Instituto Universitário de Lisboa

MESTRADO EM ENGENHARIA DE TELECOMUNICAÇÕES E INFORMÁTICA

COMPUTAÇÃO GRÁFICA

Health Monitor

Sistema de monitoria remota de sinais vitais
de pacientes em ambientes domiciliar e hospitalar
através da Internet

Lisboa, 10/06/2014

Grupo

Rui Pereira	Nº 63421
João Guiomar	Nº 50508
Júlio Ribeiro	Nº 63828

Docentes

Prof. Pedro Santana

Índice

Objetivo

O objetivo deste projeto é o de conceber um sistema para a monitoria de sinais vitais através da Internet, em um ambiente virtualizado através de recursos gráficos baseado no *standard* X3D e de uma aplicação móvel desenvolvida em Android.

O desafio se põe a partir do desenvolvimento de uma plataforma de sistemas embebidos, baseada em placas Arduino e *transceivers* XBee, e a integração desta com uma plataforma web desenvolvida com recurso a linguagens de programação de mais alto nível (PHP, *JavaScript*, Android e X3D) por forma a demonstrar, de forma articulada e igualmente desafiante, a aplicação de conceitos em 3 disciplinas do curso de Mestrado em Engenharia de Telecomunicações e Informática: Sistemas Embebidos, Desenvolvimento de Aplicações Móveis, Computação Gráfica.

A motivação para este projeto é a expectativa de explorar o quão abrangente podem ser as aplicações em sistemas embebidos – ambientes que geralmente impõem restrições quanto à disponibilidade de recursos computacionais – quando estão na base de um sistema muito maior e de notável utilidade cotidiana, quer seja no âmbito da exploração comercial, quer seja no desenvolvimento de soluções para consumo próprio utilizando tecnologias *opensource*.

Neste documento que se segue, serão apresentados os princípios construtivos de cada parte do sistema, com ênfase na disciplina de Sistemas Embebidos, bem como serão apresentados esquemas de ligação e excertos do código utilizado para

a programação dos microcontroladores Atmega, do *webserver* em PHP, da aplicação Android e finalmente do código *XML based* que está na origem da programação X3D.

Metodologia

Foi idealizado um desafio que cumprisse basicamente 3 critérios:

1. Tivesse na base da sua construção um sistema baseado em microcontrolador de 8 bits que interagisse com variáveis do mundo físico e com particular interesse em serem monitorizadas remotamente;
2. Que o sistema pudesse ser construído utilizando *devices* baseados em placas Arduino ® e sensores de baixo custo;
3. Que fosse possível estabelecer comunicação wireless entre os *devices,* através de *transceivers* XBee, e que pelo menos 1 dos *devices* pudesse transmitir dados para um servidor HTTP (local ou na "*cloud"*)

Após discussão em grupo, optou-se pelo desenvolvimento de um sistema para a monitoria de sinais vitais de uma pessoa (ritmo cardíaco, temperatura corporal e movimentos bruscos/quedas) denominado "***Health Monitor***".

O passo seguinte consistiu em desenhar um diagrama em blocos para representar os principais elementos, por forma a se visualizar a arquitetura macro do sistema.

De posse da arquitetura macro, procurou-se especificar funcionalmente cada bloco do sistema para finalmente chegar à lista de funcionalidades e comportamentos espectáveis da solução.

O desenvolvimento da componente no ambiente Arduino ® foi feito utilizando o IDE Arduino versão 1.0.5 **[1].**

Para o ambiente para testes e programação dos *transceivers* XBee foi utilizado o *software* XCTU versão 6.1.0 **[2]**

O desenvolvimento da componente no ambiente Android foi feito utilizando o IDE Android Development Studio versão v22.3.0v2013 **[3]** para Android *platform* 4.4 (API 19). Também foi utilizado o ambiente para desenvolvimento ágil desenvolvido pelo MIT, *App Inventor 2* **[4].**

O desenvolvimento da componente web e base de dados foi feito utilizando um *web hosting free* em www.awardspace.net, o qual disponibiliza um servidor PHP versão 5.3.22 e um servidor MySQL versão 5.1 **[5].**

O desenvolvimento do ambiente gráfico virtual foi feito utilizando o editor X3D-Edit 3.3 **[6].**

