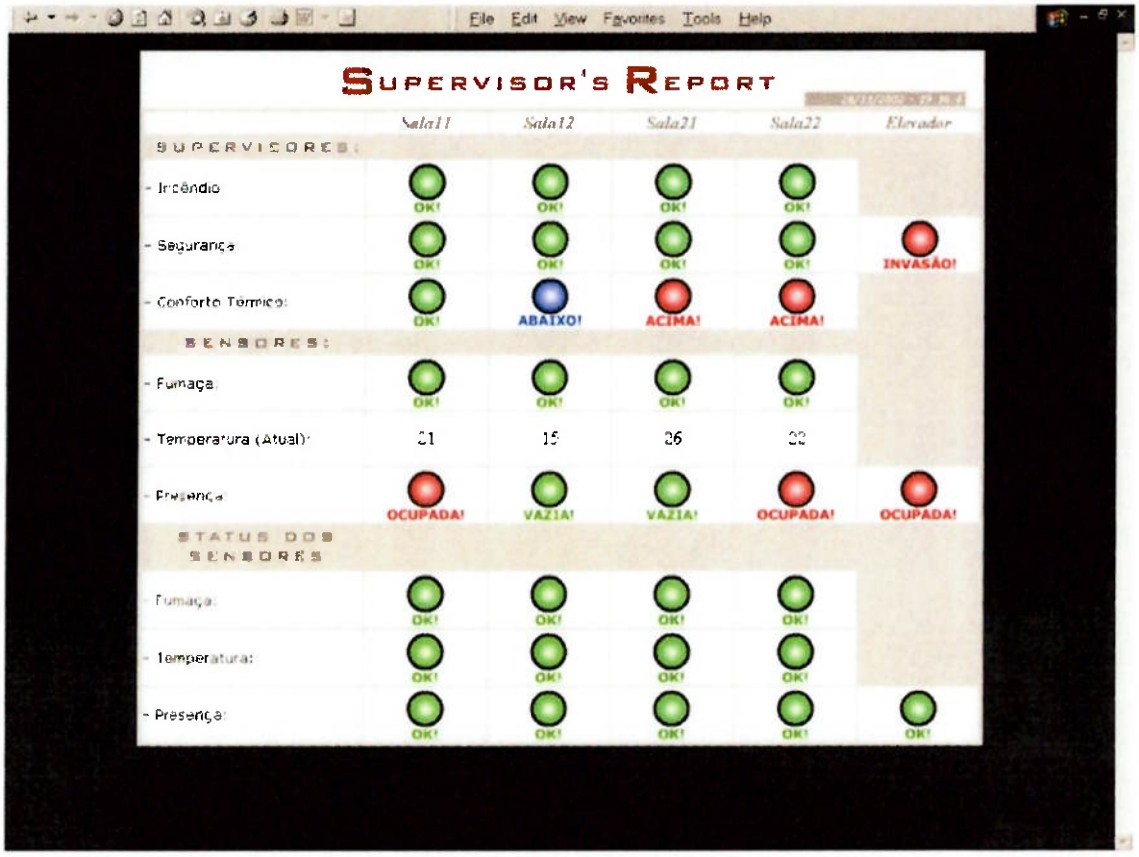


Figura 28 - Resultados.

9 Considerações Finais

O trabalho aqui desenvolvido é apenas o começo de um interessante estudo que abrange as mais diversas áreas da engenharia, tais como automação, termodinâmica, eletrônica, sistemas distribuídos, entre outros. Portanto, tem-se que o tema reserva inúmeros caminhos para serem trilhados com o intuito de cada vez mais desenvolver e ampliar a esfera de aplicações e funcionalidades da automação predial.

Tendo isso em mente, o trabalho será continuado tendo como alguns objetivos já estabelecidos, os seguintes itens:

- Desenvolvimento da interface gráfica com o usuário via *"browser"*, utilizando a linguagem JAVA (*"servelets"* ou JSP);
- Integração com o sistema de informação desenvolvido pelos alunos Marco Dyodi Takahashi e Patrick M. von Schaaffhausen, como trabalho de formatura;
- Aprimorar o critério de análise de incêndio, através da verificação do histórico da temperatura no ambiente, observando sua tendência de crescimento, por exemplo;
- Migrar a base de dados existente em Access para Oracle 9i, resultando também numa nova modelagem de acordo com as novas funcionalidades que a nova base de dados possibilita (orientação a objetos);
- Implementar a checagem dos sensores utilizando-se também a hora como critério verificador;

- Analisar a influência do tempo de ciclo do programa e do tempo de leitura e atualização dos dados dos sensores, para que não se tenha o problema de se tirar conclusões errôneas sobre os eventos que ocorrem na maquete.

Cabe ainda ressaltar que certos fatores podem influenciar na modelagem do comportamento dos eventos que podem ocorrer na maquete. Portanto, deve-se ainda observar e analisar a influência de fatores como:

- A dinâmica envolvida nos fenômenos de refrigeração de ambientes;
- Estudar a velocidade com que o calor se propaga em caso de haver focos de incêndio em um recinto e qual a influência disso nos sistemas de refrigeração e de sensoreamento;
- Analisar o tempo de resposta e a freqüência com que os sensores executam suas tarefas, para verificar se estes são adequados à analise dos fenômenos dos quais a maquete está passível;
- Estudar a vantagem do desenvolvimento de sistema que utilizam processamentos paralelos, de modo a tornar mais rápido o processamento, além de mais seguro.

10 Conclusão

A utilização de um sistema de informação com a finalidade de auxiliar a tarefa de controle e também fornecer novas informações para fomentar ações de controle mais eficientes resulta em um considerável ganho para a automação predial. Isso se deve ao fato de se utilizar informações que estão larga e facilmente acessíveis, com o intuito de se analisar e prever comportamentos de forma automática e com grande velocidade.

Além disso, como este sistema estaria atuando em uma camada acima em relação ao sistema de controle, tem-se então uma grande flexibilidade em se modificar ou até criar novas políticas de controle. Com isso, consegue-se um sistema que tem a possibilidade de se adequar a novas situações, muitas vezes não previstas na fase de projeto, além de abrir a possibilidade de se aplicar sistemas inteligentes, que possam responder de forma rápida e precisa às necessidades da planta predial.

A automação predial é um tema que seduz e incita nossa criatividade, sendo também fonte de inspiração para sistemas ligados à automação residencial. Esse tema já é realidade e pode ser encontrado em diversos lugares do mundo, com os mais diversos graus de sofisticação. Mas sempre haverá lugar para novas idéias e novos projetos, pois nem todos ainda podem usufruir os benefícios que a tecnologia pode trazer. E a tecnologia, inerentemente, deve ser feita para servir a humanidade como um todo, não devendo ser um fator desagregador.

ANEXO A

Segue o código fonte em JAVA do programa para supervisão da maquete. Foram utilizados *"Java Development Kit 1.3"*, *"Java Servlet Development Kit 2.1"* e o ambiente de desenvolvimento *Jbuilder 5*, da Borland.

- SUPERVISOR:

Super Classe "Ambient"

```
/**
 * Title:        Sistema supervisório
 * Description:  CAmbient Class
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */

package supervisor;

/* Super Classe Ambient: implementa os ambientes da maquete */
public class CAmbient {
```

```
  int EnvID;       // número que identifica univocamente o ambiete
  String Location; // descrição da localização do ambiente


  /* Retorna o número de identificação do ambiente */
  public int GetID(){
   return EnvID;
  } //GetID

  /* Retorna a descrição da localização do ambiente */
  public String GetLocation(){
   return Location;
  } //GetLocation

  /* Construtor da classe Ambient */
  public CAmbient(String LocationName, int RoomID){
   Location = LocationName;
   EnvID = RoomID;
  } //CAmbient

} // CAmbient
```

Classe "Room"

```
/**
 * Title:        Sistema supervisório
 * Description:  CRoom Class
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */

package supervisor;
import java.sql.*;

/* Classe Room: implementa as salas da maquete */
public class CRoom extends CAmbient {

  float Temperature_Last;      // última leitura do sensor de temperatura
  float Temperature_Reference; // temperatura de referência
  float Smoke_Last;            // última leitura do sensor de fumaça
  float Presence_Last;         // última leitura do sensor de presença
  long InitialHour;            // horário inicial de permissão de presença
  long FinalHour;              // horário final de permissão de presença
```

```
// Retorna a última leitura do sensor de temperatura da sala
public float GetLastTemp(){
  return Temperature_Last;
} //GetLastTemp

// Retorna a temperatura de referência da sala
public float GetRefferenceTemp(){
  return Temperature_Reference;
} //GetRefferenceTemp

// Retorna a última leitura do sensor de fumaça da sala
public float GetLastSmoke(){
  return Smoke_Last;
} //GetLastSmoke

// Retorna a última leitura do sensor de presença da sala
public float GetLastPresence(){
  return Presence_Last;
} //GetLastPresence

// Retorna o horário inicial de permissão de presença da sala
public long GetInitialHour(){
  return InitialHour;
```

```
 } //GetInitialHour

 // Retorna o horário final de permissão de presença da sala
 public long GetFinalHour(){
  return FinalHour;
 } //GetFinalHour

 // Atualiza os dados relacionados com a sala
 public void RefreshRoomData(CDBInterface Interface){
  Temperature_Last = Interface.GetAmbientLastTemperature(this);
  Temperature_Reference = Interface.GetAmbientReferenceTemperature(this);
  Smoke_Last = Interface.GetAmbientLastSmoke(this);
  Presence_Last = Interface.GetAmbientLastPresence(this);
  InitialHour = Interface.GetAmbientInicialTimePermission(this);
  FinalHour = Interface.GetAmbientFinalTimePermission(this);
 } //RefreshRoomData

 // Construtor da sala
 public CRoom(String RoomName, int ID) {
  super(RoomName, ID);
 } //CRoom

} //CRoom
```

