UNIVERSIDADE FEDERAL TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JULIO CEZAR RIFFEL

SISTEMA WEB PARA REALIZAÇÃO DE TESTES DE MÚLTIPLA ESCOLHA

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2011

JULIO CEZAR RIFFEL

SISTEMA WEB PARA REALIZAÇÃO DE TESTES DE MÚLTIPLA ESCOLHA

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2011

ATA Nº: **186**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **JULIO CEZAR RIFFEL**.

Às 10:05 hrs do dia 7 de julho de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Soelaine Rodrigues Ascari (Convidada) e Cristiane T. Tartare de Carvalho (Convidada), para avaliar o Trabalho de Diplomação do aluno Julio Cezar Riffel, matrícula 980439, sob o título **Sistema Web para Realização de Testes de Múltipla Escolha**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Coordenação de Informática. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 11:00 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

Profa. Cristiane T. Tartare de Carvalho, Esp.
Convidada

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

RIFFEL. Julio Cezar. Sistema *web* para realização de testes de múltipla escolha. 2011. 53 f. Trabalho de conclusão de curso (graduação de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2011.

Aplicações para ambiente Internet têm sido uma alternativa para os sistemas que são baseados na arquitetura cliente/servidor e que não podem ser facilmente disponibilizados e/ou acessado por meio em uma rede proprietária. Com o uso da Internet um computador com acesso aos seus serviços e um navegador *web* instalado é, geralmente, suficiente para acessar uma aplicação *web*. Assim, aplicações baseadas na ideia de cliente/servidor podem ser utilizadas com mais facilidade. Considerando essa facilidade de acesso e a necessidade de duas alunas do curso de Química da Universidade Tecnológica Federal do Paraná, campus Pato Branco, de realizar testes com alunos, verificou-se a possibilidade de desenvolver um sistema *web* para o cadastro de questões de múltipla escolha. Além disso, o sistema também permite a composição de testes, a resolução desses testes por alunos e a correção automática dos mesmos. Optou-se pela linguagem Flex para implementação do sistema pelos recursos que a mesma possui para o desenvolvimento de aplicações Internet ricas.

Palavras-chave: Sistemas *web*. Flex e PHP. Rich Internet Application. Aplicações Internet.

Otimista é aquele que se vê obrigado a subir numa árvore para fugir de um leão e ainda aprecia a paisagem (Walter Winchell).

AGRADECIMENTOS

Não posso deixar de agradecer aos meus pais e familiares, sem os quais não estaria aqui, e por terem me fornecido condições para me tornar a pessoa que sou.

Agradeço em especial a admirável professora Beatriz, pelos inúmeros puxões de orelha, ensinamento e dedicação dispensados, sem a qual está monografia não teria a mesma qualidade.

A todos os professores pelo conhecimento repassado, dedicação e ensinamentos disponibilizados nas aulas, cada um de forma especial contribuiu para a conclusão desse trabalho e consequentemente para minha formação profissional.

Por fim, gostaria de agradecer aos meus amigos e familiares, pelo carinho, apoio e compreensão, meu eterno agradecimento.

LISTA DE FIGURAS E QUADROS

LISTAGENS DE CÓDIGO

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
AIR	*Adobe Integrated Runtime*
AMF	*Action Message Format*
API	*Application Programming Interface*
CGI	*Common Gateway Interface*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IBM	*International Business Machines*
IECA	Inclusão, Exclusão, Consulta e Alteração
IMAP	*Internet Message Application Protocol*
JVM	*Java Virtual Machine*
MXML	*Maximum Experience Markup Language*
PDF	*Portable Document Format*
POP	*Post Office Protocol*
RIA	*Rich Internet Applications*
RPC	*Remote Procedure Call*
SDK	*Standard Development Kit*
SGBD	Gerenciamento de Banco de Dados
SMTP	*Simple Mail Transfer Protocol*
SQL	*Structured Query Language*
SWF	*Small Web Format*
UML	*Unified Modeling Language*
URL	*Universal Resource Locator*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações inicias, com uma visão geral do trabalho, os objetivos e a justificativa do mesmo e a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

O computador pode ser utilizado de diversas maneiras como auxiliar no processo de ensino e aprendizagem, seja disponibilizando informações por meio da Internet, pelo uso de objetos de aprendizagem que simulam fenômenos físicos ou pelo auxílio na elaboração e na realização de testes.

Um sistema computacional que permite a elaboração de testes como uma lista de questões é interessante porque o professor pode cadastrar esses testes e os alunos respondê-los no próprio computador. As questões que compõem um teste podem ser escolhidas pelo professor ou por um processo randômico realizado pelo computador.

Os resultados dos testes realizados pelos alunos podem ficar armazenados em um banco de dados e assim o professor pode acompanhar a evolução de cada aluno e de grupos de alunos. Com os dados armazenados é possível, ainda, identificar conceitos ou conteúdos da disciplina nos quais os alunos apresentam mais dificuldade. Facilitando, desta forma, a aplicação de recuperação de conteúdos específicos.

Contudo, para que o próprio sistema computacional possa fazer a verificação das respostas do aluno há restrições no tipo de questão elaborada. Questões de múltipla escolha, de associar colunas e de completar frases com palavras, por exemplo, facilitam o processo de verificação. Já questões com respostas abertas, nas quais o raciocínio pode estar amplamente envolvido na resposta, faz com que o processo de correção automatizado seja muito difícil ou mesmo impossível.

Considerando esse contexto, um sistema foi desenvolvido para a composição e aplicação de testes de múltipla escolha. Como forma de facilitar o acesso ao sistema, sem necessidade de instalação em cada máquina cliente ou que fosse necessária uma rede proprietária, optou-se por desenvolver um sistema para *web*. Por meio das tecnologias Flex e a linguagem PHP foi possível implementar esse sistema com conceitos de interface rica para Internet. Facilitando, a interação dos alunos e dos professores. A facilidade de interação é colocada porque foram utilizados recursos de interface semelhantes aos existentes em

aplicações *desktop.*

1.2 OBJETIVOS

O objetivo geral apresenta o resultado principal do trabalho realizado e os objetivos específicos o complementam, no sentido de valores agregados.

1.2.1 Objetivo Geral

- Implementar um banco de questões com respostas de múltipla escolha.

1.2.2 Objetivos Específicos

- Possibilitar a composição de listas de questões escolhidas por um usuário;
- Armazenar as questões por níveis de dificuldade, assunto e séries escolares;
- Permitir ao aluno responder uma lista de questões que serão corrigidas pelo sistema, as respostas armazenadas em banco de dados e o resultado disponibilizado para consulta para o usuário que o respondeu.

1.3 JUSTIFICATIVA

A implementação de um sistema para armazenamento de questões facilita o trabalho do professor e pode ser um fator motivador para o aluno. Isso porque o aluno pode responder questões em níveis de dificuldade crescente e assim avaliar o seu aprendizado e as dificuldades com determinados conteúdos.

Esse banco de questões será utilizado por duas alunas do curso de Bacharelado em Química da Universidade Tecnológica Federal do Paraná para um estudo sendo realizado por elas. Contudo, o aplicativo será implementado de maneira que possa ser utilizado em outras áreas. Serão apenas questões de múltipla escolha porque essa foi a solicitação das alunas. E, ainda, por ser mais simples realizar a correção automática das questões.

O sistema será desenvolvido em camadas com separação entre a camada de apresentação, lógica de negócio e banco de dados. O sistema possuirá uma versão *web*, desenvolvida pelo autor deste trabalho, e uma versão para *desktop*, desenvolvida pelo aluno Roberto Rosin. Esse desenvolvimento paralelo de um mesmo sistema possibilitará realizar comparações, do ponto de vista do programador. Isso ocorrerá se dados relevantes forem

identificados, de diferenças ou semelhanças, entre a implementação *web* e *desktop* de um mesmo sistema.

1.4 ORGANIZAÇÃO DO TEXTO

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a idéia e o contexto do sistema, incluindo os objetivos e a justificativa.

O Capítulo 2 contém o referencial teórico que fundamenta a proposta conceitual do sistema desenvolvido. O referencial teórico está centrado no desenvolvimento de aplicações para Internet, incluindo aplicações para Internet com interface rica.

No Capítulo 3 estão os materiais e o método empregados no desenvolvimento deste trabalho.