Diagrama em blocos da solução

O diagrama em blocos do sistema segue abaixo indicado

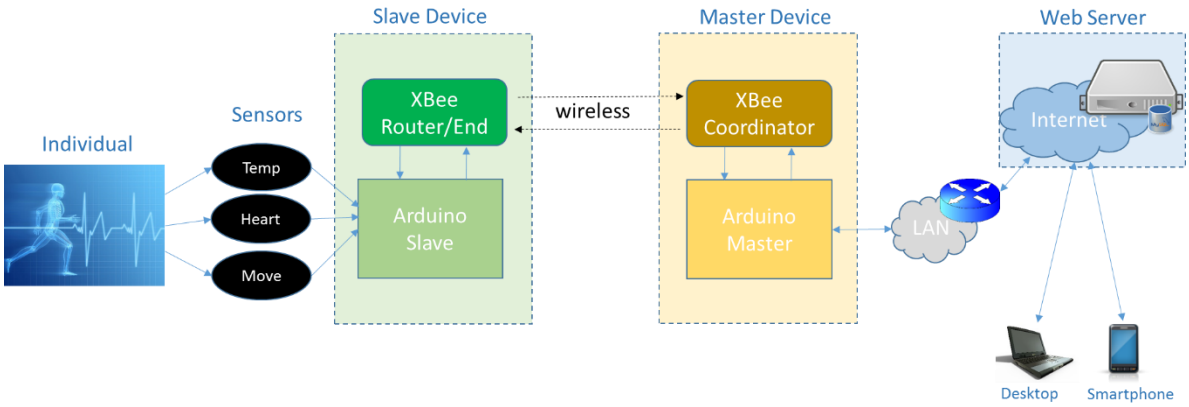

Figura 1: Diagrama de blocos da solução.

Individual: Pessoa / paciente a ser monitorizado.

Sensors: Sensores em contacto com o corpo do paciente:

- *Temp*: Sensor de temperatura TMP36 **[7]**
- *Heart*: Sensor de batimentos cardíaco Pulse Sensor SEN-11574 **[8]**
- *Move:* Acelerômetro 3-axis ADXL345 **[9]**

Slave Device: Dispositivo móvel portado pelo indivíduo, composto por:

- *Arduino Slave*: Arduino UNO ATmega 328P **[10]**
- *XBee Router/End:* Módulo XBee-Pro configurado como Router API ou End Device API **[11]**

Master Device: Dispositivo coordenador da rede de sensores e responsável pela comunicação com o Web Server através da Internet.

- *Arduino Master:* Arduino Ethernet **[12]**
- *XBee Coordinator:* Módulo XBee-Pro configurado como Coordinator API **[11]**

Web Server: Servidor PHP com MySQL alojado em um provedor *free web hosting* para armazenamento e consulta dos dados recolhidos da rede de sensores. *Server side script* criado em PHP para submeter e consultar dados na base de dados. **[13].**

Desktop: Estação cliente com *browser* ou outro software que permita a renderização dos gráficos X3D com o modelo do serviço ambulatório possibilitando a visualização dos dados transmitidos pelo Master Device.

Smartphone: Terminal *mobile* Android com aplicação desenvolvida para consultar e mostrar os dados transmitidos pelo Master Device bem como outros dados relevantes de cada paciente (diagnóstico, background, medicação e horário, etc.).