Classe "Elevator"

```
/**
 * Title:        Sistema supervisório
 * Description:  CElevator Class
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */


package supervisor;

/* Classe Elevator: implementa o elevador */
public class CElevator extends CAmbient {

  float Presence_Last;  // última leitura do sensor de presença
  long InitialHour;     // horário inicial de permissão de presença
  long FinalHour;       // horário final de permissão de presença

  // Retorna a última leitura do sensor de presença do elevador
  public float GetLastPresence(){
    return Presence_Last;
```

```
} //GetLastPresence


// Retorna o horário inicial de permissão de presença do elevador
public long GetInitialHour(){
  return InitialHour;
} //GetInitialHour


// Retorna o horário final de permissão de presença do elevador
public long GetFinalHour(){
  return FinalHour;
} //GetFinalHour


// Atualiza os dados relacionados com o elevador
public void RefreshElevatorData(CDBInterface Interface){
 Presence_Last = Interface.GetAmbientLastPresence(this);
 InitialHour = Interface.GetAmbientInicialTimePermission(this);
 FinalHour = Interface.GetAmbientFinalTimePermission(this);
} //RefreshElevatorData

// Construtor do elevador
```

```
  public CElevator(String LocationName, int ID) {
   super(LocationName, ID);
  } //CElevator

} //CElevator
```

Classe "Supervisor"

```
/**
 * Title:        Sistema supervisório
 * Description:  CSupervisor Class
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */


package supervisor;
import java.util.*;

// Classe Supevisor: implementa o sistema de supervisão
public class CSupervisor {
```

```
LinkedList refAmbientList;  // Lista com as referências dos ambientes
CDBInterface refInterface;

/* Executa o sistema de supervisão para cada ambiente da maquete */
public void run(){

 Iterator i;
 i = refAmbientList.iterator();
 while(i.hasNext()){
  CAmbient Ambient = (CAmbient) i.next();
  run(Ambient);
 } //while

} // run


/* Supervisão de Segurança: retorna 1 caso haja presença não permitida e 0
  caso contrário */
public void run_security(CAmbient Ambient){
 int PresenceSensorOk; // Variável que guarda o status do sensor

 // Verifica a segurança nas salas
```

```
if(Ambient instanceof CRoom){
 PresenceSensorOk = refInterface.PresenceSensor(Ambient);

 if(PresenceSensorOk == 1){
  CRoom Room = (CRoom)Ambient;
  // Guarda a hora em que foi feita a leitura do sensor
  long SensorTime = refInterface.TimePresenceSensor(Ambient);
  // Guarda a hora mínima de presença permitida
  long InTime = Room.GetInitialHour();
  // Guarda a hora máxima de presença permitida
  long OutTime = Room.GetFinalHour();
  // Guarda a última leitura do sensor
  float Sensor = refInterface.GetAmbientLastPresence(Ambient);

  if(Sensor == 1){
    // Verificação se a presença está dentro do horário permitida
    if(SensorTime >= InTime && SensorTime <= OutTime)
     refInterface.SetAmbientSecurityResult(Ambient, 0);
    else
     refInterface.SetAmbientSecurityResult(Ambient, 1);
  } //if

  else
```

```
        refInterface.SetAmbientSecurityResult(Ambient, 0);
    } //if

    else{
     System.out.println("WARNING:Presence Sensor of "+
     Ambient.GetLocation() +" is NOT working");
     refInterface.SetAmbientSecurityResult(Ambient, 33);
    } //else

} //if

// Verifica a segurança no Elevador
else if(Ambient instanceof CElevator){
  PresenceSensorOk = refInterface.PresenceSensor(Ambient);

  if(PresenceSensorOk == 1){
    CElevator Elevator = (CElevator)Ambient;
    // Guarda a hora em que foi feita a leitura do sensor
    long SensorTime = refInterface.TimePresenceSensor(Ambient);
    // Guarda a hora mínima de presença permitida
    long InTime = Elevator.GetInitialHour();
    // Guarda a hora máxima de presença permitida
    long OutTime = Elevator.GetFinalHour();
```

```
      // Guarda a última leitura do sensor
      float Sensor = refInterface.GetAmbientLastPresence(Ambient);

      if(Sensor == 1){
         // Verificação se a presença está dentro do horário permitida
         if(SensorTime >= InTime && SensorTime <= OutTime)
          refInterface.SetAmbientSecurityResult(Ambient, 0);
         else
          refInterface.SetAmbientSecurityResult(Ambient, 1);
      } //if
      else
        refInterface.SetAmbientSecurityResult(Ambient, 0);
    } //if

    else{
     System.out.println("WARNING:Presence Sensor of "+
     Ambient.GetLocation() +" is NOT working!");
     refInterface.SetAmbientSecurityResult(Ambient, 33);
    } //else
  } //else if

} //run_security
```

```
/* Supervisão de Conforto térmico: retorna -1 caso a temperatura atual esteja
   abaixo da deseja, 0 caso a temperatura atual esteja de acordo ou 1 caso a
   temperatura atual esteja acima da desejada */
public void run_comfort(CAmbient Ambient){
 int TempSensorOk; // Variável que guarda o status do sensor

 if(Ambient instanceof CRoom){

  TempSensorOk = refInterface.TempSensor(Ambient);

  if(TempSensorOk == 1){
   CRoom room = (CRoom)Ambient;

   // Guarda a diferença entre a temperatura atual e a desejada para a sala
   float Diff = room.GetLastTemp() - room.GetRefferenceTemp();

   if(Diff < 0)
    refInterface.SetAmbientComfortResult(Ambient, -1);

   else if(Diff == 0)
    refInterface.SetAmbientComfortResult(Ambient, 0);
```

```
      else
        refInterface.SetAmbientComfortResult(Ambient, 1);
     } //if

     else{
      System.out.println("WARNING:Temperature Sensor of "+
      Ambient.GetLocation() +" is NOT working!");
      // Armazena no BD o valor 33 caso o sensor não esteja funcionando
      refInterface.SetAmbientComfortResult(Ambient, 33);
     } //else
   } //if

} //run_comfort


/* Supervisão de Incêndio: retorna 1 caso haja incêndio e 0 caso contrário */
public void run_fire(CAmbient Ambient){
  int SmokeSensorOk, // Variável que guarda o status do sensor
      TempSensorOk; // Variável que guarda o status do sensor

  if(Ambient instanceof CRoom){

    SmokeSensorOk = refInterface.SmokeSensor(Ambient);
```

```
TempSensorOk = refInterface.TempSensor(Ambient);

if(SmokeSensorOk == 1 && TempSensorOk == 1){
 CRoom room = (CRoom)Ambient;
 float AmbTemp = room.GetLastTemp();
 float RefTemp = room.GetRefferenceTemp();

 // Se houver fumaça e a temperatura atual estiver 5 graus acima da desejada
 // o alarme de incêndio é acionado.
 if(room.GetLastSmoke() == 1 && (AmbTemp - RefTemp > 5))
  refInterface.SetAmbientFireResult(Ambient, 1);

 else
  refInterface.SetAmbientFireResult(Ambient, 0);
} //if

else{
 System.out.println("WARNING:Smoke and/or Temperature Sensors of "+
 Ambient.GetLocation() +" are NOT working!");
 refInterface.SetAmbientFireResult(Ambient, 33);
} //else
} //if
```

```
 } //run_fire


 /* Executa a supervisão de incêndio, segurança e conforto para um determinado
    ambiente */
 public void run(CAmbient Ambient){
  if(Ambient instanceof CRoom){
     run_fire(Ambient);
     run_security(Ambient);
     run_comfort(Ambient);
  } //if

  else if(Ambient instanceof CElevator){
     run_security(Ambient);
  } //else if

} //run


// Construtor do Supervisor
public CSupervisor(LinkedList refList, CDBInterface refInterf) {
 refAmbientList = refList;
 refInterface = refInterf;
```

```
  } //CSupervisor

} //CSupervisor
```