O Capítulo 4 contém o sistema desenvolvido, com exemplos de documentação da modelagem e de implementação. A modelagem é exemplificada por documentos de análise e projeto. A implementação é exemplificada pela apresentação do sistema com telas e descrição de suas funcionalidades e por exemplos da codificação do sistema. Esse capítulo também apresenta uma discussão, com a análise realizada da implementação do mesmo sistema em versão *web* e *desktop*.

No Capítulo 5 está a conclusão com as considerações finais.

2 APLICAÇÕES PARA AMBIENTE INTERNET

Este capítulo apresenta o referencial teórico utilizado para fundamentar o sistema proposto. O texto está focado em aplicações ricas para ambiente Internet. As RIAs (*Rich Internet Application)* seguem o paradigma cliente/servidor, mas diferentemente das aplicações *web* tradicionais, elas são capazes de transferir processamento da interface, da lógica de negócio e de dados para o cliente, utilizando comunicação assíncrona.

2.1 DAS APLICAÇÕES *WEB* TRADICIONAIS ÀS RIAS

De maneira informal, são denominadas arquiteturas *web* convencionais as que possuem a interface baseada em HTML (*HyperText Markup Language*). Nas aplicações baseadas nessa arquitetura a comunicação entre cliente e servidor é realizada pelo protocolo HTTP (*HyperText Transfer Protocol*) e a camada de aplicação e de dados é desenvolvida com base em CGIs (*Common Gateway Interface*) que estão vinculadas a bases de dados relacionais.

Fundamentado em um conceito mais estruturado, aplicações *web* se referem ao conjunto de programas que implementa um sistema de informação de acordo com o paradigma cliente/servidor suportado pelo protocolo de comunicação HTTP e que possuem a camada interativa (de apresentação) escrita em HTML (OLIVEIRA, PEREIRA, HENRIQUES, 2005). O uso de HTML faz com que a interface com o usuário seja suportada pelos aplicativos para navegação na rede de hiperdocumentos, a *web*.

As aplicações *web* podem variar desde páginas compostas unicamente de texto até sistemas complexos, como aplicações empresariais de grande porte (Internet ou intranet), que conectam cadeias de valor ou informatizam o chão de fábrica. Para o desenvolvimento de aplicações mais complexas, HTML e HTTP não têm mais sido suficientes pelos recursos e as novas necessidades de interação exigidas pelas aplicações e por seus usuários. Isso conduz a um repensar do próprio conceito de aplicações *web*.

Para este trabalho, aplicação *web* é definida como a aplicação que usa a Internet como rede de comunicação local ou pública e os seus serviços. De acordo com esse conceito é indiferente quais tecnologias são utilizadas para implementar a aplicação (interface, lógica de negócio ou banco de dados) e quais são utilizadas na execução, como, por exemplo, *plugins*

vinculados ao navegador na máquina cliente.

Como forma de compreender a transição do conceito de aplicações *web* baseadas em HTML e HTTP às RIA, as Figuras 1 a 4 ilustram essa evolução. A Figura 1 contém uma representação ilustrativa da arquitetura de aplicações *web* baseadas em HTML e HTTP. Essa pode ser considerada a primeira forma de arquitetura *web*, porque é definida a partir dos primeiros elementos de tecnologia utilizados para uma comunicação entre cliente e servidor utilizando ambiente Internet.

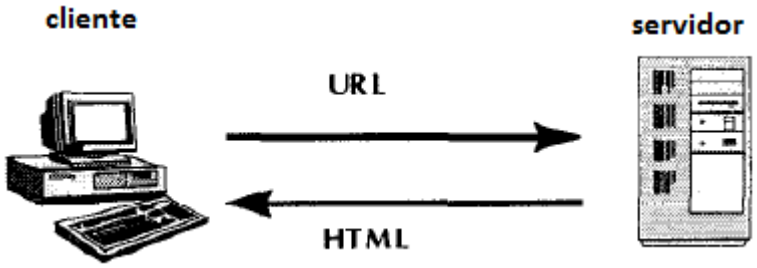


Figura 1 – Navegação em hipertexto usando HTML
Fonte: adaptado de CHO et al. (1997).

Nessa arquitetura, o cliente especifica a URL (*Uniform Resouce Locator*) de destino por meio do *browser web* e o servidor transmite os documentos HTML solicitados para o cliente. A URL contém o endereço *web* da página a ser acessada. Esse é o modelo para páginas *web* baseadas em texto em formato HTML e não há processamento ou funcionalidades realizadas pelo servidor, além de armazenar o texto que é formatado pelas *tags* HTML no momento da sua exibição. Os textos exibidos podem estar vinculados por ligações (*hiperlinks*).

Uma evolução da exibição de conteúdo *web* baseado em hipertexto usando HTML é apresentada na Figura 2. Essa arquitetura é caracterizada pelas funcionalidades providas pelos programas CGI. O CGI provê processamento vinculado à página HTML.

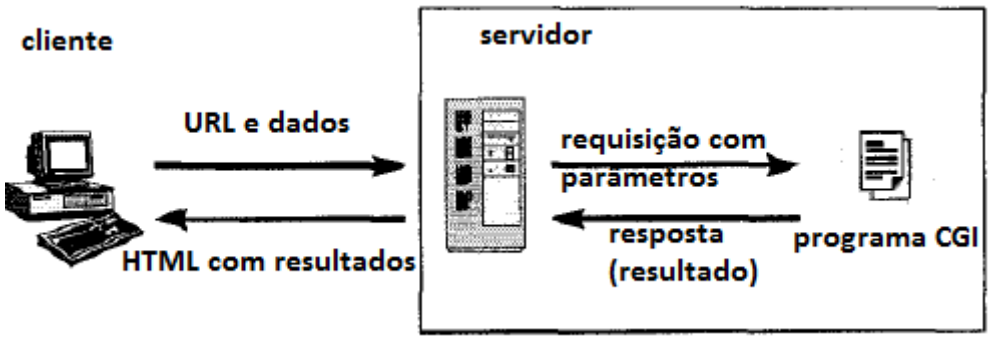


Figura 2 – Aplicação web usando CGI
Fonte: adaptado de CHO et al. (1997).

Na arquitetura representada na Figura 2, o cliente especifica, mesclado com *tags* HTML, o programa CGI e um conjunto de argumentos para esse programa. Essa especificação é transmitida ao servidor. O servidor decodifica os parâmetros e invoca o programa CGI com os argumentos recebidos. Ao finalizar a execução do programa CGI, o servidor transmite os dados para o cliente que são exibidos por meio de HTML.

Com o modelo representado na Figura 2, o cliente pode invocar programas residentes no servidor *web* e a aplicação *web* se torna mais funcional. A principal desvantagem dessa abordagem é o tempo que é necessário para invocar os programas CGI e decodificar/codificar os parâmetros, resultando em respostas demoradas para as requisições dos clientes (CHO et al., 1997).

A interação com bases de dados no servidor é uma evolução aos programas CGI. Contudo, o cliente ainda especifica o programa CGI com parâmetros que são transformados pelo programa em consultas SQL (*Structured Query Language*) para que o SGBD (Sistema de Gerenciamento de Banco de Dados) possa processar. Os resultados produzidos pelas operações com a base de dados são mesclados com *tags* HTML e transmitidos ao cliente. No cliente esses resultados são apresentados em documentos HTML. A representação da arquitetura de uma aplicação *web* com a inclusão de banco de dados no servidor é apresentada na Figura 3.

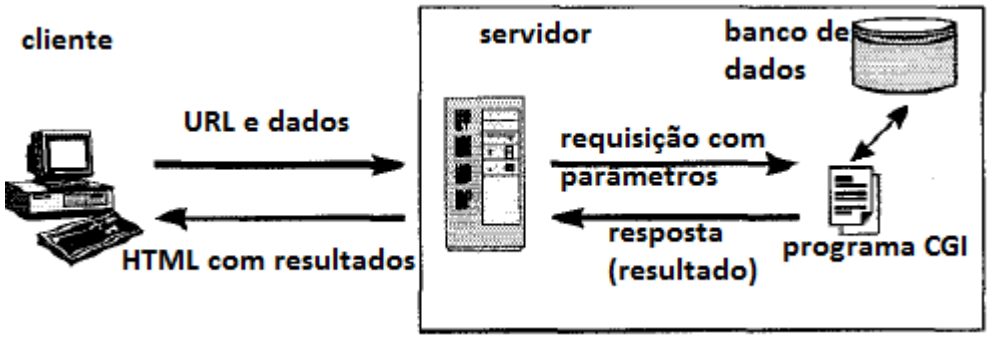


Figura 3 – Servidor *web* com interface para banco de dados
Fonte: adaptado de CHO et al. (1997).

