**ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES**
**EACH- USP**

PROGRAMA ENSINAR COM PESQUISA 2009

# AVALIAÇÃO AUTOMATIZADA DE FERRAMENTAS DE REALIDADE VIRTUAL PARA TREINAMENTO MÉDICO

Relatório Final

**Lucas Prieto Nogueira**

Orientadora: Fátima de Lourdes dos Santos Nunes Marques

# Sumário

# Lista de Ilustrações

**Resumo**

A Realidade Virtual vem sendo muito utilizada para aplicações na área médica, com o intuito de ajudar os estudantes de medicina e profissionais de saúde no treinamento de procedimentos médicos antes de os mesmos serem executados em pacientes reais. A avaliação do aprendizado do usuário é fundamental em sistemas de Realidade Virtual para treinamento médico porque é necessário verificar o desempenho do usuário a partir dos parâmetros fornecidos por docentes da área. Para definir a avaliação é necessário conhecer as finalidades do treinamento e as formas de mensuração do desempenho. Não obstante, é imperativo permitir flexibilidade à avaliação, de forma que o docente possa configurar o sistema de acordo com suas necessidades. Este projeto visa a implementação de um sistema automatizado de avaliação de usuários de ferramentas de treinamento médico que utilizam a Realidade Virtual. Tais ferramentas são geradas por um *framework* denominado *ViMeT* que está em desenvolvimento usando a tecnologia Java. Para realizar os objetivos propostos foram definidas as seguintes etapas: estudo do *framework ViMeT*, revisão bibliográfica; estudo de sistema gerenciador de banco de dados; definição do modelo lógico do sistema; implementação do modelo de dados; elaboração do relatório parcial; implementação da avaliação; execução de testes e elaboração do relatório final e material didático. Nesse relatório estão contempladas as etapas referentes à implementação da avaliação. No relatório parcial foram abordadas as etapas referentes à revisão bibliográfica e definição do modelo lógico inicial do sistema de avaliação

## 1 Introdução

É notável a utilização de aplicações projetadas com base na Realidade Virtual (RV) para diversas funções. Seja para o aprendizado, treinamento ou entretenimento, os benefícios que essa tecnologia tem a oferecer são inúmeros.

Além de chamar a atenção por si só e despertar a curiosidade dos usuários, a RV, mesmo podendo ser uma tecnologia com o custo elevado por conta dos diversos equipamentos que podem ser utilizados, tem como vantagem o fácil aprendizado para seu manuseio e, acima de tudo, em vários casos o preço pago por ela pode ser mínimo se comparado aos ganhos proporcionados.

Nos sistemas de treinamento médico, que têm como objetivo treinar profissionais da área para executarem procedimentos de forma primorosa e minimizar os erros cometidos, o sucesso pode implicar até no salvamento de vidas, o que propicia um alto retorno social.

No entanto, não há sistema de treinamento que seja totalmente eficaz se não for possível medir o nível de aprendizado que o usuário obteve. Desde os primeiros anos escolares somos submetidos a provas constantes para medir nosso aprendizado, e como

não poderia ser diferente, softwares desenvolvidos para tais fins também têm que apresentar tal funcionalidade.

O framework *ViMeT* é um conjunto de classes que permitem implementar funcionalidades para simular exames de biópsia usando conceitos de RV e encontra-se em fase de desenvolvimento e utiliza a tecnologia Java (OLIVEIRA *et al.*, 2006 e NUNES *et al.*, 2007). Porém, ainda não inclui um sistema de avaliação de usuário compatível com as necessidades. É necessário criar um sistema de avaliação adequado com as ideias e funcionamento do *ViMeT*, para que o mesmo possa assim avaliar com precisão o desempenho dos usuários, considerando parâmetros de aprendizado que podem ser definidos por docentes.

## 1.1 Objetivos

Baseando-se no contexto apresentado, o objetivo deste trabalho é o desenvolvimento de um módulo de avaliação do usuário para o *framework ViMeT*. O objetivo geral é composto pelos seguintes objetivos específicos:

- Identificar na literatura métodos de avaliação aplicados a ferramentas de RV destinadas a simulações de treinamento;
- Definir formas de rastrear movimentos do usuário durante o treinamento;
- Definir parâmetros para mensurar o desempenho do usuário a partir de orientações de docentes da área médica;
- Implementar banco de dados para armazenar adequadamente os dados obtidos a partir da interação do usuário com a ferramenta, considerando a necessidade de flexibilidade aos docentes da área de aplicação.