Classe "DBInterface"

```
/**
 * Title:        Sistema supervisório
 * Description:  CDBInterface Class
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */

package supervisor;

import java.sql.*;
import java.util.Date;
import java.lang.*;
import java.lang.Object;



/* Classe DBInterface: interface com o BD */
```

```
public class CDBInterface {

 Connection connection;  // variável relacionada com a conexão com o BD

 /* Realiza a conexão com o BD via ODBC */
 public CDBInterface() {
  String url = "jdbc:odbc:supervisao";
  String username = "Administrador";
  String password = "1234";
  try {
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   connection = DriverManager.getConnection(url, username, password);
  }
  catch(ClassNotFoundException cnfex ) {
   System.err.println("Supervisor: Fail to load JDBC");
   cnfex.printStackTrace();
   System.exit(1);
  }
  catch( SQLException sqlex) {
   System.err.println("Supervisor: unable to connect");
   sqlex.printStackTrace();
  }
}
```

```
 /* Finaliza a conexão com o BD */
 protected void finalize(){
  try {
    connection.close();
  }
  catch ( SQLException sqlex ){
    System.err.println("Supervisor: unable to disconnect");
    sqlex.printStackTrace();
  }
}

/* Executa a query para consulta ao BD */
private float Consulting (String query, String Column){
    float Result = 0;
    Statement statement;
    ResultSet resultset;
    try {
      statement = connection.createStatement();
      resultset = statement.executeQuery(query);

      if(resultset.next()){
        Result = resultset.getFloat(Column);
```

```
    }
    else {
      System.out.println("Supervisor: there is no related value!");
      System.exit(1);
    }
   }
   catch (SQLException sqlex){
    sqlex.printStackTrace();
   }
 return Result;
 }

/* Retorna o horário em que a leitura da presença foi feita no ambiente */
public long TimePresenceSensor(CAmbient Ambient){
 long STime = 0;
 if(Ambient instanceof CRoom || Ambient instanceof CElevator){
  Integer ID = new Integer(Ambient.GetID());

  String query = "SELECT TOP 1 Time FROM (SELECT Presence, Time FROM "+
          "(SELECT Presence, Date, Time FROM PresenceSensor WHERE "+
          "EnvID =" + ID.toString()+ ") WHERE Date = (SELECT TOP 1 "+
          "Date FROM (SELECT Presence, Date, Time FROM PresenceSensor"+
          " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC)) "+
```

```
            "ORDER BY Time Desc);";

    Statement statement;
    ResultSet resultset;
    Time SensorTime = new Time(0);

    try {
     statement = connection.createStatement();
     resultset = statement.executeQuery(query);

     if(resultset.next()){
      long curr = System.currentTimeMillis();
      Date timeref = new Date(curr);
      Integer year = new Integer(timeref.getYear() + 1900);
      Integer month = new Integer(timeref.getMonth() + 1);
      Integer day = new Integer(timeref.getDate());
      SensorTime = resultset.getTime("Time");

      String s = year.toString()+ "/"+ month.toString()+ "/"+
            day.toString()+ ","+ SensorTime.toString();

      Date time = new Date(Time.parse(s));
```

```
        STime = time.getTime();
      }
      else {
        System.out.println("Supervisor: there is no related value!");
        System.exit(1);
      }
    }
    catch (SQLException sqlex){
     sqlex.printStackTrace();
    }
  }
 return STime;
}

/* Retorna 1 se o sensor de fumaça estiver funcionando e 0 caso contrário */
public int SmokeSensor(CAmbient Ambient){
 int Function = 0;

 if(Ambient instanceof CRoom){
   Integer ID = new Integer(Ambient.GetID());

   String query = "SELECT TOP 1 Date FROM (SELECT Date FROM SmokeSensor"+
                  " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC);";
```

```
Statement statement;
ResultSet resultset;
Date Sensor = new Date();
String Data;

try {
   statement = connection.createStatement();
   resultset = statement.executeQuery(query);

   if(resultset.next()){
    Sensor = resultset.getDate("Date");

    Integer Syear = new Integer(Sensor.getYear());
    Integer Smonth = new Integer(Sensor.getMonth());
    Integer Sday = new Integer(Sensor.getDate());

    String SensorDate = Syear.toString() +"-"+ Smonth.toString() +"-"+
            Sday.toString();
    String SDate = new String(SensorDate);

    long curr = System.currentTimeMillis();
    Date timeref = new Date(curr);
```

```
        Integer year = new Integer(timeref.getYear());
        Integer month = new Integer(timeref.getMonth());
        Integer day = new Integer(timeref.getDate());

        String today = year.toString() +"-"+ month.toString() +"-"+
                day.toString();

        if(SDate.equalsIgnoreCase(today))
         Function = 1;
        else
         Function = 0;
      }
    }
    catch (SQLException sqlex){
     sqlex.printStackTrace();
    }
  }
  return Function;
}

/* Retorna 1 se o sensor de presença estiver funcionando e 0 caso contrário */
public int PresenceSensor(CAmbient Ambient){
 int Function = 0;
```

```
if(Ambient instanceof CRoom || Ambient instanceof CElevator){
 Integer ID = new Integer(Ambient.GetID());

 String query = "SELECT TOP 1 Date FROM (SELECT Date FROM PresenceSensor"+
        " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC);";

 Statement statement;
 ResultSet resultset;
 Date Sensor = new Date();
 String Data;

 try {
   statement = connection.createStatement();
   resultset = statement.executeQuery(query);

   if(resultset.next()){
     Sensor = resultset.getDate("Date");

     Integer Syear = new Integer(Sensor.getYear());
     Integer Smonth = new Integer(Sensor.getMonth());
     Integer Sday = new Integer(Sensor.getDate());
```

```
        String SensorDate = Syear.toString() +"-"+ Smonth.toString() +"-"+
                    Sday.toString();
        String SDate = new String(SensorDate);

        long curr = System.currentTimeMillis();
        Date timeref = new Date(curr);
        Integer year = new Integer(timeref.getYear());
        Integer month = new Integer(timeref.getMonth());
        Integer day = new Integer(timeref.getDate());

        String today = year.toString() +"-"+ month.toString() +"-"+
                    day.toString();

        if(SDate.equalsIgnoreCase(today))
          Function = 1;
        else
          Function = 0;
      }
    }
    catch (SQLException sqlex){
     sqlex.printStackTrace();
    }
  }
```

```
  return Function;
 }

 /* Retorna 1 se o sensor de temperatura estiver funcionando e 0 caso contrário */
 public int TempSensor(CAmbient Ambient){
  int Function = 0;

  if(Ambient instanceof CRoom){
   Integer ID = new Integer(Ambient.GetID());

   String query = "SELECT TOP 1 Date FROM (SELECT Date FROM TempSensor"+
            " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC);";

   Statement statement;
   ResultSet resultset;
   Date Sensor = new Date();
   String Data;

   try {
     statement = connection.createStatement();
     resultset = statement.executeQuery(query);

     if(resultset.next()){
```

```
Sensor = resultset.getDate("Date");

Integer Syear = new Integer(Sensor.getYear());
Integer Smonth = new Integer(Sensor.getMonth());
Integer Sday = new Integer(Sensor.getDate());

String SensorDate = Syear.toString() +"-"+ Smonth.toString() +"-"+
        Sday.toString();
String SDate = new String(SensorDate);

long curr = System.currentTimeMillis();
Date timeref = new Date(curr);
Integer year = new Integer(timeref.getYear());
Integer month = new Integer(timeref.getMonth());
Integer day = new Integer(timeref.getDate());

String today = year.toString() +"-"+ month.toString() +"-"+
        day.toString();

if(SDate.equalsIgnoreCase(today))
 Function = 1;
else
 Function = 0;
```

```
      }
    }
    catch (SQLException sqlex){
     sqlex.printStackTrace();
    }
  }
  return Function;
}

/* Retorna a última leitura do sensor de temperatura do ambiente */
public float GetAmbientLastTemperature(CAmbient Ambient){
  float LastTemperature = 0;
  Integer ID = new Integer(Ambient.GetID());

  String query = "SELECT TOP 1 Temp FROM (SELECT Temp, Time FROM "+
            "(SELECT Temp, Date, Time FROM TempSensor WHERE "+
            "EnvID =" + ID.toString()+ ") WHERE Date = (SELECT TOP 1 "+
            "Date FROM (SELECT Temp, Date, Time FROM TempSensor"+
            " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC)) "+
            "ORDER BY Time Desc);";

 if(Ambient instanceof CRoom)
   LastTemperature = Consulting(query, "Temp");
```

```
  return LastTemperature;
 }

 /* Retorna a última leitura do sensor de presença do ambiente */
 public float GetAmbientLastPresence(CAmbient Ambient){
  float Presence = 0;
  Integer ID = new Integer(Ambient.GetID());

  String query = "SELECT TOP 1 Presence FROM (SELECT Presence, Time FROM "+
                 "(SELECT Presence, Date, Time FROM PresenceSensor WHERE "+
                 "EnvID =" + ID.toString()+ ") WHERE Date = (SELECT TOP 1 "+
                 "Date FROM (SELECT Presence, Date, Time FROM PresenceSensor"+
                 " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC)) "+
                 "ORDER BY Time Desc);";

  if(Ambient instanceof CRoom || Ambient instanceof CElevator)
    Presence = Consulting(query, "Presence");

  return Presence;
 }

 /* Retorna a última leitura do sensor de fumaça do ambiente */
```

```
public float GetAmbientLastSmoke(CAmbient Ambient){
  float Smoke = 0;
  Integer ID = new Integer(Ambient.GetID());

  String query = "SELECT TOP 1 Smoke FROM (SELECT Smoke, Time FROM "+
          "(SELECT Smoke, Date, Time FROM SmokeSensor WHERE "+
          "EnvID =" + ID.toString()+ ") WHERE Date = (SELECT TOP 1 "+
          "Date FROM (SELECT Smoke, Date, Time FROM SmokeSensor"+
          " WHERE EnvID = " + ID.toString()+ " ORDER BY Date DESC)) "+
          "ORDER BY Time Desc);";

  if(Ambient instanceof CRoom)
    Smoke = Consulting(query, "Smoke");

  return Smoke;
}

/* Retorna a temperatura de referênicia ou desejada para o ambiente */
public float GetAmbientReferenceTemperature(CAmbient Ambient){
  float RefTemperature = 0;
  Integer ID = new Integer(Ambient.GetID());
```

```
  String query = "SELECT TOP 1 RefTemp FROM (SELECT RefTemp, TTime FROM
"+
            "(SELECT RefTemp, DDate, TTime FROM UserInterface WHERE "+
            "RefTemp IS NOT NULL AND EnvID =" + ID.toString()+
            ")  WHERE DDate = (SELECT TOP 1 DDate FROM (SELECT RefTemp, "+
            "DDate, TTime FROM UserInterface WHERE RefTemp IS NOT NULL "+
            "AND EnvID = " + ID.toString()+ " ORDER BY DDate DESC)) "+
            "ORDER BY TTime Desc);";

  if(Ambient instanceof CRoom)
    RefTemperature = Consulting(query, "RefTemp");

  return RefTemperature;
 }

 /* Retorna o horário inicial de presença permitida para o ambiente */
 public long GetAmbientInicialTimePermission(CAmbient Ambient){
  Integer ID = new Integer(Ambient.GetID());
  long Start = 0;

  String query1 = "SELECT TOP 1 InicialPermTime FROM (SELECT
InicialPermTime"+
            ", TTime FROM(SELECT InicialPermTime, DDate, TTime FROM "+
```

```
            "UserInterface WHERE InicialPermTime IS NOT NULL AND "+
            "EnvID =" + ID.toString()+ ")  WHERE DDate = (SELECT TOP 1 "+
            "DDate FROM (SELECT InicialPermTime, DDate, TTime FROM "+
            "UserInterface WHERE InicialPermTime IS NOT NULL AND "+
            "EnvID = " + ID.toString()+ " ORDER BY DDate DESC)) "+
            "ORDER BY TTime Desc);";


if(Ambient instanceof CRoom || Ambient instanceof CElevator){
  Statement statement;
  ResultSet resultset1;
  Time InicialTime = new Time(0);

  try {
   statement = connection.createStatement();
   resultset1 = statement.executeQuery(query1);


   if(resultset1.next()){
    long curr = System.currentTimeMillis();
    Date timeref = new Date(curr);
    Integer year = new Integer(timeref.getYear() + 1900);
    Integer month = new Integer(timeref.getMonth() + 1);
```

```
        Integer day = new Integer(timeref.getDate());

        InicialTime = resultset1.getTime("InicialPermTime");
        String s1 = year.toString()+ "/"+ month.toString()+ "/"+
                day.toString()+ ","+ InicialTime.toString();

        Date time = new Date(Time.parse(s1));

        Start = time.getTime();
      }
      else {
        System.out.println("Supervisor: there is no related value!");
        System.exit(1);
      }
    }
    catch (SQLException sqlex){
      sqlex.printStackTrace();
    }
  }
  return Start;
}

/* Retorna o horário final de presença permitida para o ambiente */
```

```
public long GetAmbientFinalTimePermission(CAmbient Ambient){
  Integer ID = new Integer(Ambient.GetID());
  long End = 0;

  String query2 = "SELECT TOP 1 EndPermTime FROM (SELECT EndPermTime"+
                  ", TTime FROM(SELECT EndPermTime, DDate, TTime FROM "+
                  "UserInterface WHERE EndPermTime IS NOT NULL AND "+
                  "EnvID =" + ID.toString()+ ")  WHERE DDate = (SELECT TOP 1 "+
                  "DDate FROM (SELECT EndPermTime, DDate, TTime FROM "+
                  "UserInterface WHERE EndPermTime IS NOT NULL AND "+
                  "EnvID = " + ID.toString()+ " ORDER BY DDate DESC)) "+
                  "ORDER BY TTime Desc);";

  if(Ambient instanceof CRoom || Ambient instanceof CElevator){
    Statement statement;
    ResultSet resultset2;
    Time EndTime = new Time(0);

    try {
      statement = connection.createStatement();
      resultset2 = statement.executeQuery(query2);
      if(resultset2.next()){
        long curr = System.currentTimeMillis();
```

```
        Date timeref = new Date(curr);
        Integer year = new Integer(timeref.getYear() + 1900);
        Integer month = new Integer(timeref.getMonth() + 1);
        Integer day = new Integer(timeref.getDate());
        EndTime = resultset2.getTime("EndPermTime");

        String s2 = year.toString()+ "/"+ month.toString()+ "/"+
                day.toString()+ ","+ EndTime.toString();

        Date time = new Date(Time.parse(s2));

        End = time.getTime();
      }
      else {
        System.out.println("Supervisor: there is no related value!");
        System.exit(1);
      }
    }
    catch (SQLException sqlex){
      sqlex.printStackTrace();
    }
  }
  return End;
```

```
 }

 /* Executa a query para inserção no banco de dados */
 private void SendQuery(String query){
    try{
     Statement statement = connection.createStatement();
     int success = statement.executeUpdate(query);
     /*if (success == 1)
      System.out.println("Supervisor: Insertion successful");
     else
      System.out.println("Supervisor: Insertion not successful");*/
     statement.close();
    }
    catch (SQLException sqlex) {
     sqlex.printStackTrace();
     System.out.println("Supervisor: Insertion Failed");
    }
}

/* Insere no banco de dados o resultado da análise do sistema de supervisão
  de incêndio */
public void SetAmbientFireResult(CAmbient Ambient, int Result){
 if(Ambient instanceof CRoom){
```

```
    long curr = System.currentTimeMillis();
    Date timeref = new Date(curr);
    Integer year = new Integer(timeref.getYear() + 1900);
    Integer month = new Integer(timeref.getMonth() + 1);
    Integer day = new Integer(timeref.getDate());
    Integer hour = new Integer(timeref.getHours());
    Integer min = new Integer(timeref.getMinutes());
    Integer sec = new Integer(timeref.getSeconds());

    String today = day.toString()+ "/" + month.toString()+ "/" +
                year.toString();

    String time = hour.toString()+ ":" + min.toString()+ ":" +
                sec.toString();

    String query = "INSERT INTO Results (EnvID, FireAlarm, DDate, TTime)"+
                " VALUES(" +Ambient.GetID()+","+ Result+ ",'"+
                today +"','" + time +"');";
    SendQuery(query);
  }
}

/* Insere no banco de dados o resultado da análise do sistema de supervisão
```

```
   de segurança */
 public void SetAmbientSecurityResult(CAmbient Ambient, int Result){
  if(Ambient instanceof CRoom || Ambient instanceof CElevator){
   long curr = System.currentTimeMillis();
   Date timeref = new Date(curr);
   Integer year = new Integer(timeref.getYear() + 1900);
   Integer month = new Integer(timeref.getMonth() + 1);
   Integer day = new Integer(timeref.getDate());
   Integer hour = new Integer(timeref.getHours());
   Integer min = new Integer(timeref.getMinutes());
   Integer sec = new Integer(timeref.getSeconds());

   String today = day.toString()+ "/" + month.toString()+ "/" +
            year.toString();

   String time = hour.toString()+ ":" + min.toString()+ ":" +
           sec.toString();

   String query = "INSERT INTO Results (EnvID, IntrusionAlarm, DDate, "+
            "TTime) VALUES(" +Ambient.GetID()+","+ Result+ ",'"+
           today +"','" + time +"');";

  SendQuery(query);
```

```
    }
  }

  /* SIMULATOR: objeto usado para simular o sistema Supervisório enquanto não há
     a integração com a bancada de teste. Retorna 1 se for para simular, 0 para
     parar o programa e 33 caso não haja comando no momento da leitura do BD,
     significando que não se deve simular e nem parar o programa. */
  public int Simulator(){
    int Function = 0;

    /* Seleciona a data da última ordem de simular. */
    String query1 = "SELECT TOP 1 Date FROM (SELECT Date FROM "+
                    "Simulator ORDER BY Date DESC);";

    /* Seleciona a hora da última ordem de simular. */
    String query2 = "SELECT TOP 1 Time FROM (SELECT Time FROM Simulator"+
                    " WHERE Date = (SELECT TOP 1 Date FROM (SELECT Date FROM "+
                    "Simulator ORDER BY Date DESC))) ORDER BY Time  DESC;";

    /* Seleciona a última ordem de simular. */
    String query3 = "SELECT  Simulate  FROM (SELECT TOP 1 Time, Simulate
FROM"+
                    " (SELECT Time, Simulate FROM Simulator WHERE Date = "+
```

```
        "(SELECT TOP 1 Date FROM (SELECT Date FROM Simulator ORDER"+
        " BY Date DESC))) ORDER BY Time  DESC);";

Statement statement1;
Statement statement2;
Statement statement3;
ResultSet resultset1;
ResultSet resultset2;
ResultSet resultset3;
Date SimDate = new Date();
Time SimTime = new Time(0);
String Data;

try {
  statement1 = connection.createStatement();
  resultset1 = statement1.executeQuery(query1);

  if(resultset1.next()){
    /* Recebe a data da última ordem de simular */
    SimDate = resultset1.getDate("Date");

    /* Recupera o ano da data */
    Integer Syear = new Integer(SimDate.getYear());
```

```
/* Recupera o mês da data */
Integer Smonth = new Integer(SimDate.getMonth());
/* Recupera o dia da data */
Integer Sday = new Integer(SimDate.getDate());

/* Monta o String para comparação entre datas */
String SDate = Syear.toString() +"-"+ Smonth.toString() +"-"+
            Sday.toString();
//System.out.println("Data do Simulator:" + SDate);

/* Recebe a data e a hora atual em milissegundos */
long currday = System.currentTimeMillis();
/* Transforma a data em milissegundos em objeto DATE */
Date dayref = new Date(currday);
/* Recupera o ano da data atual */
Integer year = new Integer(dayref.getYear());
/* Recupera o mês da data atual */
Integer month = new Integer(dayref.getMonth());
/* Recupera o dia da data atual */
Integer day = new Integer(dayref.getDate());

/* Monta o String para comparação entre datas */
String todayd = year.toString() +"-"+ month.toString() +"-"+
```

```
            day.toString();
//System.out.println("Data de hoje:" + todayd);

/* Compara se a data da ordem é igual a atual, para validar a ordem*/
if(SDate.equalsIgnoreCase(todayd)){
  //System.out.println("Ok 1");
  try{
    statement2 = connection.createStatement();
    resultset2 = statement2.executeQuery(query2);

    if(resultset2.next()){
      /* Recebe a hora da última ordem de simular */
      SimTime = resultset2.getTime("Time");

      int Shour,
        Smin,
        Ssec;

      // Guarda a hora da ordem de simular
      Shour = SimTime.getHours();
      // Guarda os minutos da ordem de simular
      Smin = SimTime.getMinutes();
      // Guarda os segundos da ordem de simular
```

```
Ssec = SimTime.getSeconds();

int H2S,
   M2S,
   Sseconds;

// Constante para converter horas em segundos
H2S = 3600;
// Constante para converter minutos em segundos
M2S = 60;

// Guarda os segundos totais da hora da ordem de simular
Sseconds = Shour * H2S + Smin * M2S + Ssec;

/* Recebe a data e a hora atual em milissegundos */
long currhour = System.currentTimeMillis();
/* Transforma a data em milissegundos em objeto DATE */
Date timeref = new Date(currhour);

int hour,
   min,
   sec,
   Seconds,
```

```
    Range,
    Min,
    Max;

// Guarda a hora atual
hour = timeref.getHours();
// Guarda os minutos atuais
min = timeref.getMinutes();
// Guarda os segundos atuais
sec = timeref.getSeconds();

// Guarda os segundos totais da hora atual
Seconds = (hour * H2S) + (min * M2S) + sec;

/* Variável que limita a validade da ordem de simular nesse
  caso a 1 segundo */
Range = 10;
// Limite inferior
Min = Seconds - Range;
// Limite superior
Max = Seconds + Range;

/* Compara se a ordem de simular é válida, ou seja, foi dada
```

```
    dentro de um intervalo de 1 segundo em relação o tempo
    atual */
  //System.out.println("Ok 2");
  if(Min < Sseconds && Sseconds < Max){

    try{
      statement3 = connection.createStatement();
      resultset3 = statement3.executeQuery(query3);

      if(resultset3.next()){
        //System.out.println("Ok 3");
        // Recupera o valor do campo Simulate do BD
        Function = resultset3.getInt("Simulate");
        //System.out.println("Valor do Simulate:" + Function);
      } //if
    } //try
    catch (SQLException sqlex){
    sqlex.printStackTrace();
    } //catch
  } //if
  else
    // Retorna 33 para indicar que nenhuma ordem nova foi dada
    Function = 33;
```

```
            } //if
          } //try
          catch (SQLException sqlex){
            sqlex.printStackTrace();
          } //catch
        } //if
        else
          // Retorna 33 para indicar que nenhuma ordem nova foi dada
          Function = 33;
      } //if
      else
        // Retorna 33 para indicar que nenhuma ordem nova foi dada
        Function = 33;
    } //try
    catch (SQLException sqlex){
      sqlex.printStackTrace();
    } //catch

// Retorna o status da ordem de simular vigente
System.out.println(Function);
return Function;
}
```

```
/* Insere no banco de dados o resultado da análise do sistema supervisão
   de conforto térmico */
public void SetAmbientComfortResult(CAmbient Ambient, int Result){
  Integer Comfort = new Integer(Result);
  if(Ambient instanceof CRoom){
    long curr = System.currentTimeMillis();
    Date timeref = new Date(curr);
    Integer year = new Integer(timeref.getYear() + 1900);
    Integer month = new Integer(timeref.getMonth() + 1);
    Integer day = new Integer(timeref.getDate());
    Integer hour = new Integer(timeref.getHours());
    Integer min = new Integer(timeref.getMinutes());
    Integer sec = new Integer(timeref.getSeconds());

    String today = day.toString()+ "/" + month.toString()+ "/" +
                year.toString();

    String time = hour.toString()+ ":" + min.toString()+ ":" +
                sec.toString();

    String query = "INSERT INTO Results (EnvID, TempDifference, DDate, "+
                "TTime) VALUES(" +Ambient.GetID()+","+ Comfort.toString()+
                ",'"+ today +"','" + time +"');";
```

```
    SendQuery(query);
   }
  }
}
```