Cho et al. (1997) ressalta que a arquitetura representada na Figura 3, apesar de apresentar uma evolução significativa ao CGI puro, ainda apresenta demora de resposta devido ao fraco acoplamento entre o servidor e os programas CGI.

Esse tipo de arquitetura baseada em CGI não se apresenta mais como suficiente para a maioria das aplicações atuais que executam no ambiente Internet, como comércio eletrônico e outras que manipulam bases de dados que podem armazenar a manipular grandes volumes de dados e de tipos complexos, como imagens e vídeos.

Além dessas características requeridas às aplicações *web*, os usuários que utilizam sistemas *desktop* com finalidades profissionais estão acostumados a recursos como menus dinâmicos, efeitos de arrastar-e-soltar, botões diferenciados e outros. Esses recursos podem ser providos pelas aplicações Internet Ricas. Duhl (2003) ressalta que as RIAs foram propostas com o objetivo de resolver problemas encontrados nas aplicações *web*. Elas provêm interface para representar processos e dados complexos, ao mesmo tempo em que minimizam a transferência de dados entre cliente e servidor (FUKUDA, YAMAMOTO, 2008).

Para Fraternalli e Paolini (1998), o projeto de uma aplicação *web* deve considerar particularidades de três dimensões:

a) Estrutural (conceitual) - define a organização das informações a ser tratada pela aplicação e os seus relacionamentos;

b) Navegacional - define como as informações serão acessadas através da aplicação;

c) Apresentação - define como as informações serão apresentadas e o acesso será realizado pelo usuário.

Cada uma dessas dimensões define visões diferentes do projeto, independentemente da arquitetura a ser adotada para o sistema.

Utilizando uma linguagem como Java ou PHP, por exemplo, a representação esquemática da arquitetura de uma aplicação *web* com banco de dados apresentada na Figura 3 é ampliada, possibilitando que tanto o cliente quanto o servidor possuam dados armazenados e realizem processamento. Essa nova arquitetura é representada na Figura 4.

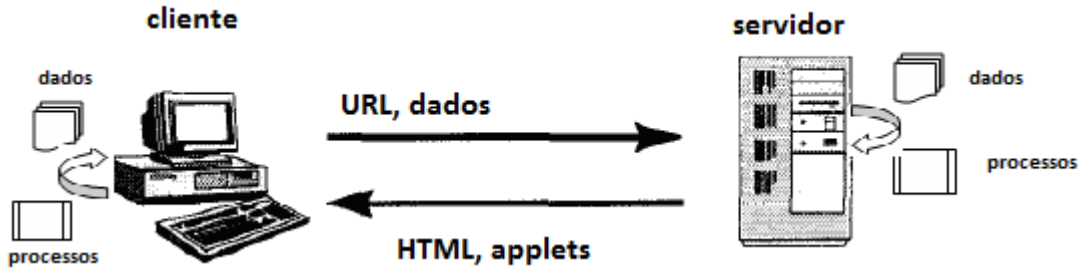


Figura 4 – Arquitetura web
Fonte: adaptado de CHO et al. (1997).

No modelo de arquitetura esquematicamente representado na Figura 4, tanto o cliente quanto o servidor podem armazenar dados e realizar processamento. Essa é a base para as Aplicações Internet Ricas apresentadas na próxima seção.

2.2 RICH INTERNET APPLICATIONS

O crescimento de aplicativos que utilizam serviços Internet tem feito com que os usuários demandem novas funcionalidades às mesmas. Essas funcionalidades também se referem a superar essas desvantagens apresentadas pelas aplicações *web* atuais. Dentre essas desvantagens estão a de prover integração efetiva de áudio e vídeo e de altos níveis de interatividade (PRECIADO et al., 2005). Essas novas necessidades decorrem, em partes, porque os usuários de aplicativos comerciais, e mesmo outros, estão acostumados com os recursos oferecidos pelas aplicações *desktop*.

As aplicações *web* que procuram prover essas funcionalidades são denominadas Rich Internet Applications. De forma simples, as RIAs são aplicações *web* que possuem características e funcionalidades das aplicações *desktop* tradicionais. Como muitos usuários estão acostumados com recursos das aplicações *desktop*, as RIAs fazem com que o seu uso seja facilitado e oferecem uma interface mais rica em termos de recursos e funcionalidades (LOOSLEY, 2006).

A possibilidade das RIAs superarem as restrições das aplicações *web* tradicionais decorre de um conjunto de capacidades para clientes ricos que incluem habilidades para:

a) Recursos avançados para interação com o usuário, incluindo características como janelas e controles de navegação de dados como botões, caixas de verificação, botões de rádio e componentes de mídia, incluindo filmes e sons.

b) Integração de origens de dados locais e remotas.

c) Interface (camada de apresentação) fracamente acoplada à lógica de negócio, facilitando a realização de manutenções e alterações na interface.

d) Prove eficiência em computação distribuída pela facilidade de comunicação pela possibilidade de processamento no cliente e modo de comunicação assíncrono.

e) Não necessita de instalação porque a aplicação é baixada automaticamente e executa o navegador *web*. Contudo, pode ser necessário instalar *plugins* como é o caso do Flash Player para aplicações Flex.

f) Redução do tráfego na rede porque a aplicação somente envia dados pertinentes ao servidor e recebe informações dos dados que mudam. A aplicação se comunica com o servidor em taxas menores do que quando páginas inteiras são transmitidas. Isso sugere que clientes que possuem conexões de baixa largura de banda (por exemplo, conexões *dial up*) possam executar aplicações Internet ricas (MULLET, 2003).

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados na realização deste trabalho. Os materiais se referem às tecnologias como linguagens e ferramentas para a modelagem e a implementação do sistema. O método contém as etapas com os principais procedimentos utilizados para o desenvolvimento do sistema.

3.1 MATERIAIS

As ferramentas e tecnologias utilizadas para modelagem e implementação do sistema foram:

a) Astah Community para a modelagem do sistema;

b) Adobe Flex SDK 4.0 (*Standard Development Kit)* vinculado ao Flash Builder para a implementação da interface;

c) Linguagem PHP para a implementação da camada de aplicação;

d) AMFPHP para interação entre os objetos Flex da interface e o código PHP.

e) MySQL para o banco de dados;

f) MySQL Workbench 5.2 CE como administrador do banco de dados;

g) Adobe Dreamweaver CS5 para criar a classe de conexão com o banco de dados;

h) Apache como servidor *web*.

A Figura 5 contém uma representação esquemática, elaborada pelo autor deste trabalho, da interação entre as tecnologias responsáveis pela execução do sistema. Com Flex, independente do *browser* ou sistema operacional a interpretação e a apresentação do aplicativo ocorre da mesma maneira. Evitando, assim, a necessidade de múltiplos testes em vários sistemas operacionais e navegadores. A Figura 5 ilustra o funcionamento de um aplicativo web utilizando as tecnologias Flex, AMFPHP e MySql.

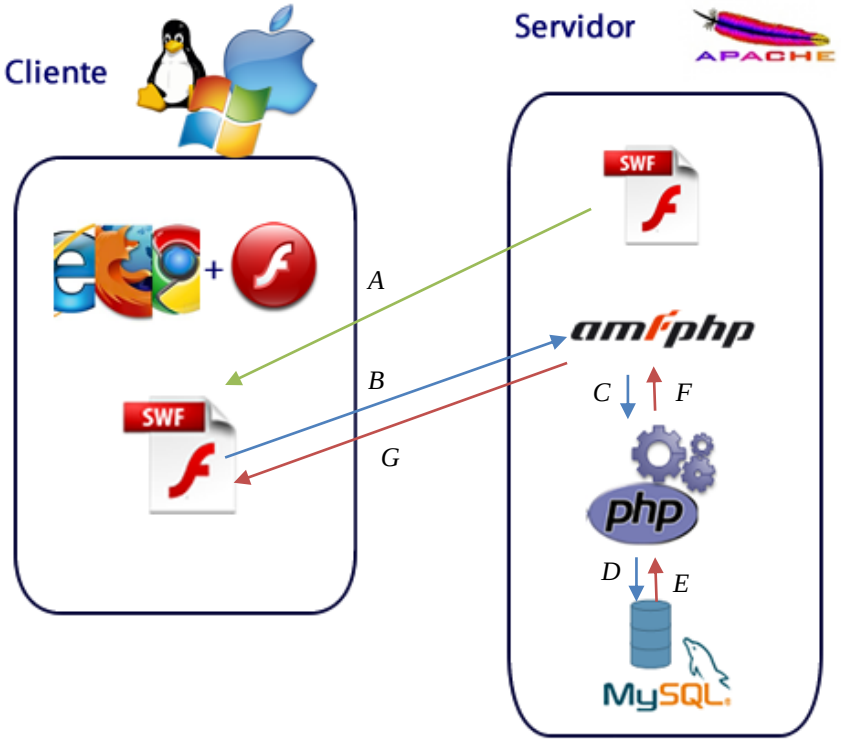


Figura 5 – Arquitetura de comunicação

A Figura 5 representa que vários navegadores podem ser utilizados e que a eles está acoplando a máquina virtual Flash Player. No primeiro acesso ao aplicativo que é realizado pelo computador cliente é feito o *download* do arquivo SWF (*Small Web Format)* do servidor para o navegador do cliente. E, também, é necessário que o *plugin* Flash Player esteja instalado na máquina do cliente.