Web Server

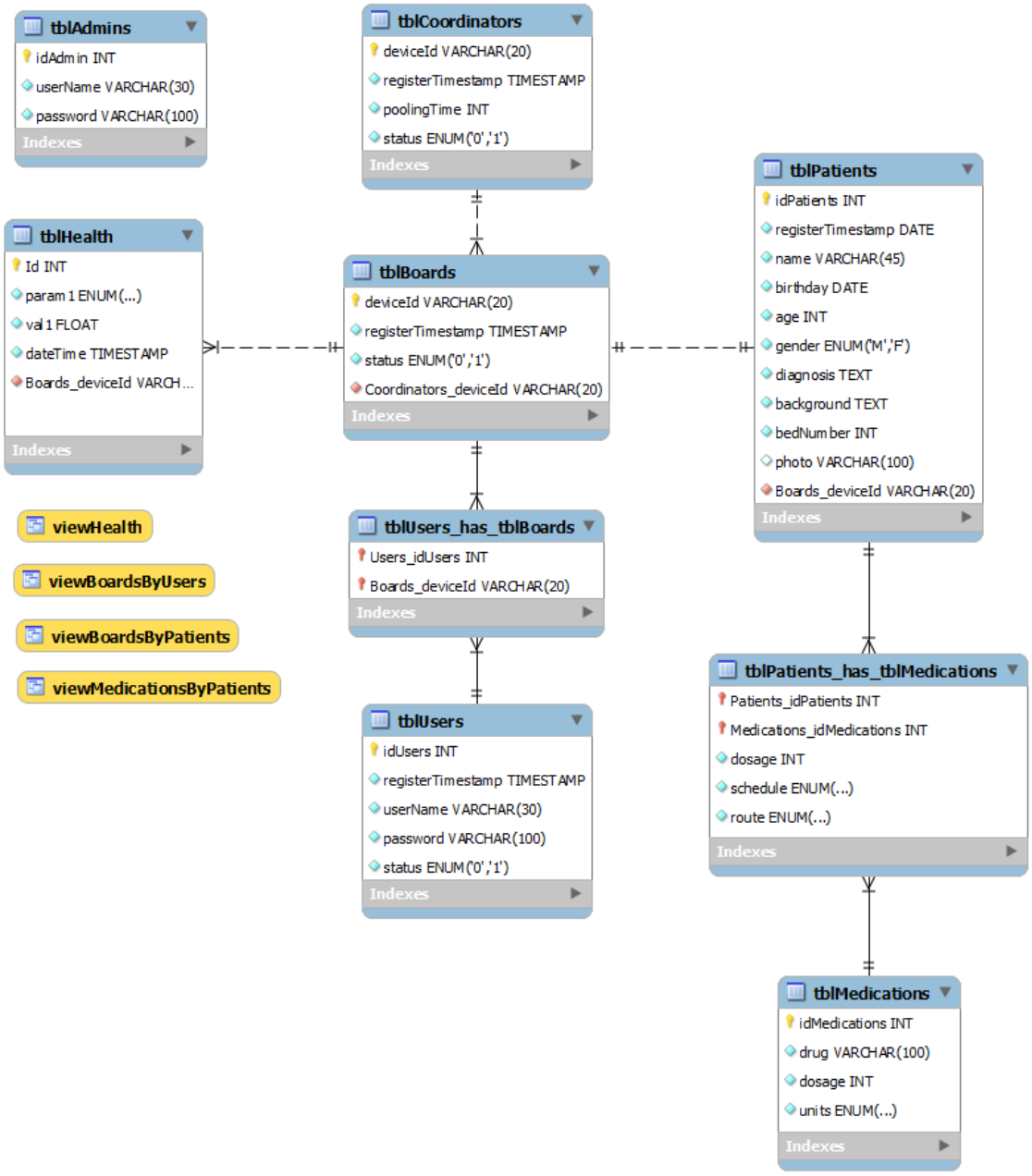

Figura 2: Estrutura de Dados

A estrutura de dados (figura 2) é composta por cinco tabelas relacionadas principais:

tblCoordinators: Regista a identificação das placas Arduino que funcionam como coordenadores (a que chamamos a este nível de *coordinators*) e que têm permissão para introduzir informação na base de dados.

tblBoards: Regista a identificação das placas Arduino que são coordenadas (a que chamamos a este nível de *boards*). São estas placas as que leem a informação dos sensores.

tblHealth: Regista os valores lidos e processados pelas *boards* para cada sensor em cada *timestamp*.

tblUsers: Regista a identificação bem como as credenciais de acesso, de cada utilizador (profissional de saúde, por exemplo) relacionando-o com uma ou múltiplas *boards*. Um utilizador apenas consegue observar os valores relacionados com as *boards* que lhe estão associadas.

tblPatients: Regista a identificação bem como outros dados relevantes de cada paciente. Cada paciente encontra-se relacionado com uma e apenas uma *board* que regista os seus valores vitais (neste momento apenas estão a ser recolhidos a temperatura e os batimentos cardíacos).

Para gerir e registar estes dados e relações foi criado um *website* simples (figura 3) acedido apenas pelos utilizadores registados como administradores (ver tabela tblAdmin da figura 2).

Este website apresenta quatro páginas principais (Boards, Users, Pharmacy e Patients) para registar e visualizar os dados contidos na base de dados.
De referir ainda a existência de uma quinta página para efeitos de teste e desenvolvimento. Esta permite “simular” a introdução valores pelos sensores apenas para efeitos de teste.