Main

```
/**
 * Title:        Sistema supervisório
 * Description:  Supervisor Class
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */


package supervisor;


import java.util.*;
import java.sql.Time;
import java.lang.*;
```

```
public class Supervisor {

 public static CDBInterface Interface;
 public static LinkedList AmbientList;

 public Supervisor() {
  // Inicialização: aqui é criada a lista de ambientes
  AmbientList = new LinkedList();
  CAmbient Ambiente;

  Interface = new CDBInterface();
  String Locais[] = {"Elevator","Room 11", "Room 12", "Sala 21", "Room 22"};
  int ids[] = {2, 11, 12, 21, 22};

  // Adicionando ambientes à lista
  for(int i=0; i<5; i++){
   if(Locais[i] == "Elevator"){
    Ambiente = new CElevator(Locais[i], ids[i]);
    AmbientList.add(Ambiente);
   } //if

   else{
    Ambiente = new CRoom(Locais[i], ids[i]);
```

```
      AmbientList.add(Ambiente);
    } //else
  } //for

} //Supervisor

public static void main(String[] args) {
 // construtor do PROGRAMA
 Supervisor supervisor1 = new Supervisor();
 int count = 0; // Variável utilizada para controlar a execução do programa

 //construtor da classe supervisor CSupervisor
 CSupervisor sup = new CSupervisor(AmbientList, Interface);

 // Execução da Supervisão
 do{
  // Recebe a ordem do simulador
  count = Interface.Simulator();
  // Simula se count for igual a 1

  if (count == 1){
    System.out.println("Start of the Supervision Program!");
    RefreshData();
```

```
      sup.run();
    } //if

    try{
      Thread.sleep(3000);  // Espera de 3 segundo
    } //try

    catch(InterruptedException ex){
      System.err.println(ex.toString());
    } //catch

  } while(count != 0);  // Para de executar o programa se count for igual a 0

  System.out.println("End of the Supervision Program!");
  Interface.finalize();  // Finaliza a conexão com o BD
  System.exit(0);

} //main

// Atualiza os dados referentes a cada ambiente
public static void RefreshData(){
  Iterator i;
  i = AmbientList.iterator();
```

```
    while(i.hasNext()){
     CAmbient Ambient = (CAmbient) i.next();

     if(Ambient instanceof CRoom)
      ((CRoom) Ambient).RefreshRoomData(Interface);

     if(Ambient instanceof CElevator)
      ((CElevator) Ambient).RefreshElevatorData(Interface);
    } //while

  } // RefreshData


} //Supervisor
```