A forma de comunicação entre o cliente e o servidor inclui:

a) Envio de requisição com os parâmetros via *remote object* para o AMFPHP (B na Figura 5);

b) O AMFPHP direciona a requisição e os parâmetros para a classe (C na Figura 5) localizada no *package services* com método desejado;

c) A classe PHP faz a regra de negócio processando o método solicitado e se tudo estiver correto, o processamento é continuado;

d) A classe PHP realiza a comunicação (*select*, *insert*, *update*, ...) com o banco de dados, No sistema implementado como resultado deste trabalho, foi utilizado o banco de dados MYSQL, mas pode ser utilizado qualquer banco de dados suportado pelo PHP;

e) O banco de dados retorna o resultado da sentença (*query*) para o método localizado na classe;

f) O método realiza as operações e retorna os dados para o AMFPHP;

g) O AMFPHP realiza a serialização binária dos dados e os transmite para a aplicação SWF que está executando no navegador *web* do cliente.

3.1.1 Astah Community

Astah Community (ASTAH, 2010) é uma ferramenta de modelagem gratuita para projeto de sistemas orientados a objeto. É baseada nos diagramas e na notação da UML 2.0 (*Unified Modeling Language*) e pode gerar código em Java. A Figura 6 apresenta a interface principal dessa ferramenta que é basicamente composta por uma área de edição, à direita, na qual são colocados os componentes gráficos para a representação dos modelos. Esses componentes são pré-definidos e estão representados na barra imediatamente superior à área de edição. Os componentes são apresentados de acordo com o tipo de modelo de diagrama escolhido.

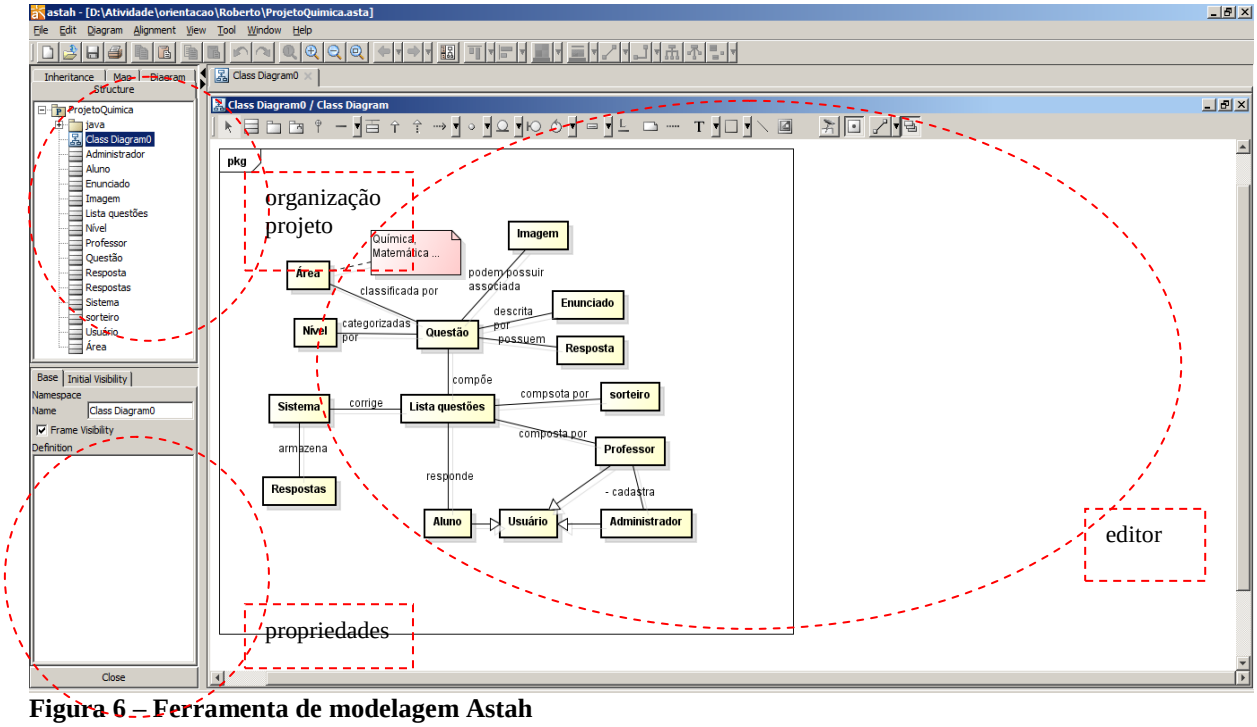


Figura 6 – Ferramenta de modelagem Astah

A ferramenta Astah Community está dividida em três partes principais, como pode ser visualizado na Figura 6 pelas áreas circuladas:

a) Organização do projeto (área superior à esquerda da Figura 6) - é a área na qual estão as diferentes visões do projeto, são elas: *Structure* que é a árvore de estrutura do projeto,

Inheritance que exibe as heranças identificadas, *Map* para mostrar todo o editor de diagrama e *Diagram* que mostra a lista de diagramas do projeto.

b) Visão das propriedades (área inferior à esquerda da Figura 6) - é a área na qual podem ser alteradas as propriedades dos elementos do diagrama. As propriedades de um item selecionado são exibidas e podem ser editadas.

c) Editor do diagrama (área à direita da Figura 6) - é nessa área que são exibidos e compostos os diagramas. Ao ser selecionado um determinado diagrama, dos constantes na lista, o mesmo é carregado e todos os seus elementos gráficos são mostrados nesta área.

A ferramenta Astah Community gera código na linguagem Java, mas apenas para a definição da classe e de seus atributos e métodos. Para que essa geração de código possa ser feita, o diagrama de classes deve estar pronto, as classes bem definidas com a especificação dos seus atributos e os métodos precisam ter os seus parâmetros definidos.

3.1.2 Adobe Flex 4 SDK

Flex utiliza uma linguagem de marcação, a MXML (*Maximum Experience Markup Language*) e a ActionScript. ActionScript 3.0 é a linguagem que define a lógica da aplicação que executa no Flash *player*. Flex possui vários componentes MXML (FUKUDA, YAMAMOTO, 2008), como: *TextInput*, *DataGrid*, *Button*; e componentes RPC (Remote Procedure Call), como: *HttpRequest*, *WebService*, *RemoteObject*. As aplicações Flex possuem a extensão *.mxml* e podem ser criadas em qualquer editor de texto que permita a produção de texto não formatado.

Aplicações Flex trabalham com o modelo de arquitetura orientada por eventos. Em Flex há duas formas de associar eventos com seus manipuladores (FUKUDA, YAMAMOTO, 2008):

a) Cada componente MXML possui seus próprios eventos de envio como nomes de atributos e funções de manipulação de eventos que são definidos como valores de atributos.

b) Cada componente MXML possui um atributo identificador. Assim, é possível identificar cada componente, referir-se ao mesmo por meio dos atributos desse identificador. E utilizar um método *addEventListener* para essas associações.