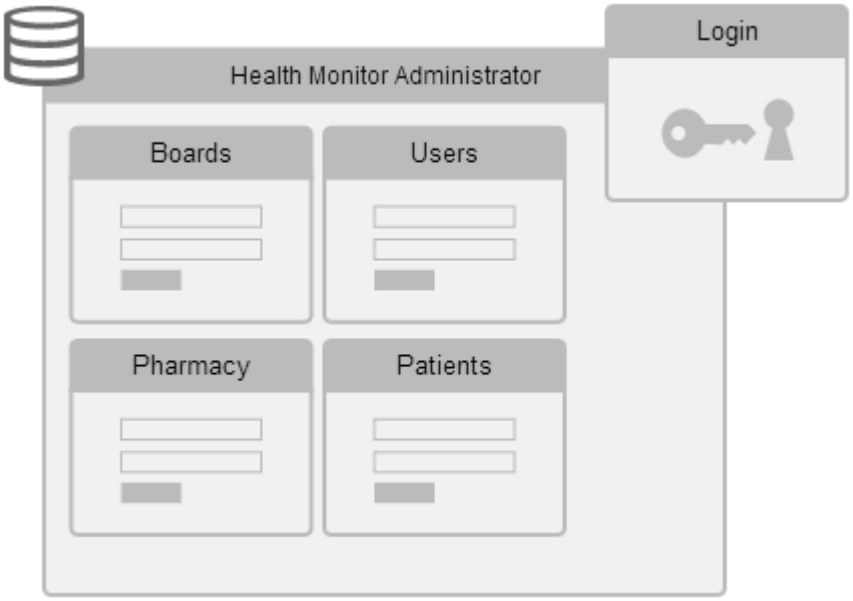

Figura 3: Health Monitor Administrator

Desktop

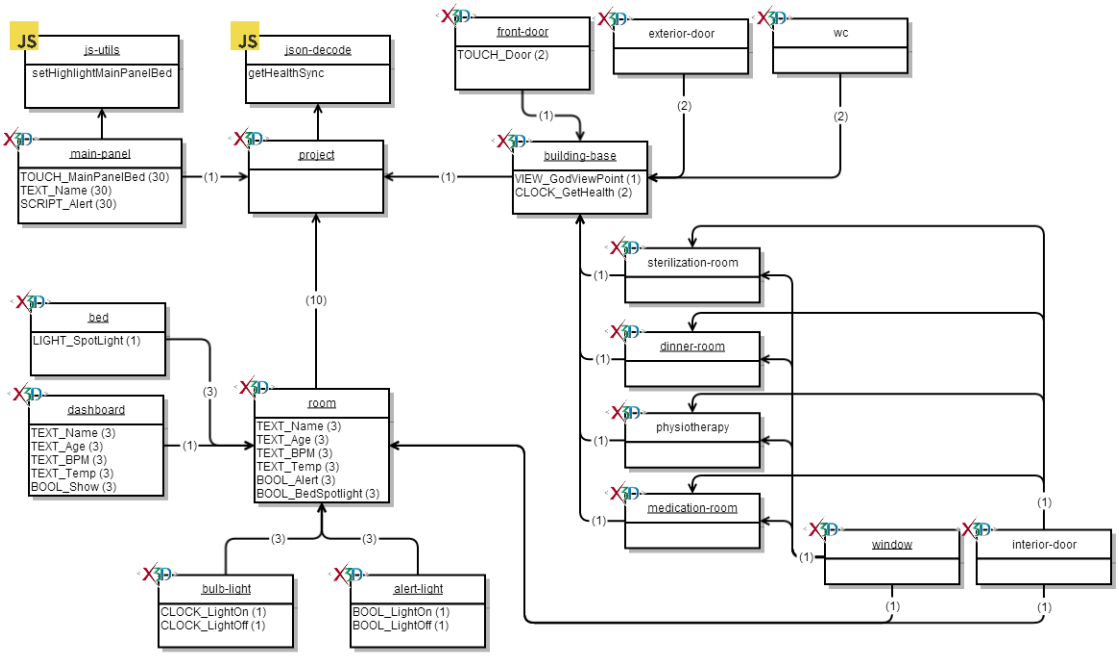

Figura 4: X3D Client

Foi desenvolvido um interface que se apresenta como um modelo que representa o interior de um ambulatório num hospital, composto por quartos destinado a pacientes em observação médica. O modelo é baseado numa planta real de um serviço ambulatório e a interação com os quartos e camas apresenta informação relativa aos pacientes (batimentos cardíacos por minuto e temperatura do paciente).

O cliente *desktop* foi desenvolvido usando o standard X3D para gráficos 3D e alguns métodos desenvolvidos em *JavaScript*. Na **figura 4** apresentamos os diferentes módulos desenvolvidos e as relações (de inclusão) entre estes.

De referir que apesar de se pretender transformar o modelo num cliente web, atualmente, para corretamente visualizar o interface deverá ser utilizado o software Instante Player ®.

Módulo project: Trata-se do módulo principal. Este módulo apresenta a estrutura final do modelo 3D (figura 5). É neste módulo onde se encontra a principal lógica que permite a interação com o modelo enquanto um interface.

A lógica necessária implementada foi desenvolvida diretamente no X3D usando, essencialmente, as entidades *BooleanFilter* e *TimeTrigger* que são alteradas através da interação do utilizador por *TouchSensors.* Os *TouchSensors,* por sua vez, acionam *TimeSensors* que obviamente alteram interpoladores conforme o exemplo apresentado de seguida que permite acender a luz associada a uma das cama:

```
<ROUTE fromNode='TOUCH_Bed1' fromField='touchTime' toNode='BOOL_BulblightFilterBed1'
toField='set_boolean'/>
<ROUTE fromField='touchTime' fromNode='TOUCH_Bed1' toField='set_startTime'
toNode='CLOCK_LightOn1'/>
<ROUTE fromNode='BOOL_BulblightFilterBed1' fromField='inputTrue' toNode='CLOCK_LightOn1'
toField='set_enabled'/>
<ROUTE fromNode='BOOL_BulblightFilterBed1' fromField='inputNegate' toNode='CLOCK_LightOn2'
toField='set_enabled'/>
<ROUTE fromNode='BOOL_BulblightFilterBed1' fromField='inputNegate' toNode='CLOCK_LightOn3'
toField='set_enabled'/>
<ROUTE fromField='touchTime' fromNode='TOUCH_Bed1' toField='set_startTime'
toNode='CLOCK_LightOff2'/>
 <ROUTE fromField='touchTime' fromNode='TOUCH_Bed1' toField='set_startTime'
toNode='CLOCK_LightOff3'/>
<ROUTE fromNode='BOOL_BulblightFilterBed1' fromField='inputTrue' toNode='CLOCK_LightOff2'
toField='set_enabled'/>
<ROUTE fromNode='BOOL_BulblightFilterBed1' fromField='inputTrue' toNode='CLOCK_LightOff3'
toField='set_enabled'/>
```