- SIMULADOR:

SimServlet

```
/**
 * Title:        Sistema supervisório
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */


import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;
import java.sql.*;
import java.lang.*;
import java.lang.Object;



// Servlet utilizado para executar a simulação via Browser: recupera os dados inseridos
// no Browser, processa-os e gera uma página em HTML com os resultados do Supervisor.
```

```
public class SimServlet extends HttpServlet {
  // Variáveis para conexão à base de dados
  private Statement statement = null;
  private Connection connection = null;
  private String url = "jdbc:odbc:supervisao";
  private String username = "Administrador";
  private String password = "1234";

  // Conexão com a base de dados
  public void init( ServletConfig config )
    throws ServletException {

    super.init( config );

    try {
      Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
      connection = DriverManager.getConnection(url, username, password);
    } //try

    catch ( Exception e ) {
      e.printStackTrace();
      connection = null;
    } // catch
```

```
} //init

// Recebimento, processamento e geração dos resultados do sistema Supervisor
public void doPost( HttpServletRequest req, HttpServletResponse res )
  throws ServletException, IOException {

  // Variáveis para entrada de dados via Browser
  String  //Temperatura desejada para cada sala
        Temp_Des[] = {"0", "0", "0", "0"},
        //Horário Permitido de Entrada para cada sala
        I_Perm_Hour[] = {"0", "0", "0", "0", "0"},
        //Horário Permitido de Saída para cada sala
        F_Perm_Hour[] = {"0", "0", "0", "0", "0"},
        //Temperatura de cada sala
        Temp_Now[] = {"0", "0", "0", "0"},
        //Presença no ambiente
        Pres[] = {"0", "0", "0", "0", "0"},
        //Presença no ambiente se dá no momento da simulação
        Pres_Now[] = {"0", "0", "0", "0", "0"},
        //Horário da Presença no ambiente
        Pres_Hour[] = {"0", "0", "0", "0", "0"},
        //Fumaça no ambiente
```

```
        Smoke[] = {"0", "0", "0", "0"},
        //Fumaça no ambiente se dá no momento da simulação
        Smoke_Now[] = {"0", "0", "0", "0"},
        //Horário da Fumaça no ambiente
        Smoke_Hour[] = {"0", "0", "0", "0"};

    // Variáveis para receber os resultados do Supervisor
    String  //Resposta de Incêndio para cada sala
        Fire[] = {"0", "0", "0", "0"},
        //Resposta de Segurança para cada sala
        Security[] = {"0", "0", "0", "0", "0"},
        //Resposta de Conforto Térmico para cada sala
        Comfort[] = {"0", "0", "0", "0"},
        //Fumaça em cada sala
        Smoke_Sensor[] = {"0", "0", "0", "0"},
        //Temperatura em cada sala
        Temp_Sensor[] = {"0", "0", "0", "0"},
        //Presença no ambiente
        Pres_Sensor[] = {"0", "0", "0", "0", "0"};

    // Armazenamento dos dados provenientes do Browser para simulação
    Temp_Des[0] = req.getParameter( "td_11" );
    Temp_Des[1] = req.getParameter( "td_12" );
```

```
Temp_Des[2] = req.getParameter( "td_21" );
Temp_Des[3] = req.getParameter( "td_22" );
I_Perm_Hour[0] = req.getParameter( "i_p_hour_11" );
I_Perm_Hour[1] = req.getParameter( "i_p_hour_12" );
I_Perm_Hour[2] = req.getParameter( "i_p_hour_21" );
I_Perm_Hour[3] = req.getParameter( "i_p_hour_22" );
I_Perm_Hour[4] = req.getParameter( "i_p_hour_el" );
F_Perm_Hour[0] = req.getParameter( "f_p_hour_11" );
F_Perm_Hour[1] = req.getParameter( "f_p_hour_12" );
F_Perm_Hour[2] = req.getParameter( "f_p_hour_21" );
F_Perm_Hour[3] = req.getParameter( "f_p_hour_22" );
F_Perm_Hour[4] = req.getParameter( "f_p_hour_el" );
Temp_Now[0] = req.getParameter( "ta_11" );
Temp_Now[1] = req.getParameter( "ta_12" );
Temp_Now[2] = req.getParameter( "ta_21" );
Temp_Now[3] = req.getParameter( "ta_22" );
Pres[0] = req.getParameter( "pres_11" );
Pres[1] = req.getParameter( "pres_12" );
Pres[2] = req.getParameter( "pres_21" );
Pres[3] = req.getParameter( "pres_22" );
Pres[4] = req.getParameter( "pres_el" );
Pres_Now[0] = req.getParameter( "pres_atual_11" );
Pres_Now[1] = req.getParameter( "pres_atual_12" );
```

```
Pres_Now[2] = req.getParameter( "pres_atual_21" );
Pres_Now[3] = req.getParameter( "pres_atual_22" );
Pres_Now[4] = req.getParameter( "pres_atual_el" );
Pres_Hour[0] = req.getParameter( "pres_hour_11" );
Pres_Hour[1] = req.getParameter( "pres_hour_12" );
Pres_Hour[2] = req.getParameter( "pres_hour_21" );
Pres_Hour[3] = req.getParameter( "pres_hour_22" );
Pres_Hour[4] = req.getParameter( "pres_hour_el" );
Smoke[0] = req.getParameter( "smoke_11" );
Smoke[1] = req.getParameter( "smoke_12" );
Smoke[2] = req.getParameter( "smoke_21" );
Smoke[3] = req.getParameter( "smoke_22" );
Smoke_Now[0] = req.getParameter( "smoke_atual_11" );
Smoke_Now[1] = req.getParameter( "smoke_atual_12" );
Smoke_Now[2] = req.getParameter( "smoke_atual_21" );
Smoke_Now[3] = req.getParameter( "smoke_atual_22" );
Smoke_Hour[0] = req.getParameter( "s_hour_11" );
Smoke_Hour[1] = req.getParameter( "s_hour_12" );
Smoke_Hour[2] = req.getParameter( "s_hour_21" );
Smoke_Hour[3] = req.getParameter( "s_hour_22" );

// Variável que guarda os índices dos ambientes
int ids[] = {11, 12, 21, 22, 2};
```

```
String insert; //variável auxiliar

// Adicionando os dados de simulação em suas respectivas tabelas
for(int i=0; i<ids.length; i++){
  // Variáveis que guardam o dia e a hora atual
  long curr = System.currentTimeMillis();
  Date timeref = new Date(curr);
  Integer year = new Integer(timeref.getYear() + 1900);
  Integer month = new Integer(timeref.getMonth() + 1);
  Integer day = new Integer(timeref.getDate());
  Integer hour = new Integer(timeref.getHours());
  Integer min = new Integer(timeref.getMinutes());
  Integer sec = new Integer(timeref.getSeconds());

  String today = day.toString()+ "/" + month.toString()+ "/" +
        year.toString();

  String time = hour.toString()+ ":" + min.toString()+ ":" +
        sec.toString();

  if(ids[i] == 2){
    // Inserção na tabela UserInterface
    insert = acessDB(1, "UserInterface (EnvID, InicialPermTime, EndPermTime, "
```

```
    + "DDate, TTime) values (" + ids[i] + ",'" + I_Perm_Hour[i] +
     "','" + F_Perm_Hour[i] + "', '" + today + "', '" + time + "');" , "");
  } //if

  else{
    // Inserção na tabela UserInterface
    insert = acessDB(1, "UserInterface values (" + ids[i] + "," + Temp_Des[i] +
                  ",'" + I_Perm_Hour[i] + "','" + F_Perm_Hour[i] + "', '" +
                  today + "', '" + time + "');" , "");

    // Inserção na tabela TempSensor
    insert = acessDB(1, "TempSensor values (" + ids[i] + "," + Temp_Now[i] +
                  ",'" + today + "', '" + time + "');" , "");

    // Inserção na tabela SmokeSensor: com hora atual
    if (Smoke_Now[i].equalsIgnoreCase("1"))
      insert = acessDB(1, "SmokeSensor values (" + ids[i] + "," + Smoke[i] +
                    ",'" + today + "', '" + time + "');" ,"");

  // Inserção na tabela SmokeSensor: com hora escolhida pelo usuário
  else
    insert = acessDB(1, "SmokeSensor values (" + ids[i] + "," + Smoke[i] +
                  ",'" + today + "', '" + Smoke_Hour[i] + "');" , "");
```

```
    } //else

    // Inserção na tabela PresenceSensor: com hora atual
    if (Pres_Now[i].equalsIgnoreCase("1"))
      insert = acessDB(1, "PresenceSensor values (" + ids[i] + "," + Pres[i] +
                  ",'" + today + "', '" + time + "');" , "");

    // Inserção na tabela PresenceSensor: com hora escolhida pelo usuário
    else
      insert = acessDB(1, "PresenceSensor values (" + ids[i] + "," + Pres[i] +
                  ",'" + today + "', '" + Pres_Hour[i] + "');" , "");

  } //for

  // Iniciando o processamento do Supervisor
  long curr = System.currentTimeMillis();
    Date timeref = new Date(curr);
    Integer year = new Integer(timeref.getYear() + 1900);
    Integer month = new Integer(timeref.getMonth() + 1);
    Integer day = new Integer(timeref.getDate());
    Integer hour = new Integer(timeref.getHours());
    Integer min = new Integer(timeref.getMinutes());
    Integer sec = new Integer(timeref.getSeconds());
```

```
String today = day.toString()+ "/" + month.toString()+ "/" +
        year.toString();

String time = hour.toString()+ ":" + min.toString()+ ":" +
        sec.toString();

// Envia ao BD a ordem para iniciar o Supervisor
insert = acessDB(1, "Simulator values ( 1, '" + today + "', '" +
        time + "');" ,"");

// Espera de 10 segundo para que o Supervisor processe todas as informaçoes
try{
  Thread.sleep(10000);
} //try

catch(InterruptedException ex){
  System.err.println(ex.toString());
} //catch

// Verificação dos resultados da Supervisão
for(int i=0; i<ids.length; i++){
```

```
// Verificaçoes dos sistemas que não cobrem o elavador
if(ids[i] != 2){
  // Verificação de Incêndio
  Fire[i] = acessDB(2, " FireAlarm FROM (SELECT TOP 1 TTime,"+
            " FireAlarm FROM (SELECT TTime, FireAlarm FROM"+
            " Results WHERE DDate = (SELECT TOP 1 DDate FROM"+
            " (SELECT DDate FROM Results ORDER BY DDate DESC)"+
            " WHERE EnvID = "+ ids[i] +")) WHERE FireAlarm IS"+
            " NOT NULL ORDER BY TTime  DESC );", "FireAlarm");

  // Se os sensores estão funcionando, faz-se a leitura dos valores dos mesmos
  if(Fire[i] != "33"){
    //Leitura do sensor de Fumaça de cada sala
    Smoke_Sensor[i] = acessDB(2, " TOP 1 Smoke FROM (SELECT Smoke, Time
FROM "+
                  "(SELECT Smoke, Date, Time FROM SmokeSensor"+
                  " WHERE EnvID =" + ids[i] + ") WHERE "+
                  "Date = (SELECT TOP 1 Date FROM (SELECT "+
                  "Smoke, Date, Time FROM SmokeSensor"+
                  " WHERE EnvID = " + ids[i] +
                  " ORDER BY Date DESC)) ORDER BY Time "+
                  "Desc);", "Smoke");
```

```
        //Leitura do sensor de Temperatura Atual de cada sala
        Temp_Sensor[i] = acessDB(2, " TOP 1 Temp FROM (SELECT Temp, Time
FROM "+
                                "(SELECT Temp, Date, Time FROM TempSensor"+
                                " WHERE EnvID =" + ids[i] + ") WHERE "+
                                "Date = (SELECT TOP 1 Date FROM (SELECT "+
                                "Temp, Date, Time FROM TempSensor"+
                                " WHERE EnvID = " + ids[i] +
                                " ORDER BY Date DESC)) "+
                                " ORDER BY Time Desc);", "Temp");
    } //if

    // Verificação de Conforto Térmico
    Comfort[i] = acessDB(2, " TempDifference FROM (SELECT TOP 1 "+
                        "TTime, TempDifference FROM (SELECT TTime, "+
                        "TempDifference FROM Results WHERE DDate"+
                        " = (SELECT TOP 1 DDate FROM (SELECT DDate "+
                        "FROM Results ORDER BY DDate DESC) WHERE EnvID = "+
                        ids[i] +")) WHERE TempDifference IS NOT NULL"+
                        " ORDER BY TTime  DESC );", "TempDifference");

} // if
```

```
// Verificaçoes dos sistemas que cobrem também o elavador
Security[i] = acessDB(2, " IntrusionAlarm FROM (SELECT TOP 1 TTime,"+
                " IntrusionAlarm  FROM (SELECT TTime, IntrusionAlarm FROM"+
                " Results WHERE DDate = (SELECT TOP 1 DDate FROM"+
                " (SELECT DDate FROM Results ORDER BY DDate DESC)"+
                " WHERE EnvID = "+ ids[i] +")) WHERE IntrusionAlarm IS"+
                " NOT NULL ORDER BY TTime  DESC );", "IntrusionAlarm");

// Se o sensor de presença está funcionando, faz-se a leitura do valore do mesmo
if(Security[i] != "33")
  Pres_Sensor[i] = acessDB(2, " TOP 1 Presence FROM (SELECT Presence, "+
                    "Time FROM (SELECT Presence, Date, Time FROM "+
                    "PresenceSensor WHERE EnvID =" + ids[i] +
                    ") WHERE Date = (SELECT TOP 1 Date FROM (SELECT"+
                    " Presence, Date, Time FROM PresenceSensor"+
                    " WHERE EnvID = " + ids[i] + " ORDER BY Date "+
                    "DESC)) ORDER BY Time Desc);", "Presence");

}

int j; // Variável auxiliar

// Construção da página em HTML com as respostas do Supervisor
```

```
PrintWriter responseOutput = res.getWriter();
StringBuffer buf = new StringBuffer();

buf.append("<html> \n");
buf.append("<head> \n");
buf.append("<title>Supervisor's Report</title> \n");
buf.append("<meta http-equiv=" + "Content-Type " + "content=" +
        "text/html; charset=iso-8859-1" + "> \n");
buf.append("</head> \n");

buf.append("<body bgcolor=" + "#000000" + "> \n");
buf.append("<table width=" + "750 " + "border=" + "1 " + "align=" +
        "center " + "cellpadding=" + "0" + " cellspacing=" + "0" +
        " bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<tr> \n");
buf.append("<td><table width=" + "750" + " border="+ "0" + " cellspacing="
        + "0" + " cellpadding=" + "0" + "> \n");
buf.append("<tr> \n");
buf.append("<td width=" + "150 " + "><img src=" + "../figuras/shim.gif" +
        " width=" + "1" + " height=" + "1" + "></td> \n");
buf.append("<td align=" + "center" + " valign=" + "top" + "><div align="
        + "center" + "><img src=" + "../figuras/report.jpg " + " width="
        + "400" + " height=" + "50" + "></div></td> \n");
```

```
buf.append("<td width=" + "150" + "><div align=" + "center" + "> \n");
buf.append("<table width=" + "150" + " height=" + "50" + " border=" + "0"
        + " cellpadding=" + "0" + " cellspacing=" + "0" + "> \n");
buf.append("<tr> \n");
buf.append("<td><img src=" + "../figuras/shim.gif " + " width=" + "1" +
        " height=" + "1" + "></td> \n");
buf.append("</tr> \n");
buf.append("<tr> \n");
buf.append("<td align=" + "right" + " valign=" + "bottom" + "><table width="
        + "100%" + " border=" + "1" + " cellpadding=" + "0" +
        " cellspacing=" + "0" + " bordercolor=" + "#CCCCCC" + "> \n");
buf.append("<tr> \n");
buf.append("<td bgcolor=" + "#999999" + "><div align=" + "right" +
        "><font color=" + "#FFFFFF" + " size=" + "1" + " face=" +
        "Verdana, Arial, Helvetica, sans-serif" +
        "><em>"+ today +" - "+ time +"</em></font></div></td> \n");
buf.append(" </tr> \n");
buf.append("</table></td> \n");
buf.append("</tr> \n");
buf.append("</table> \n");
buf.append("</div></td> \n");
buf.append("</tr> \n");
buf.append("</table></td> \n");
```

```
buf.append(" </tr> \n");