3.1.3 Adobe Flash Builder 4

O Adobe Flash Builder 4™ é uma ferramenta de desenvolvimento baseada no Eclipse, da IBM (*International Business Machines*) (ADOBE, 2011). O Adobe Flash Builder 4™ é um ambiente para o desenvolvimento da interface do *software* e os seus componentes podem ser facilmente estendidos e customizados. O resultado é uma aplicação que pode executar em ambiente *web*, *desktop* e em dispositivos portáteis que suportem o Flash Player.

O Adobe Flash Builder 4™ é um *framework* de desenvolvimento baseado em componentes. Esses componentes podem ser arrastados e soltados e ter seus parâmetros configurados. Contudo, também, é permitida a programação no modo de código.

O Flex oferece dois ambientes ao desenvolvedor. Um deles é o modo projeto (*design*) que habilita o modo de trabalho visual, permitindo o uso de componentes padrão e de componentes customizados. A interface do projeto é configurada pelo usuário, assim como o comportamento, e os efeitos dos componentes são definidos utilizando a paleta de propriedades. O outro é o modo *source* que exibe o código fonte, os componentes visuais são transformados em *tags* MXML e seguem uma hierarquia no padrão XML (*Extensible Markup Language).* A Figura 7 exibe a tela inicial do Flash Builder 4.

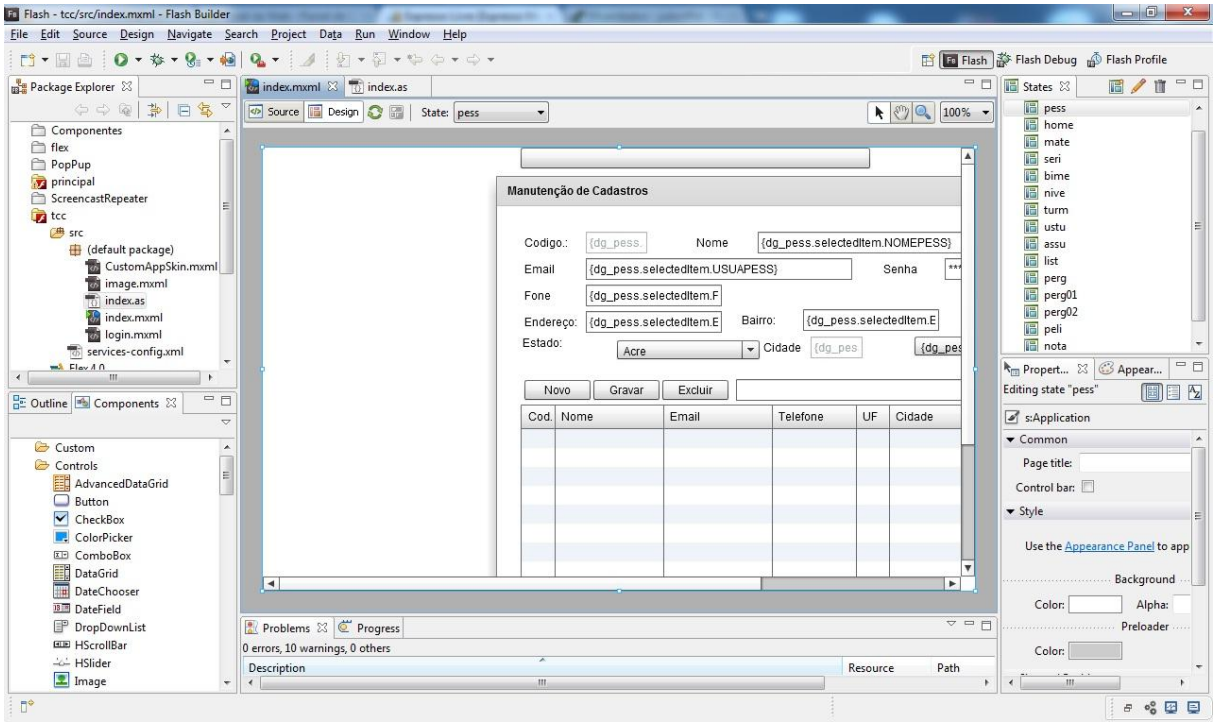


Figura 7 – Tela inicial do Flash Builder 4

As principais funcionalidades dessa ferramenta são:

a) *Perspectives* - nesta aba esta as perspectivas que podem ser Flex Development Perspective, Flex Debugging Perspective entre outras, principalmente se ele estiver instalado como *plugin* no Eclipse.

b) *Flex navigator* - exibe os projetos do *workspace* e os arquivos dos projetos.

c) *Outline* - caso esteja em modo *source* exibe os métodos e as variáveis, caso seja o modo *design* então exibe a hierarquia dos componentes partindo da aplicação.

d) *Components* – exibe os componentes padrão que acompanham o Flex Builder e também os componentes criados ou estendidos pelo usuário.

e) *Flex properties* - exibe as propriedades do objeto visual selecionado, podendo ser customizado facilmente. É possível que o desenvolvedor crie novas propriedades e estas apareçam na paleta.

f) *States* - são estados diferentes da aplicação, como se fossem nós da aplicação.

g) *Web application* - esta opção compilará o código fonte para ser executado pelo Flash *player*.

h) *Desktop application* - compila o código fonte para ser executado pelo Adobe AIR (*Adobe Integrated Runtime)* instalado na máquina do cliente e executando como um sistema *desktop*, podendo utilizar alguns recursos a mais que uma *web application*, como, por exemplo, o acesso à lista de pastas do sistema.

3.1.4 AMFPHP

AMFPHP é uma implementação PHP de código fonte aberto (*free open source*) do AMF (*Action Message Format*) (AMFPHP, 2011). AMF permite a serialização binária de tipos nativos e objetos ActionScript em suas versões 2 e 3 para serem enviados para o servidor.

O AMFPHP permite que aplicações com clientes simples (*thin client*) construídas em linguagens como Flash, Flex e AIR comuniquem-se diretamente com classes de objetos PHP no servidor.

AMFPHP é um dos protocolos de comunicação disponíveis para Flash *player* mais rápidos que existem porque a comunicação é serializada em formato binário, que é mais compacto que outras representações, como XML (AMFPHP, 2011). Além disso, o AMF3 disponível em ActionScript 3 também compacta a comunicação binária para aumentar o desempenho.

A opção pelo uso da tecnologia AMFPHP Remote Object decorre da sua velocidade de transferência de dados e a compatibilidade com servidores Apache responsável pela grande maioria dos servidores web disponíveis no mercado. A Figura 8 fundamenta essa afirmativa.

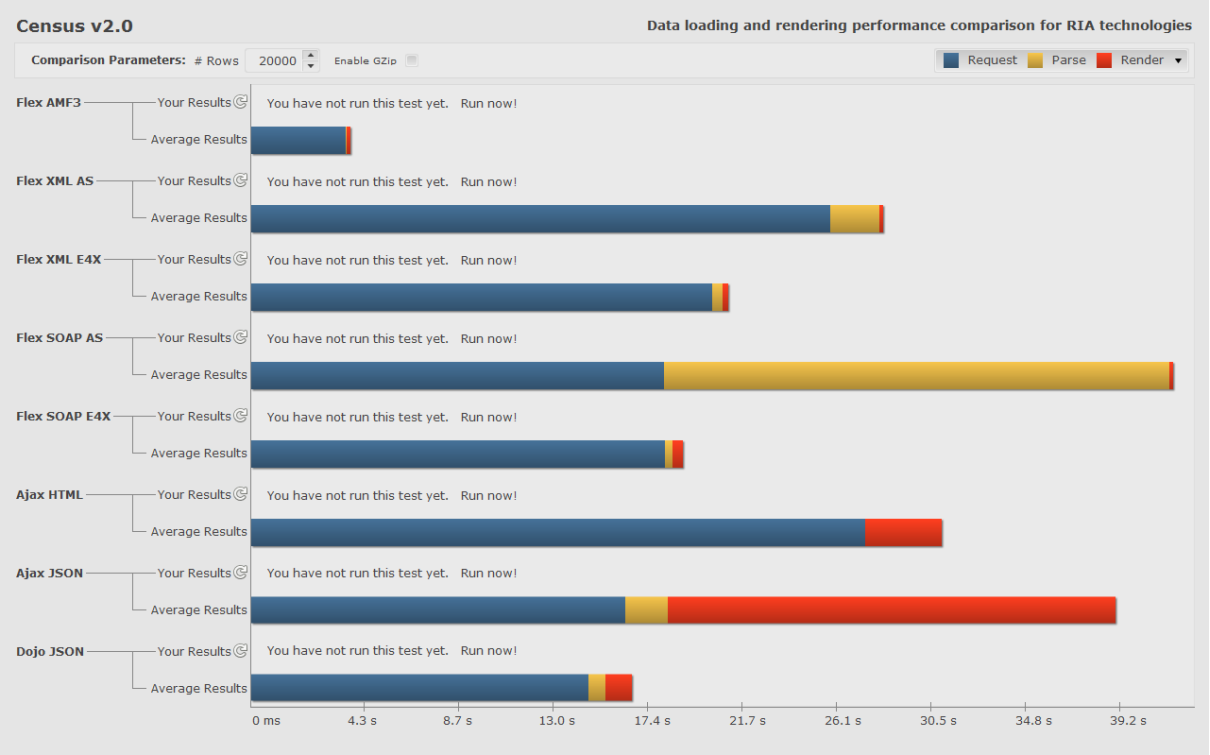


Figura 8 – Comparativo de desempenho entre métodos de transferência de dados
Fonte: http://www.jamesward.com/census2/