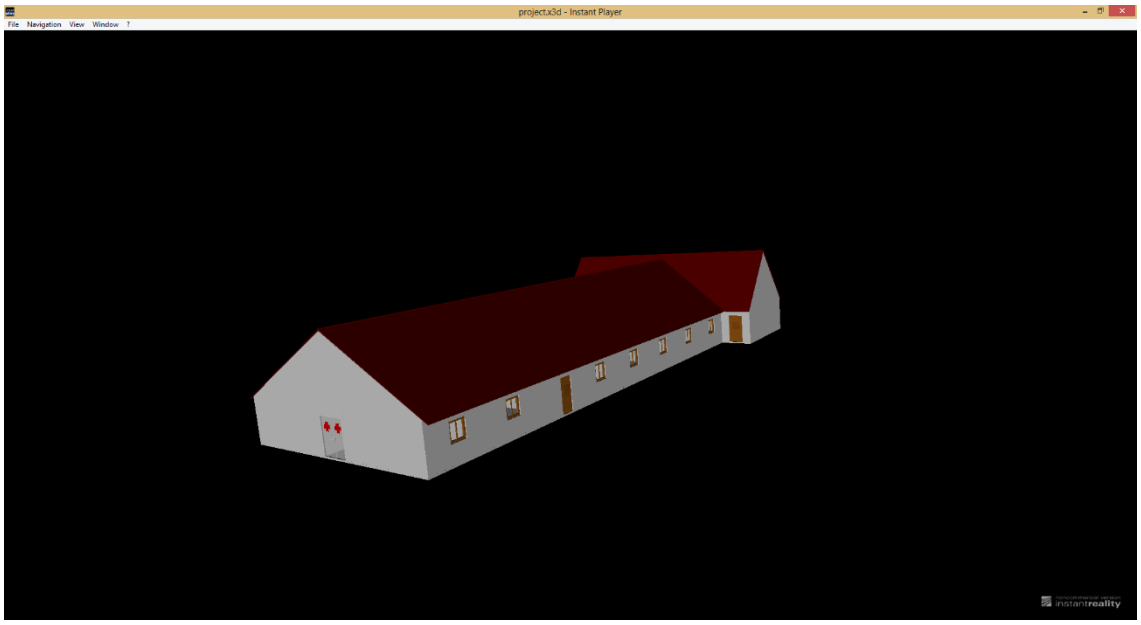

Figura 5: Exterior do serviço ambulatório

A execução da aplicação inicia-se clicando na porta de entrada. Uma animação abre a porta e desloca o utilizador para o interior do modelo do edifício. Colocando-o por fim no ponto de visualização a que chamamos “God Viewpoint” (figura 6).

Figura 6: Interior do serviço ambulatório ("God Viewpoint")

Para visualizar a informação de cada paciente seleciona-se uma cama (as camas encontram-se associadas aos pacientes através do modelo de dados) clicando sobre esta no modelo tridimensional. O utilizador será colocado imediatamente na perspetiva da cama.

Módulo json-decode: Neste módulo *JavaScript* encontram-se os métodos que permitem obter da base de dados a informação relacionada com cada paciente, *getHealthSync*. Estes métodos são usados no módulo *project:*

```
<Script DEF='BedN' url='"json-decode.js"'>
        <field name='bedNumber' type='SFString' accessType='initializeOnly' value='N'/>
        <field name='getHealthSync' type='SFString' accessType='inputOnly'/>
        <field name='bpm' type='SFString' accessType='outputOnly'/>
        <field name='temp' type='SFString' accessType='outputOnly'/>
         <field name='name' type='SFString' accessType='outputOnly'/>
        <field name='age' type='SFString' accessType='outputOnly'/>
        <field name='alert' type='SFBool' accessType='outputOnly'/>
</Script>
```

Cada execução escreve a informação sobre os utilizadores (nome, idade, batimentos por minuto e temperatura) no *dashboard* e caso os valores estejam fora de um determinado intervalo considerado normal é acionado um alerta (para temperatura foi considerado o intervalo entre 35 e 37 ºC e para os batimentos por segundo o intervalo entre 55 e 100 bpm). O método está a ser executado ciclicamente.