buf.append("<tr> \n");
buf.append("<td><table width=" + "750" + " border=" + "1" + " cellpadding="
        + "0" + " cellspacing=" + "0" +
        " bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><img src=" + "../figuras/shim.gif" + " width=" + "1" +
        " height=" + "1" + "></td> \n");
buf.append("<td width=" + "110" + "><div align=" + "center" + "><font color="
        + "#666666" + "><em><strong>Sala11</strong></em></font></div></td> \n");
buf.append("<td width="+ "110" + "><div align=" + "center" + "><font color="
        + "#666666" + "><em><strong>Sala12</strong></em></font></div></td> \n");
buf.append("<td width=" + "110" + "><div align=" + "center" + "><font color="
        + "#666666" + "><em><strong>Sala21</strong></em></font></div></td> \n");
buf.append("<td width=" + "110" + "><div align=" + "center" + "><font color="
        + "#666666" + "><em><strong>Sala22</strong></em></font></div></td> \n");
buf.append("<td width=" + "110" + "><div align=" + "center" + "><font color="
        + "#666666" + "><em><strong>Elevador</strong></em></font></div></td>
\n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td width=" + "200" + " height=" + "20" + "><img src="
        + "../figuras/supervisores.jpg" + " width=" + "200" + " height="
```

```
        + "20" + "></td> \n");
    buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + " bgcolor=" + "#CCCCCC" + "><div align=" + "center" + "><img src="
        + "../figuras/shim.gif" + " width=" + "1" + " height=" + "1"
        + "></div></td> \n");
    buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + " bgcolor=" + "#CCCCCC" + "><div align=" + "center" + "><img src="
        + "../figuras/shim.gif" + " width=" + "1" + " height=" + "1"
        + "></div></td> \n");
    buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + " bgcolor=" + "#CCCCCC" + "><div align=" + "center" +
        "><img src=" + "../figuras/shim.gif" + " width=" + "1" + " height="
        + "1" + "></div></td> \n");
    buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + " bgcolor=" + "#CCCCCC" + "><div align=" + "center" +
        "><img src=" + "../figuras/shim.gif" + " width=" + "1" + " height="
        + "1" + "></div></td> \n");
    buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + " bgcolor=" + "#CCCCCC" + "><div align=" + "center" + "><img src="
        + "../figuras/shim.gif" + " width=" + "1" + " height=" + "1"
        + "></div></td> \n");
    buf.append("</tr> \n");
    buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
```

```
buf.append("<td><font size=" + "2" +
        " face=" + "Verdana, Arial, Helvetica, sans-serif" +
        ">- Inc&ecirc;ndio:</font></td> \n");

// Incêndio: verifica se deve-se ou não mostrar alarme ou um erro no processo
for(j=0 ; j<4 ; j++){
if(Fire[j].equalsIgnoreCase("1"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + "><div align=" + "center" + "><img src="
        + "../figuras/alarme.gif" + " width=" + "110" + " height=" + "50"
        + "></div></td> \n");
else if(Fire[j].equalsIgnoreCase("0"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + "><div align=" + "center" + "><img src="
        + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
        + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + "><div align=" + "center" + "><img src="
        + "../figuras/erro.gif" + " width=" + "110" + " height=" + "50"
        + "></div></td> \n");
} //for
```

```
buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + " bgcolor=" + "#CCCCCC" + "><div align="
        + "center" + "><img src=" + "../figuras/shim.gif" + " width="
        + "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><font size=" + "2" + " face=" + "Verdana, Arial, Helvetica, sans-serif"
        + ">- Seguran&ccedil;a:</font></td> \n");

// Segurança: verifica se deve-se ou não mostrar alarme ou um erro no processo
for(j=0 ; j<5 ; j++){
if(Security[j].equalsIgnoreCase("1"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + "><div align=" + "center" + "><img src="
        + "../figuras/invasao.gif" + " width=" + "110" + " height=" + "50"
        + "></div></td> \n");
else if(Security[j].equalsIgnoreCase("0"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + "><div align=" + "center" + "><img src="
        + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
        + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
```

```
            + " valign=" + "middle" + "><div align=" + "center" + "><img src="
            + "../figuras/erro.gif" + " width=" + "110" + " height=" + "50"
            + "></div></td> \n");
    } //for

    buf.append("</tr> \n");
    buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
    buf.append("<td><font size=" + "2" + " face=" +
            "Verdana, Arial, Helvetica, sans-serif"
            + ">- Conforto T&eacute;rmico:</font></td> \n");

    // Conforto Térmico: verifica que indicador mostrar ou erro no processo
    for(j=0 ; j<4 ; j++){
    if(Comfort[j].equalsIgnoreCase("1"))
       buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
            + " valign=" + "middle" + "><div align=" + "center" + "><img src="
            + "../figuras/acima.gif" + " width=" + "110" + " height=" + "50"
            + "></div></td> \n");
    else if(Comfort[j].equalsIgnoreCase("0"))
       buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
            + " valign=" + "middle" + "><div align=" + "center" + "><img src="
            + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
            + "></div></td> \n");
```

```
else if(Comfort[j].equalsIgnoreCase("-1"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/abaixo.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/erro.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
} //for

buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
      + " valign=" + "middle" + " bgcolor=" + "#CCCCCC" + "><div align="
      + "center" + "><img src=" + "../figuras/shim.gif" + " width="
      + "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#CCCCCC" + "> \n");
buf.append("<td height=" + "20" + "><img src=" + "../figuras/sensores.jpg" +
      " width=" + "200" + " height=" + "20" + "></td> \n");
buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
      + "><div align=" + "center" + "><img src=" + "../figuras/shim.gif"
      + " width=" + "1" + " height=" + "1" + "></div></td> \n");
```

```
buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + "><div align=" + "center" + "><img src=" + "../figuras/shim.gif"
        + " width=" + "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + "><div align=" + "center" + "><img src=" + "../figuras/shim.gif"
        + " width=" + "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + "><div align=" + "center" + "><img src=" + "../figuras/shim.gif"
        + " width=" + "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td height=" + "20" + " align=" + "center" + " valign=" + "middle"
        + "><div align=" + "center" + "><img src=" + "../figuras/shim.gif"
        + " width=" + "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><font size=" + "2" + " face=" +
        "Verdana, Arial, Helvetica, sans-serif"
        + ">- Fuma&ccedil;a:</font></td> \n");

// Valor do Sensor de Fumaça
for(j=0 ; j<4 ; j++){
if(Fire[j].equalsIgnoreCase("33"))
  buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
        + " valign=" + "middle" + "><div align=" + "center" + "><img src="
```

```
            + "../figuras/erro.gif" + " width=" + "110" + " height=" + "50"
            + "></div></td> \n");
    else if(Smoke_Sensor[j].equalsIgnoreCase("0"))
       buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
            + " valign=" + "middle" + "><div align=" + "center" + "><img src="
            + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
            + "></div></td> \n");
    else
       buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
            + " valign=" + "middle" + "><div align=" + "center" + "><img src="
            + "../figuras/fumaca.gif" + " width=" + "110" + " height=" + "50"
            + "></div></td> \n");
    } //for

    buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
          + " valign=" + "middle" + " bgcolor=" + "#CCCCCC" + "><div align="
          + "center" + "><img src=" + "../figuras/shim.gif" + " width="
          + "1" + " height=" + "1" + "></div></td> \n");
    buf.append("</tr> \n");
    buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
    buf.append("<td><font size=" + "2" + " face=" +
          "Verdana, Arial, Helvetica, sans-serif"
          + ">- Temperatura (Atual):</font></td> \n");
```

```
// Valor da Temperatura Atual de cada Sala
for(j=0 ; j<4 ; j++){
if(Fire[j].equalsIgnoreCase("33") || Comfort[j].equalsIgnoreCase("33"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/erro.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + ">"+
         Temp_Sensor[j] +"</div></td> \n");
}// for

buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
      + " valign=" + "middle" + " bgcolor=" + "#CCCCCC" + "><div align="
      + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
      "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><font size=" + "2" + " face=" +
      "Verdana, Arial, Helvetica, sans-serif" +
      ">- Presen&ccedil;a:</font></td> \n");
```

```
// Valor do Sensor de Presença
for(j=0 ; j<5 ; j++){
if(Security[j].equalsIgnoreCase("33"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/erro.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else if(Pres_Sensor[j].equalsIgnoreCase("0"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/vazia.jpg" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/ocupada.jpg" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
} //for

buf.append("</tr>\n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#CCCCCC" + "> \n");
buf.append("<td height=" + "40" + "><img src=" + "../figuras/status.jpg" +
```

```
        " width=" + "200" + " height=" + "40" + "></td> \n");
buf.append("<td align=" + "center" + " valign=" + "middle" + "><div align="
        + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
        "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td align=" + "center" + " valign=" + "middle" + "><div align="
        + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
        "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td align=" + "center" + " valign=" + "middle" + "><div align="
        + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
        "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td align=" + "center" + " valign=" + "middle" + "><div align="
        + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
        "1" + " height=" + "1" + "></div></td> \n");
buf.append("<td align=" + "center" + " valign=" + "middle" + "><div align="
        + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
        "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><font size=" + "2" + " face=" +
        "Verdana, Arial, Helvetica, sans-serif" +
        ">- Fuma&ccedil;a:</font></td> \n");

// Status do Sensor de Fumaça
```

```
for(j=0 ; j<4 ; j++){
if(Fire[j].equalsIgnoreCase("33"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/defeito.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
} //for

buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
      + " valign=" + "middle" + " bgcolor=" + "#CCCCCC" + "><div align="
      + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
      "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><font size=" + "2" + " face=" +
      "Verdana, Arial, Helvetica, sans-serif" +
      ">- Temperatura:</font></td> \n");
```

```
// Status do Sensor de Temperatura
for(j=0 ; j<4 ; j++){
if(Fire[j].equalsIgnoreCase("33") || Comfort[j].equalsIgnoreCase("33"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/defeito.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
} //for

buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
      + " valign=" + "middle" + " bgcolor=" + "#CCCCCC" + "><div align="
      + "center" + "><img src=" + "../figuras/shim.gif" + " width=" +
      "1" + " height=" + "1" + "></div></td> \n");
buf.append("</tr> \n");
buf.append("<tr bordercolor=" + "#CCCCCC" + " bgcolor=" + "#FFFFFF" + "> \n");
buf.append("<td><font size=" + "2" + " face=" +
      "Verdana, Arial, Helvetica, sans-serif" +
      ">- Presen&ccedil;a:</font></td> \n");
```

```
// Status do Sensor de Presença
for(j=0 ; j<5 ; j++){
if(Security[j].equalsIgnoreCase("33"))
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/defeito.gif" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
else
   buf.append("<td width=" + "110" + " height=" + "50" + " align=" + "center"
         + " valign=" + "middle" + "><div align=" + "center" + "><img src="
         + "../figuras/ok.jpg" + " width=" + "110" + " height=" + "50"
         + "></div></td> \n");
} //for

buf.append("</tr> \n");
buf.append("</table></td> \n");
buf.append("</tr> \n");
buf.append("</table> \n");
buf.append("</body> \n");
buf.append("</html> \n");

responseOutput.println( buf.toString());
```

```
    responseOutput.close();

  } //doPost

  private String acessDB(int query_type, String string, String Column ){
    String Result = "Null";
    ResultSet resultset;
    try {
      statement = connection.createStatement();

      if(query_type == 1){
        statement.execute("INSERT INTO " + string);
        statement.close();
        Result = "Insert Ok";
      } //if

      else{
        resultset = statement.executeQuery("SELECT " + string);

        if(resultset.next())
          Result = resultset.getString(Column);

        else {
```

```
            System.out.println("Supervisor: there is no related value!");
            System.exit(1);
            Result = "Select Failed";
          } //else

        } //else
      } //try

      catch ( Exception e ) {
        System.err.println(
          "ERROR: Problems with new entry" );
        e.printStackTrace();
        Result = "Consult Failed";
      } //catch

      return Result;

  } //acessDB

  public void destroy() {

    try {
      connection.close();
```

```
    } //try

    catch( Exception e ) {
      System.err.println( "Problem closing the database" );
    } //catch
  } //destroy

} //SimServlet
```