Action Message Format (AMF) funciona por meio de HTTP, como o XML, mas em vez de enviar diversas informações extras, o AMF envia as informações essenciais retirando os caracteres desnecessários, ou seja, é serializado dentro de um formato binário que bem menos espaço do que um mesmo dado enviado com XML. Isso é útil quando muitos dados são enviados. Em vez de enviar várias *tags* redundantes em XML, são enviados pequenos registros AMF serializados.

Outro benefício do AMF é que este é um tipo de dados nativo do Flash Player. Com isso não necessita fazer um "*parse*" dos resultados para obter os dados. Para usar AMF é necessário trabalhar com objetos no PHP. O *gateway* AMFPHP ajuda a traduzir esses objetos PHP em objetos ActionScript no Flash Player. Com isso é possível criar os próprios objetos tipados no PHP e manter uma cópia desses objetos no ActionScript.

3.1.5 Linguagem PHP

PHP é uma linguagem de programação dinâmica para *web* (PHP, 2011) que é processada no servidor, retornando para o cliente somente HTML. Desta forma o código fonte não é exposto e isso é importante na interação com banco de dados ou outros componentes que possuam informações sigilosas que precisem ser especificadas no código.

A linguagem PHP tem suporte a praticamente todos os bancos de dados existentes no mercado, o que torna simples a integração com aplicações que necessitem desta tecnologia. A linguagem também suporta outros protocolos como SMTP (*Simple Mail Transfer Protocol)*, POP3 (*Post Office Protocol)* e IMAP (*Internet Message Application Protocol*).

3.1.6 MySQL

O MySQL é um servidor e gerenciador de banco de dados relacional que utiliza a linguagem SQL (MYSQL, 2011). Ele é de licença dupla (*software* livre e paga), projetado inicialmente para trabalhar com aplicações de pequeno e médio portes, mas atualmente atende também aplicações de grande porte.

As principais características incorporadas na versão 5 do MySQL (MILANI, 2007) são:

a) Visões - são consultas pré-programadas ao banco de dados que permitem unir duas ou mais tabelas e retornar uma única tabela como resultado. Além disso, podem ser utilizadas para filtrar informações, exibindo somente os dados específicos de uma determinada categoria de uma ou mais colunas da tabela. Com o uso de visões, operações frequentes com uniões de tabelas podem ser centralizadas. É possível também utilizá-las para controle de acesso, permitindo que determinados usuários acessem dados de uma visão, mas não as tabelas utilizadas para compor a visão, restringindo acesso a informações.

b) Cursores - cursores possibilitam a navegação em conjuntos de resultados. De forma simples, é possível navegar pelos registros de uma tabela a partir de laços de repetição, permitindo realizar operações necessárias e transações à parte para cada linha da tabela.

c) *Information Schema* - são tabelas responsáveis apenas pela organização dos recursos do banco de dados, conhecidos como dicionário de dados, ou metadados. Desta forma, é possível realizar consultas sobre a estrutura do banco de dados por meio dessas tabelas.

d) Transações distribuídas XA - são uma espécie de extensão da ACID (Atomicidade, Consistência, Isolamento, Durabilidade) que fornece a possibilidade de gerenciamento dessas transações realizadas com a união de múltiplos bancos de dados (transações distribuídas) para a execução de uma mesma transação. Por exemplo, em determinadas situações pode surgir a necessidade de integração de duas bases de dados distintas, mas que em algum momento necessitem uma da outra para realizar uma operação.

e) Integridade referencial - os relacionamentos entre tabelas distintas são gerenciados pelo banco de dados quando de inclusão, alteração e exclusão. Esse recurso visa manter as relações existentes no banco de dados confiáveis.

f) Clusterização - a clusterização é baseada na integração e sincronismo de dois ou mais servidores para dividirem a demanda de execuções entre si. Além da sincronização de um *cluster*, é possível especificar um balanceador de cargas. Desta forma, esse recurso gerenciará o redirecionamento de servidores secundários no caso de parar o funcionamento e balanceará as consultas recebidas pelo *cluster*, distribuindo-as pelos servidores de acordo com sua disponibilidade.

3.1.7 MySQL Workbench

MySQL Workbench (WORKBENCH, 2011) é uma ferramenta gráfica para modelagem de dados. A ferramenta possibilita trabalhar diretamente com objetos *schema*, além de fazer a separação do modelo lógico do catálogo de banco de dados.

Toda a criação dos relacionamentos entre as tabelas pode ser baseada em chaves estrangeiras. Outro recurso que a ferramenta possibilita é realizar a engenharia reversa de esquemas do banco de dados, bem como gerar os *scripts* em SQL.

Com essa ferramenta, a modelagem do banco de dados pode assumir níveis conceituais, lógicos e físicos. MySQL Workbench apresenta uma arquitetura extensível, sendo possível visualizar a representação de tabelas, funções, entre outros.

A Figura 9 (um *print screen* do MySQL Workbench) apresenta os três serviços disponíveis pelo MySQL Workbench, circulado pontilhado.

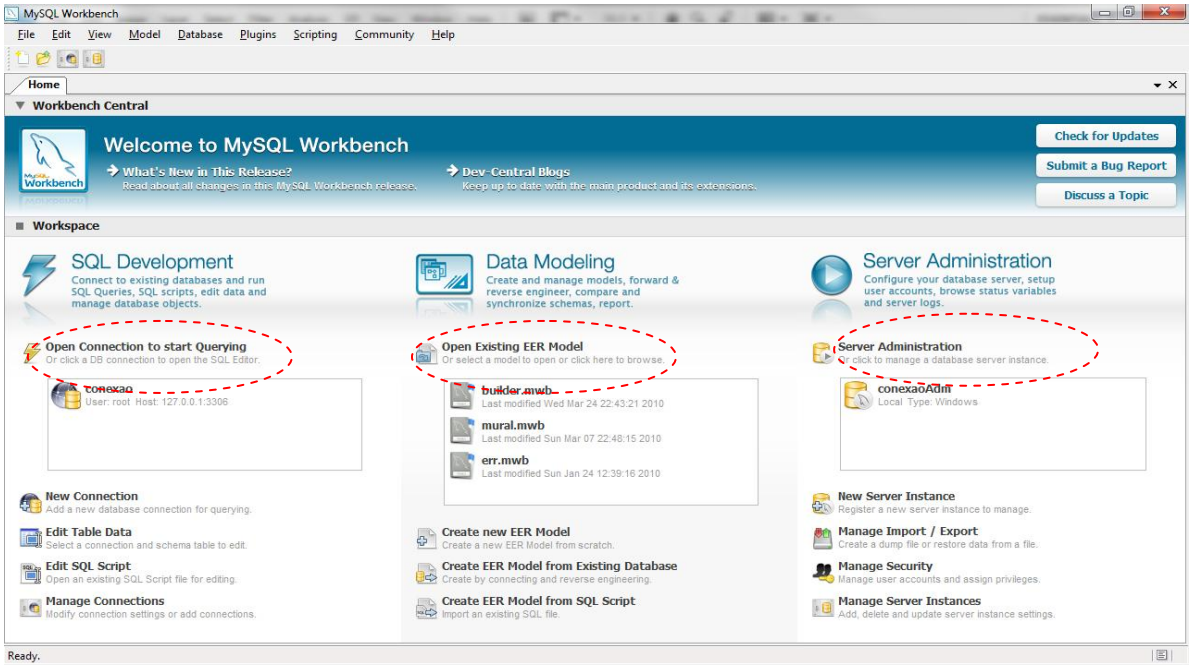


Figura 9 – Tela dos serviços do MySQL Workbench

As áreas destacadas da Figura 9:

a) *SQL Development* é utilizado para conexão a bases de dados registradas no servidor possibilitando a execução de *query*, *scripts* SQL e editar dados dessas bases. O editor SQL informa o usuário de erros de sintaxe automaticamente.

b) *Data Modeling* possibilita ao usuário criar e gerenciar modelos de dados, executar engenharia reversa utilizando uma base de dados registrada para gerar um novo diagrama, gera os *scripts* completos ou apenas os que o usuário escolher para executar no *SQL Development*. Além disso, é possível criar um modelo de dados é simples, bastando arrastar e soltar as tabelas. Um duplo clique em uma tabela torna as propriedades da referida tabela visíveis permitindo adicionar colunas, índices e demais componentes.

c) *Server Administration* contém as configurações gerais do servidor, manipulação de usuários, controle de portas e conexões do sistema e do servidor, *backups* e restaurações de bases de dados.