Módulo room:

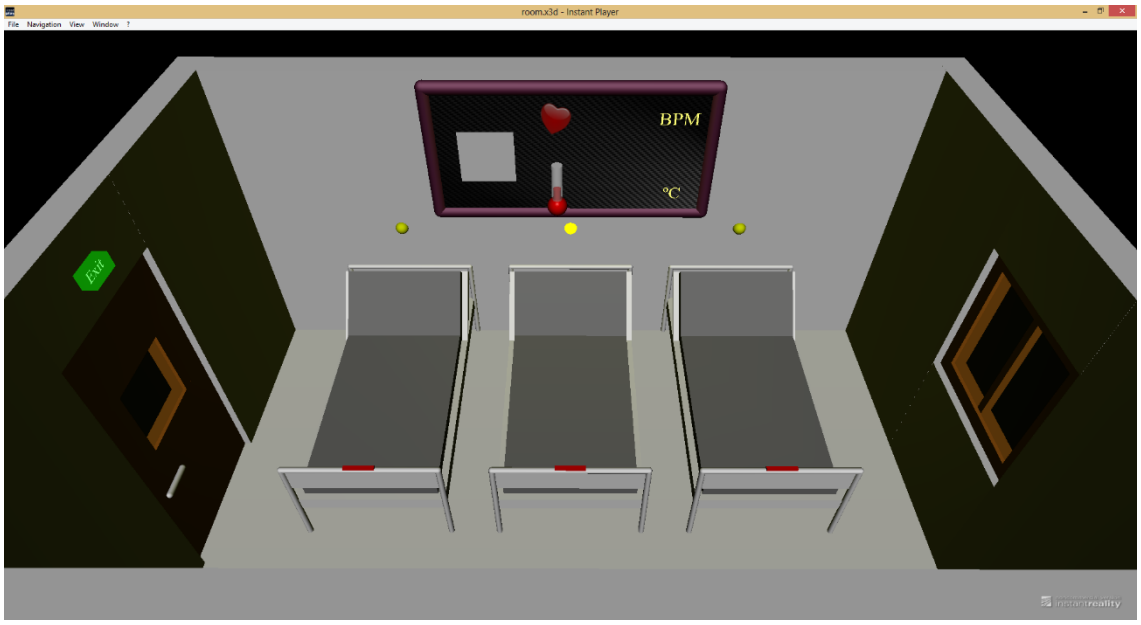

Figura 7: Modelo do quarto

Neste módulo apresentamos o modelo do quarto. Cada quarto apresenta três camas, o *dashboard* onde vai ser apresentada a informação relativa à cama/paciente selecionada/o, como se pode ver na figura 7.

Existem três *TouchSensors* que permitem ao utilizador selecionar (cama selecionada apresenta a lâmpada amarela ligada) e um quarto *TouchSensor* sobre a porta que recoloca o utilizador no ponto de vista geral *(god viewpoint*).

Caso o alarme relativo a uma cama no quarto esteja ativo a luz vermelha aos pés da cama encontrar-se-á a piscar (figura 8).

Este modelo é reproduzido 10 vezes no modelo final, ou seja, existem um total de 10 quartos, cada quarto com 10 camas o que perfaz um total de 30 camas.

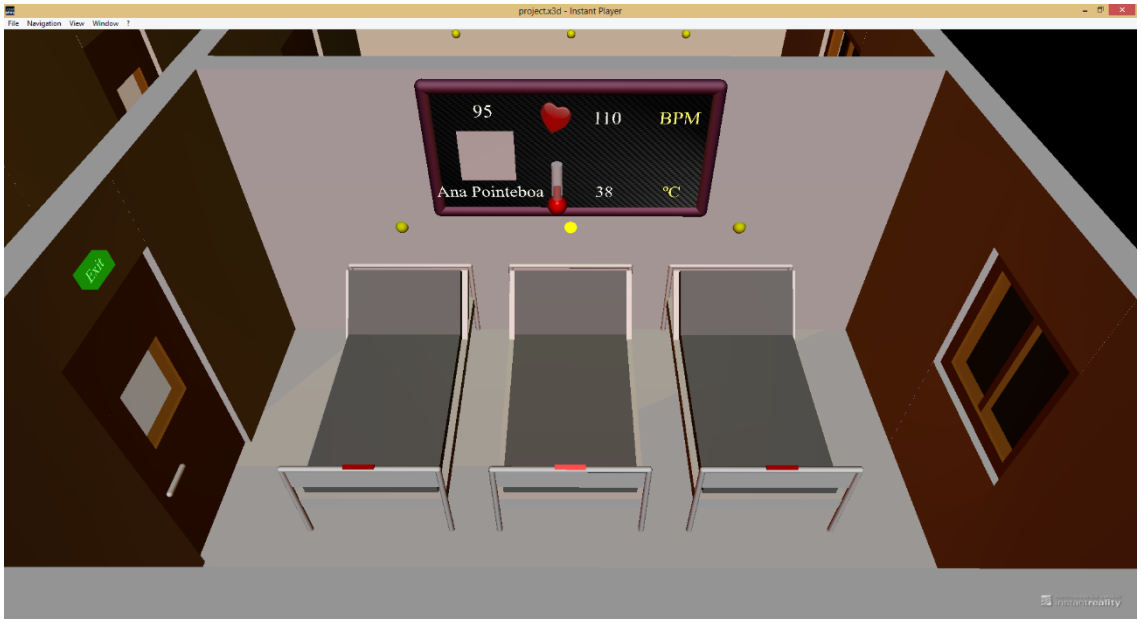

Figura 8: Modelo do quarto com alarme ativo.