## 1.2 Justificativa

Um módulo muito importante em aplicações RV para treinamento médico é a avaliação do aprendizado do usuário, devido à necessidade e importância de verificar o desempenho do mesmo a partir dos parâmetros fornecidos por docentes da área. Porém é necessário ter conhecimento das finalidades do treinamento e as formas de mensuração do desempenho. Além disso, é extremamente desejável permitir flexibilidade à avaliação, de forma que o docente possa configurar o sistema de acordo com suas necessidades.

Tendo essas considerações como base, este projeto visa a contribuir com o ensino tanto na área médica, que é o campo de aplicação, quanto na área de Sistemas de

Informação, visto que, tanto as técnicas, quanto os conceitos utilizados e desenvolvidos, além dos resultados obtidos, podem ser utilizados para o ensino de conceitos na área de programação de computadores, interação humano-computador, banco de dados e RV, entre outras. Também colabora com o projeto pedagógico da EACH, fornecendo uma abordagem multidisciplinar e procurando correspondência entre as necessidades de ensino de diversos cursos.

## 1.3 Organização do Trabalho

Este relatório apresenta, além desta introdução, duas seções, a saber:

Seção 2: Metodologia – explica as diversas tecnologias utilizadas, além de conter a implementação propriamente dita. Nesta seção também se encontra o cronograma do projeto.

Seção 3: Conclusões – aborda as conclusões acerca do trabalho realizado, identificando as disciplinas que interagem com o mesmo, além de apontar um caminho para a continuação do projeto.

# 2 Metodologia

Esta seção aborda as tecnologias utilizadas no sistema, alterações referentes ao projeto anterior e implementações de código.

## 2.1 Tecnologias Utilizadas

Tanto o *ViMeT* quanto o banco de dados utilizado no projeto possuem a mesma linguagem de programação em comum: Java. A plataforma Java tem como vantagem principal, além de ser gratuita, o fato de ser uma linguagem portável. Com essa funcionalidade é possível escrever programas em uma plataforma e executá-lo em outra plataforma qualquer. É uma linguagem testada, refinada e aprovada por uma comunidade que a torna a linguagem de programação mais ativa do planeta (Sun, 2009).

O banco de dados escolhido para o projeto é o Derby, que também utiliza a linguagem de programação Java. O Derby é um banco de dados relacional de código livre, ou seja, gratuito. Possui suporte pela Fundação Apache, sua desenvolvedora, além de só precisar da JVM para rodar, além é claro de estar instalado (Apache, 2009).

## 2.2 Banco de Dados

O banco de dados, responsável pelo armazenamento das informações referentes aos treinamentos, sofreu diversas mudanças para se readaptar ao projeto. A Figura 1 representa como era o modelo de dados antigo. A Figura 2 mostra como se tornou o mesmo, com as devidas explicações em seguida.