StopServlet

```
/**
 * Title:        Sistema supervisório
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:      Escola Politécnica - USP
 * @author Fábio A. Narita
 * @version 1.0
 */

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.util.Date;
import java.sql.*;
import java.lang.*;
import java.lang.Object;


// Servlet usado para parar o programa. Isso é feito através da inserção do valor
// '0' na tabela 'Simulator', fazendo com que o programa principal (Supervisor)
// pare a sua execução.

public class StopServlet extends HttpServlet {
  // Variáveis utilizadas para conexão à base de dados
  private Statement statement = null;
  private Connection connection = null;
  private String url = "jdbc:odbc:supervisao";
  private String username = "Administrador";
  private String password = "1234";

  // Conexão à base de dados
  public void init( ServletConfig config )
    throws ServletException {

    super.init( config );
```

```
    try {
      Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
      connection = DriverManager.getConnection(url, username, password);
    } //try

    catch ( Exception e ) {
      e.printStackTrace();
      connection = null;
    } //catch

  } //init


  public void doPost( HttpServletRequest req, HttpServletResponse res )
    throws ServletException, IOException {

    PrintWriter output = res.getWriter();
    res.setContentType ( "text/html" );

    boolean success = insertIntoDB();

    if ( success )
```

```
      output.print( "<H2>End of the Supervisor's Program.</H2>" );
    else
      output.print( "<H2>An error occurred. " +
                "Please try again later.</H2>" );
    output.close();

  } //doPost


  // Função que realiza a inserção na base de dados
  private boolean insertIntoDB() {
    try {
      statement = connection.createStatement();
      // Variáveis utilizadas para pegar a data e hora atual
      long curr = System.currentTimeMillis();
      Date timeref = new Date(curr);
      Integer year = new Integer(timeref.getYear() + 1900);
      Integer month = new Integer(timeref.getMonth() + 1);
      Integer day = new Integer(timeref.getDate());
      Integer hour = new Integer(timeref.getHours());
      Integer min = new Integer(timeref.getMinutes());
      Integer sec = new Integer(timeref.getSeconds());
```

```
        String today = day.toString()+ "/" + month.toString()+ "/" +
                year.toString();

        String time = hour.toString()+ ":" + min.toString()+ ":" +
                sec.toString();

        statement.execute("INSERT INTO Simulator VALUES (0, '" + today + "', '"
                + time + "');");
        statement.close();
      } //try

      catch ( Exception e ) {
        System.err.println(
          "ERROR: Problems with adding new entry" );
        e.printStackTrace();
        return false;
      } //catch

      return true;
    } //insertIntoDB

    // Finaliza a conexão à base de dados
    public void destroy() {
```

```
    try {
      connection.close();
    } //try

    catch( Exception e ) {
      System.err.println( "Problem closing the database" );
    } //catch

  } //destroy

} //StopServlet
```