Módulo dashboard:

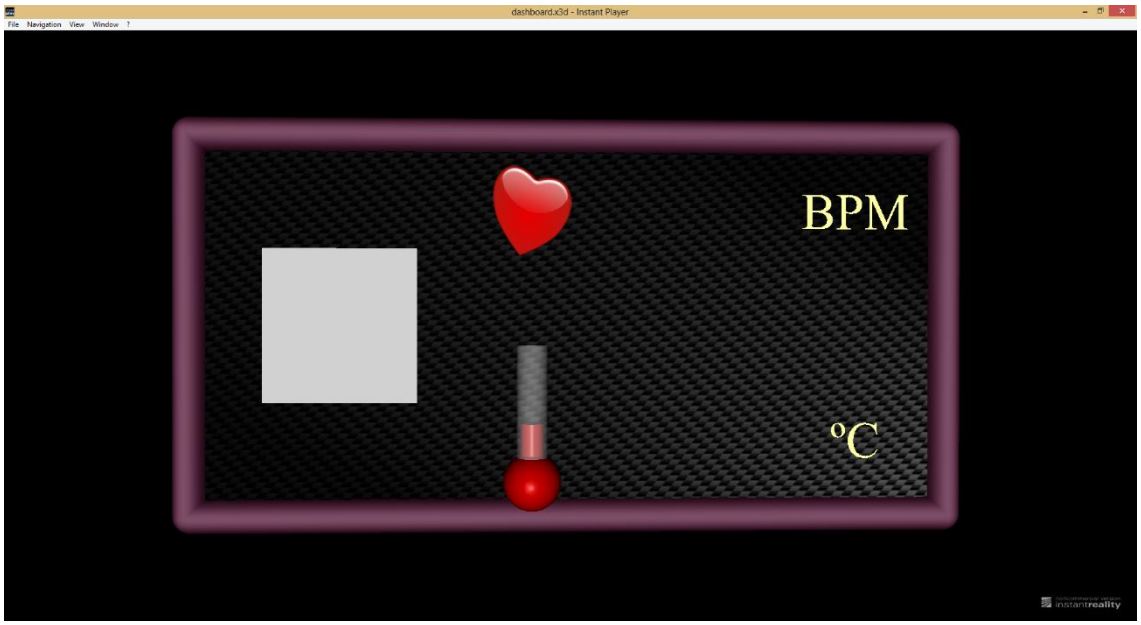

Figura 9: Design do dashboard.

Neste módulo encontra-se o modelo 3D do *dashboard* onde é visível a informação relativa a cada paciente. Existe um *dashboard* por quarto sendo apenas apresentada a informação relativa ao paciente associado à cama selecionada em cada toque no *TouchSensor* da cama.

Módulo main-panel:

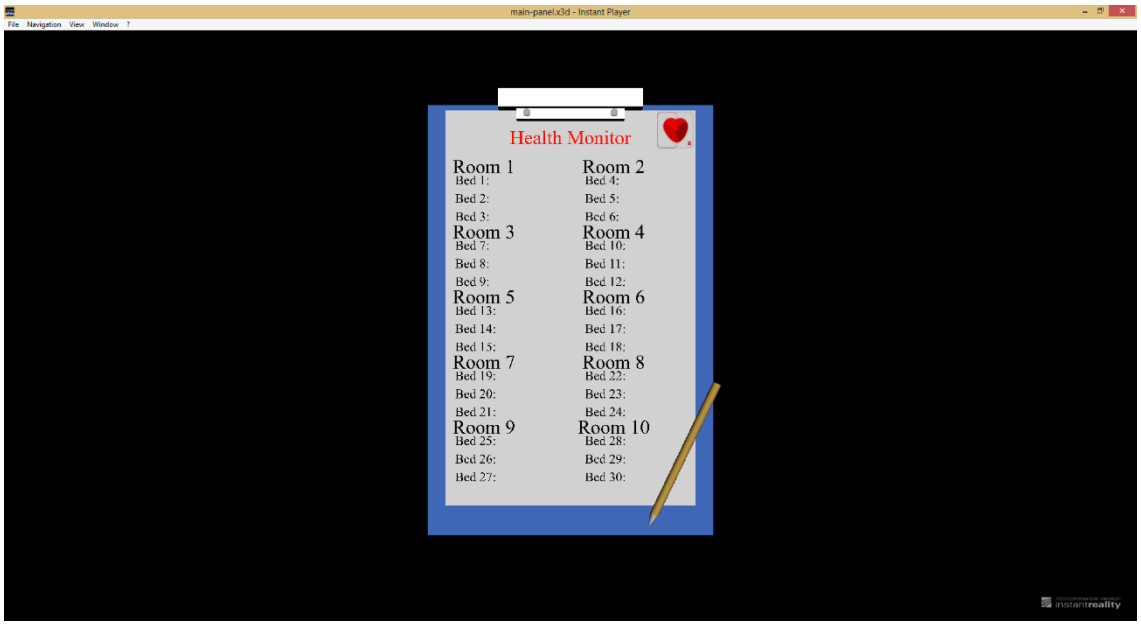

Figura 10: Painel principal de informação.

O painel principal, figura 10, apresenta-se como uma legenda para o modelo 3D do serviço ambulatório apresentando a informação dos números dos quartos e das camas bem como o nome dos pacientes em cada cama.

Sobrepondo o cursor sobre o painel principal a cama respetiva no modelo 3D será iluminada mapeando visualmente o modelo tridimensional com modelo de dados conforme se pode ver na figura 11.

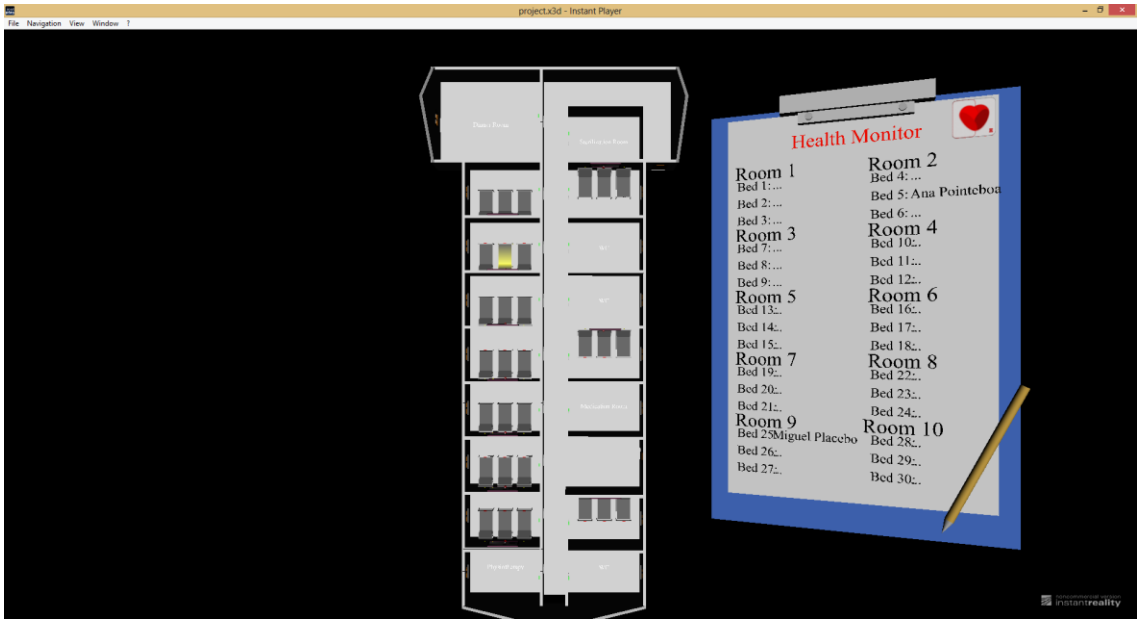

Figura 11: Cama iluminada através do painel principal.

Em caso de alarme a referência à cama no painel principal fica sublinhada informando o utilizador que há um alarme ativo (figura 12).

Figura 12: Painel principal com dois alarmes ativos

Conclusão

Apesar dos objetivos propostos terem sido alcançados e tendo em conta o facto que algumas opções no desenvolvimento tiveram em conta o pouco tempo disponível, a necessidade de apresentar um projeto final funcional e as limitações a que nos deparamos aquando a implementação e integração dos três projetos parece-nos importante referir que o "objeto" final pode ser amplamente melhorado em dois pontos essenciais:

Parece-nos que, numa perspetiva de continuação, seria importante melhorar e otimizar a comunicação com o servidor (leitura e escrita dos dados) tanto na eficiência (rapidez na execução dos pedidos, utilização pedidos assíncronos) como na segurança.

Também é importante referir que alguma da lógica do interface X3D foi implementada diretamente utilizando o X3D e poderá ser melhorada se implementada em *JavaScript* (ou mesmo Java) por serem linguagens mais eficientes a este nível e separando assim, inteiramente, a lógica do interface da implementação do modelo 3D.