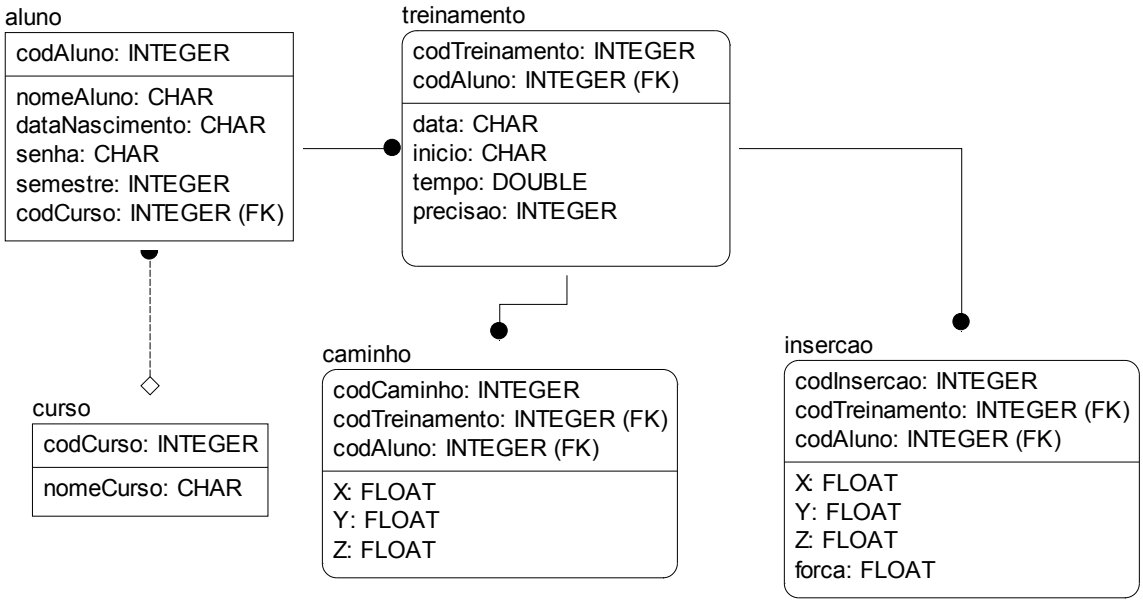


*Figura 1 – Antigo modelo de Dados*

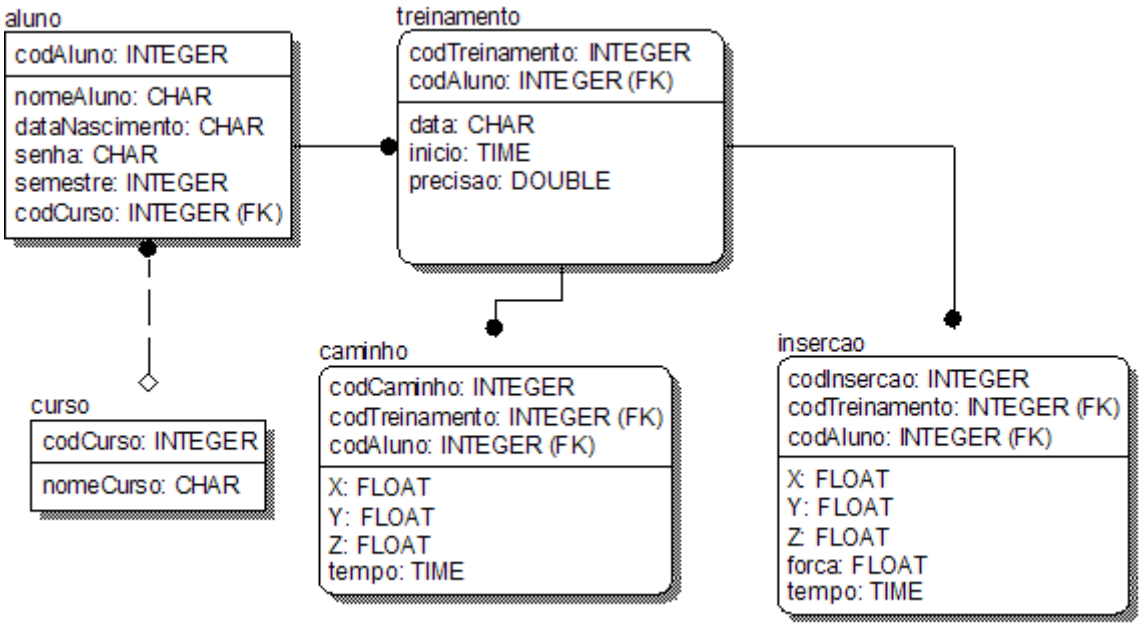

*Figura 2 – Modelo de dados Final*

Como principal mudança no modelo de dados, o campo *tempo* foi removido da tabela *treinamento* para uma melhor representação do mesmo. Com a mudança, o mesmo campo agora possui duas funções: na tabela *caminho* fica responsável por armazenar o momento exato em que as posições foram salvas no banco de dados, enquanto que na tabela *insercao*, a função do mesmo campo é armazenar o tempo exato em que ocorre a inserção no treinamento. Referente ao mesmo campo ainda podemos observar que agora seu tipo é *TIME*, mudança essa visando um armazenamento melhor dos dados.