A área superior da tela destina-se à edição e modelagem dos dados. Para inserir uma tabela, basta clicar no botão “The Table Tool” e arrastá-lo para a página de edição. As tabelas criadas ficam disponíveis em uma aba central, sendo possível editar os atributos, colunas, chaves primárias e estrangeiras, entre outros. Após modelar é necessário clicar na opção “Apply Changes” para aplicar as alterações efetuadas.

3.1.8 Apache

O Apache é um servidor *web* de código fonte aberto, configurável, compatível com linguagens dinâmicas como PHP e CGI, disponível para vários sistemas operacionais incluindo Windows e Unix e é gratuito mesmo para uso comercial (APACHE, 2011). Tem suas origens em 1995, criado por Rob McCool e em maio de 2011 era responsável por 62,71% dos servidores ativos no mundo (NETCRAFT, 2011).

3.2 MÉTODO

O método utilizado para a realização do trabalho está baseado nas fases de análise, projeto, codificação e testes do modelo sequencial linear exposto em Pressman (2005). Essas fases foram utilizadas como base, mas sofreram inclusões e alterações para adaptar-se aos interesses deste projeto. Ressalta-se que as atividades realizadas não foram estritamente sequenciais. A realização de teste permitia validar partes do sistema à medida que eram implementadas. E desta forma, ciclos iterativos e incrementais ocorriam.

As principais fases ou etapas definidas para o ciclo de vida do sistema são:

a) Planejamento – no planejamento foram definidas as principais etapas e atividades para a realização do projeto. Esse planejamento foi feito com a orientadora visando desenvolver os sistemas e a monografia em paralelo. Os sistemas, porque o acadêmico Roberto Rosin implementou uma versão *desktop* desse mesmo sistema. Reuniões periódicas permitiam avaliar e ajustar as atividades planejadas e estabelecer prazo para o término de cada atividade que seria realizada. Reuniões também ocorreram entre o autor deste trabalho e Roberto Rosin, isso porque os requisitos para o sistema deveriam ser os mesmos. E os dados obtidos com as respostas dos testes deveriam ser integrados.

Uma reunião foi realizada com o professor de Química da Universidade e suas duas alunas orientadas que usariam o sistema. Essa reunião serviu para que o autor deste trabalho e o acadêmico Roberto Rosin pudessem expor o entendimento do sistema a partir dos requisitos que haviam sido inicialmente solicitados. Nessa reunião também foram resolvidas algumas dúvidas dos acadêmicos e foi estabelecido um entendimento geral da forma de apresentação dos dados obtidos com as respostas pelos alunos às listas de exercícios.

b) Requisitos – a definição dos requisitos do *software* foi realizada com a ajuda da orientadora, do professor de Química que solicitou a implementação dos sistemas e das duas alunas orientadas desse professor que utilizarão o sistema.

c) Análise – inicialmente foi definido um diagrama que representasse a visão geral do sistema e em seguida foram definidos os seus requisitos, modelados sob a forma de caso de uso e descrições suplementares. Esses requisitos foram complementados com aspectos de qualidade, definindo os requisitos não funcionais.

d) Projeto – tendo como base os requisitos definidos na fase de análise, foram definidos os modelos para solucionar o problema, ou seja, definir como o sistema atenderia aos requisitos (funcionalidades e requisitos não funcionais) definidos na análise. Essa solução foi modelada basicamente por meio de diagramas de classes e entidade-relacionamento da base de dados. Para a modelagem dos dados foi utilizada a ferramenta MySQL Workbench.

e) Implementação – na implementação foi realizada a codificação do sistema e realização de testes pelo programador. A codificação baseou-se na criação da interface do sistema com componentes Flex, a implementação das regras de negócio (camada da aplicação) com a linguagem PHP, utilizando AMFPHP para a transformação (serialização) dos objetos Flex para objetos PHP e vice-versa (desserialização). O MySQL foi utilizado como banco de dados.

f) Testes – os testes foram informais e realizados pelo próprio programador, o autor deste trabalho. Os testes de requisitos e de uso foram realizados pelas alunas de Química e pelo seu professor orientador. Esses testes consistiram no uso do sistema para verificar se o mesmo atendia os interesses e necessidades do trabalho que pretendiam realizar com alunos de Ensino Médio.

4 RESULTADOS

Este capítulo apresenta o sistema que foi desenvolvido como resultado deste trabalho. Inicialmente é apresentada a descrição do mesmo. Em seguida está a modelagem do sistema. E, por fim, implementação do sistema. A modelagem apresenta alguns diagramas como resultado da análise e do projeto. Na implementação estão exemplos de códigos gerados.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema modelado e desenvolvido como resultado deste trabalho permite o armazenamento de questões de múltipla escolha e a composição de listas ou testes a partir dessas questões armazenadas. Cada questão possui um enunciado, que pode ter uma imagem associada e um conjunto de cinco alternativas, uma das quais é a alternativa correta. As questões são armazenadas classificadas por bimestre, série e nível de dificuldade.

O sistema possui três níveis de acesso (usuários). O Administrador com acesso a todas as funcionalidades do sistema. Professor que apenas não pode incluir usuários professor. E aluno que tem acesso somente aos testes para serem respondidos que são atribuídos a ele e ao resultado da correção dos testes que ele responde. O sistema provê a correção dos testes e informa o aluno do resultado.

4.2 MODELAGEM DO SISTEMA

A Figura 10 apresenta uma visão geral do sistema como um conjunto de conceitos relacionados entre si. Alguns desses conceitos representam tabelas e ou classes do sistema ou mesmo atributos e outros simplesmente visam facilitar o entendimento de funcionalidades e do escopo e contexto do mesmo.