Bibliografia

- SILVA, J.R. **Interactive Design of Integrated Systems**.
- QUATRANI, T. **Visual Modeling with Rational Rose 2000 and UML**, Addison-Wesley, 2000.
- JACOBSON, I. ; BOOCH, G. ; RUMBAUGH, J **The Unified Software evelopment Process, Rational Software Corporation**, 1998.
- SILVA, J. R. ; FOYO, P. M. G. **Sistema Supervisório Descentralizado Baseado en Íntegrons**, Escola Politécnica, Universidade de São Paulo.
- SILVA, J. R. ; RAMOS, R. L. C. B. ; MIYAGI, P. E. **Supervisory Control of Integrated Building Systems:** A Balanced Approach. Escola Politécnica, Universidade de São Paulo.
- DEITEL, H. M. ; DEITEL, P. J. **Java, Como Programar**. 3a. edição, Bookman, 2001.
- ABBEY, M. ; COREY, M. J. ; ABRAMSON, I. **Oracle 8i, Guia Introdutório**. Editora Campus Ltda., 2000.
- Vernadet, F. **Enterprise Modelling and Integration Principles and Applications**, Chapman & Hall, 1996.
- http://www.domotica.com.br/
- http://www.aureside.org.br/
- http://www.camaradearquitetos.com.br/
- http://java.sun.com/docs/books/tutorial/
- http://www.borland.com/jbuilder