Mudanças referentes aos tipos dos campos ocorreram outras vezes, como em *inicio* e *precisao*, visando novamente um melhor armazenamento dos dados no banco de dados.

Uma classe para o controle e manutenção do banco de dados também foi criada. A classe *BDAvaliacao* possui métodos para a abertura da conexão entre o banco de dados e o *ViMeT*, além dos respectivos métodos para adicionar parâmetros e checagem dos mesmos. A Figura 3 mostra como é feita a abertura da conexão entre ambos, através do método *AbrirBanco2()*.

```
public class BDAvaliacao
{
   Connection conn = null;
   Statement stm;

   public String framework = "incorporado";
   public String driver = "org.apache.derby.jdbc.EmbeddedDriver";
   public String protocol = "jdbc:derby:";
   private int cod, tipo;
   private String descricao, imagem;

   public static void main(String[] args)
   {
   }

      public void AbrirBanco2()
   {
      try
      {
            System.out.println("Iniciando AplicacaoBancoFramework em modo " + framework + ".\n");

            Class.forName(driver).newInstance(); //carrega o driver
            System.out.println("Carregando o Driver apropriado.\n");
            conn = DriverManager.getConnection("jdbc:derby:BD-FINAL");
            System.out.println("Conectado ao BancoFramework.\n");
            stm = conn.createStatement(); //inicializa o statement com a conexao
            conn.setAutoCommit(true); //commit da conexao true
   }

   catch (Exception e)
   {
         System.out.println("Não foi possível Conectar ao Banco " + e.getMessage() + "\n");
      }
   } //fecha Conexao
```

*Figura 3 – Classe BDAvaliacao*

## 2.3 Obtenção do Tempo

Para se obter o tempo foi necessário um estudo da melhor forma possível de se obter o mesmo e de como fazer tal ação. Foi utilizado o método *System.currentTimeMillis()*, por ser um método nativo e não requerer futuras implementações.

Este método retorna o tempo atual do sistema em milissegundos. Com isso, basta fazer os cálculos necessários para se obter com precisão os minutos e os segundos em que ocorreram a inserção. A Figura 4 mostra como ocorre os fatos descritos.

```
timeB = System.currentTimeMillis();
timeF = timeB - timeA;
timeS = (int) ((timeF / 1000) % 60);
timeM = (int) ((timeF / 1000) / 60);
System.out.println("Minutos: "+timeM+", Segundos: "+timeS);
```

*Figura 4 – Obtenção do Tempo*

A variável *timeB* armazena o tempo em que ocorre a inserção. A variável *timeA* é responsável por armazenar o tempo em que o sistema é inicializado. O resultado da subtração entre o tempo da inserção e o tempo inicial fica armazenado na variável *timeF*. Com isso, as variáveis *timeS* e *timeM* ficam responsáveis por armazenar os segundos e os minutos respectivamente, após os cálculos necessários para obter os mesmos.

## 2.4 Obtenção das Coordenadas

Para se obter as coordenadas foi utilizado um método pertencente à classe *MouseInfo*, o *getPointerInfo().getLocation()*. A coordenada *Z* se tornou uma constante com valor 1 para aplicações com o mouse. A Figura 5 mostra como foram implementadas tais funções.

```
System.out.println("MOUSE Coordenadas: ("+MouseInfo.getPointerInfo().getLocation().x+","
     ""+MouseInfo.getPointerInfo().getLocation().y+")");
```

*Figura 5 – Obtenção das Coordenadas*

De acordo com a figura, *MouseInfo.getPointerInfo().getLocation().x* corresponde ao método para se obter a coordenada *X*, enquanto *MouseInfo.getPointerInfo().getLocation().y* corresponde ao método para se obter a coordenada *Y*.

## 2.5 Adição de Parâmetros no Banco de Dados

Tendo as coordenadas e o tempo armazenados em variáveis, é necessário adicionar ambos ao banco de dados. As Figuras 6 e 7 mostram como essa ação ocorre.

```
    public boolean inserirCoordenada(int n, float xx, float yy, char min, char sec)
    {
      try {
      String insertSQL = "INSERT INTO insercao " + " VALUES (01, 01, n, xx, yy, 10, 20, '00:min:sec')";

              stm.executeUpdate(insertSQL);
              conn.commit();
              System.out.println("Insercao adicionada com sucesso!");
return true;
      }
      catch(SQLException e){
          try{
              if (e.getErrorCode() == -193){
                  System.out.println("Erro de integridade!" + e.getErrorCode());
                  return false;
              }
          }
          catch(Exception ex){
              System.out.println("Problema com tratamento de erros!");
              return false;
          }
      }
      return true;
  }
```

*Figura 6 – Método inserirCoordenada*

```
              bdl.inserirCoordenada(nInsercao, MouseInfo.getPointerInfo().getLocation().x,
MouseInfo.getPointerInfo().getLocation().y, (char)timeM,(char)timeS);
              nInsercao++;
```

*Figura 7 – Execução método inserirCoordenada*

A figura 6 representa o método na classe *BDAvaliacao* responsável pelo armazenamento dos parâmetros. As variáveis *xx* e *yy* estão relacionadas com as coordenadas, *min* e *sec* ao tempo, e *n* diz respeito ao número da inserção que está sendo armazenada. Na figura 7, a variável responsável por *n* é *nInsercao*. A cada inserção ocorrida durante o treinamento a variável é incrementada, logo o banco de dados guarda um registro para cada inserção ocorrida ao longo do treinamento.

## 2.6 Cronograma

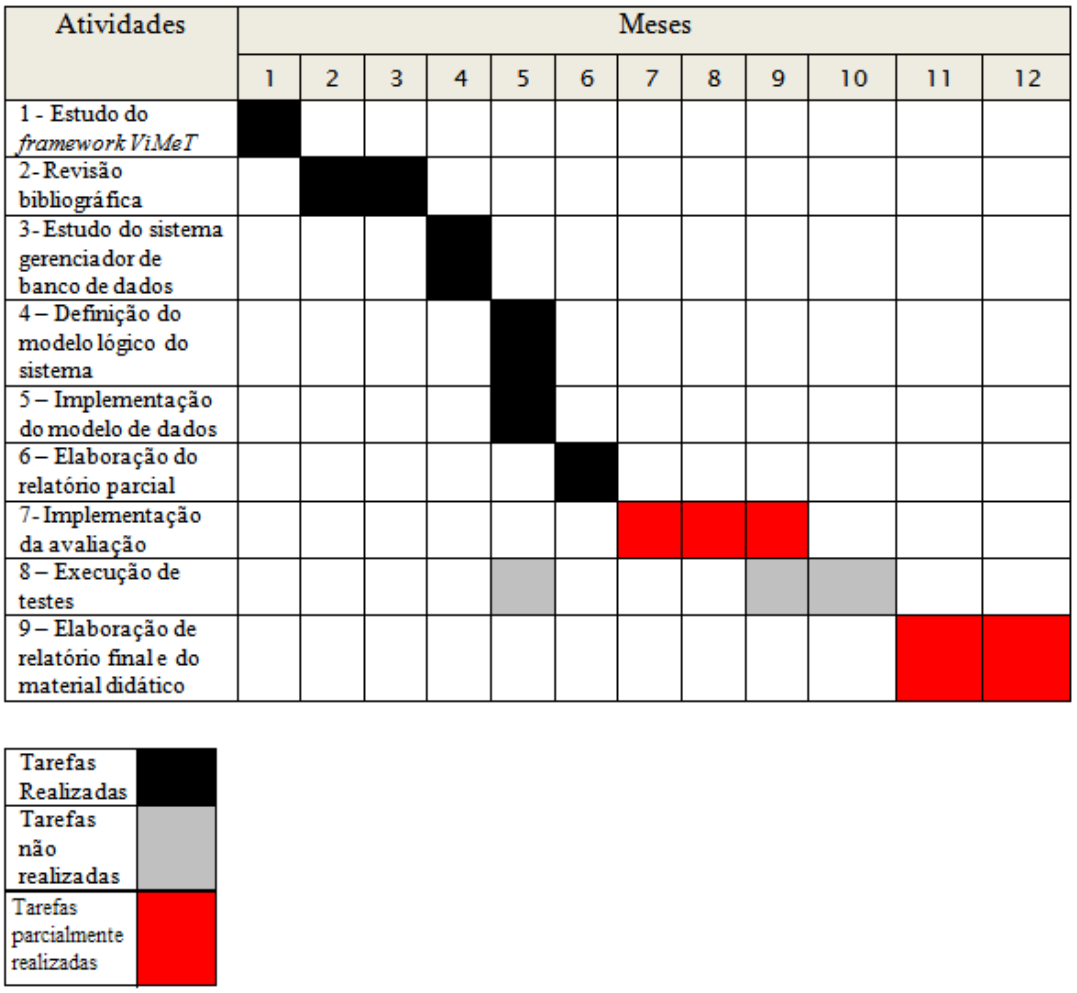


*Figura 8 – Cronograma*

Por motivos técnicos não foi possível implementar totalmente o sistema de avaliação, o que impossibilitou a execução de testes e a elaboração do relatório final e material didático.

Falhas no banco de dados impossibilitaram o perfeito armazenamento das informações, o que impediu a continuação das tarefas programadas. Primeiramente, com testes utilizando o banco de dados antigo ainda, um problema ligado as variáveis impedia que todo o conteúdo captado pelo sistema de treinamento fosse armazenado com perfeição na base de dados, gerando alguns erros e funcionando incorretamente. Com a troca no banco de dados, foi necessário refazer a classe, acrescentando e refazendo métodos. Desta vez o erro foi diferente, onde nada era armazenado na base de dados, tornando impossível gerar um resultado da avaliação.

Ocorreram problemas nas tentativas de captação das coordenadas também, onde houveram vários métodos sendo testados que se mostraram ineficazes e desnecessários ao longo do projeto.

Com esses problemas, sendo o de banco de dados o mais crucial, pois impossibilitava o andamento de todo o projeto, ficou impossível de implementar as idéias referentes ao cálculo da nota final do usuário, que compreendia uma série de análises e cálculos com base nas informações armazenadas na base de dados.

## 3 Conclusões

As maiores dificuldades relacionadas ao projeto foram em como obter corretamente os parâmetros necessários para viabilizar a avaliação com perfeição, além dos problemas relacionados ao banco de dados, seja pela necessidade de alteração do mesmo ou pelos problemas ocorridos ao longo do projeto.

O projeto acrescentou um valor importante na área de programação e de banco de dados, estando relacionado diretamente as disciplinas referentes à programação, principalmente as que utilizam a linguagem de programação Java, e também com as disciplinas que têm como abordagem Banco de Dados.

A continuação do projeto poderia ser feita através do reparo dos erros ocorridos e do término da implementação da avaliação. Para isso seria necessário corrigir os problemas técnicos relacionados ao banco de dados, programar o sistema de avaliação e ligar o mesmo com o sistema de banco de dados, para que seja possível obter os dados armazenados e calcular a nota do treinamento.

# REFERÊNCIAS

APACHE. Disponível em: <http://db.apache.org/>. Acesso em: 10 ago. 2009.

NUNES, F. L. S.;OLIVEIRA, A. C. M. T. G.;ROSSATO, D. J.;MACHADO, M. I. C. **ViMeTWizard: Uma ferramenta para instanciação de um framework de Realidade Virtual para treinamento médico** In: CONF. LATINOAMERICANA DE INFORMÁTICA, 33. **Proceedings...** 2007. p. 1–8.

OLIVEIRA, A. C. M. T. G.;PAVARINI, L.;NUNES, F. L. S.;BOTEGA, L. C.;ROSSATO, D. J.;BEZERRA, A. **Virtual Reality Framework for Medical Training: implementation of a deformation class using Java**. In: ACM INTERNATIONAL CONFERENCE ON VIRTUAL REALITY CONTINUUM AND ITS APPLICATIONS, 2006, Hong Kong. **Proceedings…** 2006. v. 1.

SUN. Disponível em: <http://www.java.com/en/about/>. Acesso em: 22 ago. 2009.