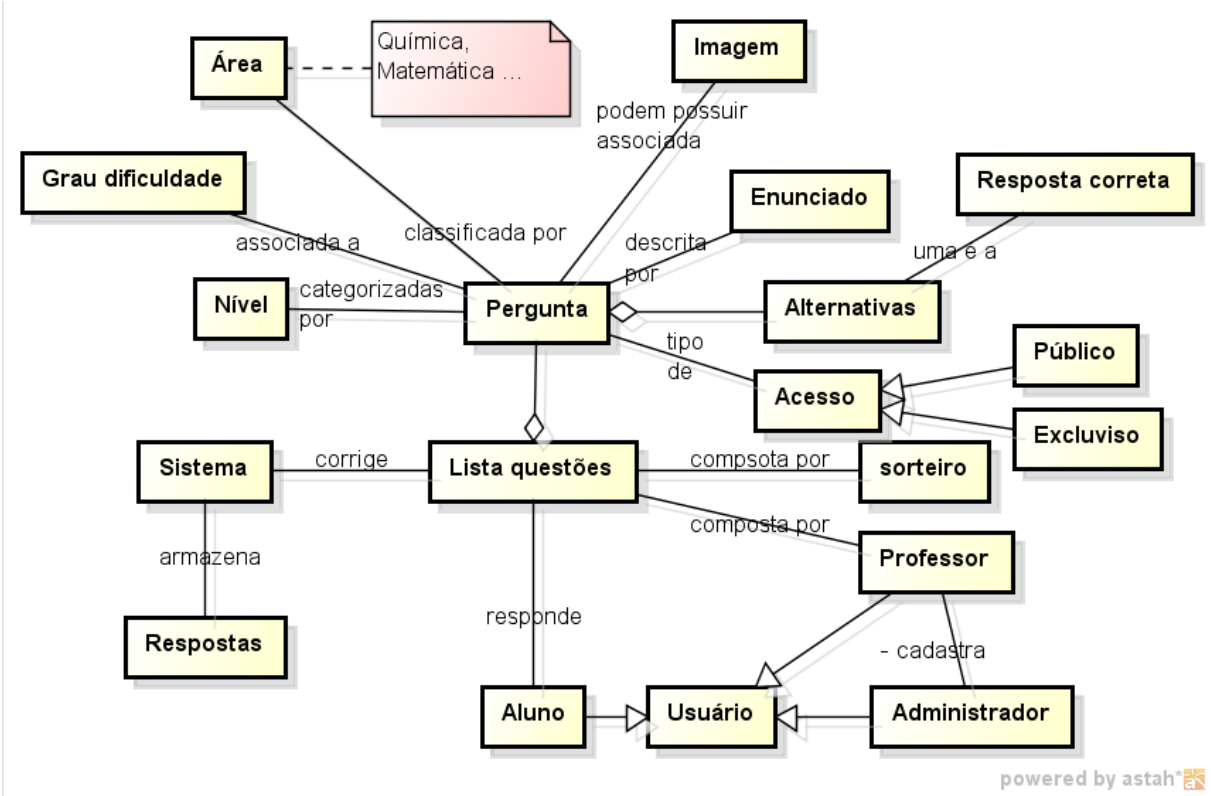


Figura 10 – Visão geral do sistema

Pela representação da Figura 10, uma questão pertence a uma área, está categorizada por um nível e é associada a um grau de dificuldade. A categorização da área da questão facilita a composição de testes quando o sistema é utilizado por áreas distintas. A categorização por nível (por exemplo, 1º ano do ensino médio, 2º ano do ensino médio, 3º ano do ensino médio e pré-vestibular) e por grau de dificuldade (como fácil, médio e difícil) que auxiliam na composição das listas.

Uma pergunta é descrita por um enunciado e é composta por um conjunto de alternativas. Sendo que uma dessas alternativas é a resposta correta à questão. O enunciado da questão é um texto que pode ter uma imagem associada. Cada pergunta pode ter seu acesso definido como público ou exclusivo. O acesso é público quando pode ser acessada por todos os usuários do sistema que são do tipo professor ou administrador. O acesso é exclusivo quando a referida pergunta somente pode ser acessada pelo usuário que a inseriu.

Uma lista de questões é uma composição de questões com o respectivo enunciado, conjunto de alternativas e com imagem associada, se houver. A alternativa que determina a resposta correta é utilizada na correção do teste. Uma lista de questões pode ser composta por sorteio pelo computador ou pelo usuário professor. A ordem das alternativas de cada pergunta é definida randomicamente.

O sistema possui como usuários, o administrador que cadastra professores, o professor e o aluno. O professor pode realizar cadastro (inclusão, exclusão, consulta e alteração) de questões, compor listas e verificar as respostas dos testes realizados. O usuário aluno somente responde testes e tem acesso à respectiva correção.

A Figura 11 apresenta o diagrama de casos de uso representado os requisitos funcionais definidos para o sistema.

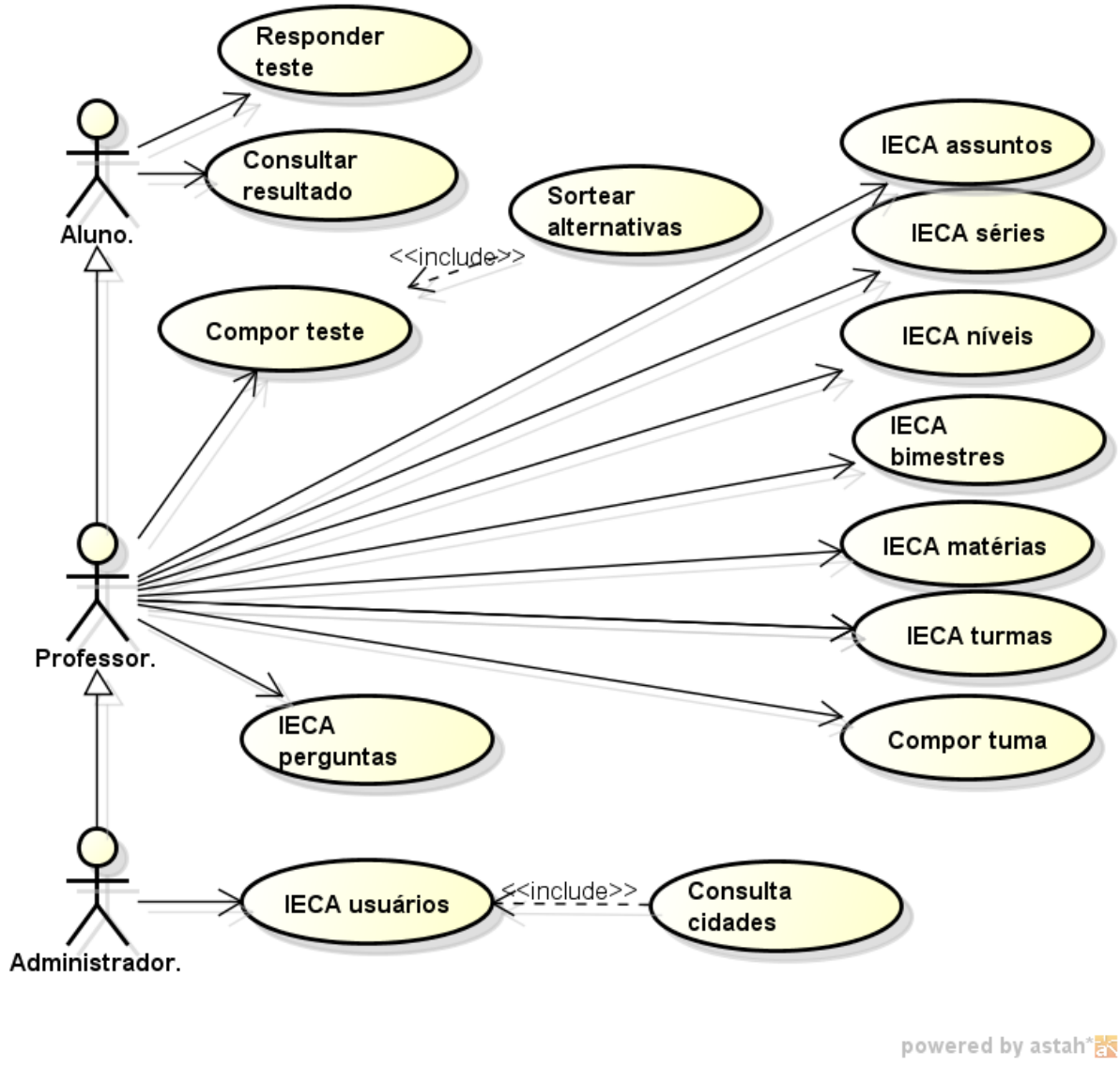


Figura 11 – Diagrama de casos de uso

Na Figura 11, a sigla IECA significa Inclusão, Exclusão, Consulta e Alteração. O Quadro 1 apresenta uma descrição sucinta dos casos de uso.

Caso de uso	Descrição
Responder teste	O aluno acessa um teste para ser respondido. E o sistema apresenta um teste para ser respondido. Quando o aluno submete o formulário com o teste o sistema faz a correção, informa o aluno do resultado da correção e armazena esses dados para relatórios posteriores.
Consultar resultados	O aluno pode consultar resultados de testes que ele respondeu.
Compor teste	O professor compõe um teste como um conjunto de questões. A composição é realizada a partir de perguntas cadastradas. O sistema faz o sorteio da ordem das alternativas.
IECA assuntos	Esse caso de uso se refere à inclusão de um assunto novo, exclusão de assuntos cadastrados, consulta de assuntos cadastrados e alteração de assuntos cadastrados.
IECA séries	Uma série se refere a um período escolar (como, por exemplo, 1º ano do Ensino Médio). Esse caso de uso ser refere à inclusão de uma série, exclusão de séries já cadastradas, consulta de séries cadastradas e alteração de dados de séries cadastradas.
IECA níveis	Níveis servem para categorizar as perguntas quanto ao grau de dificuldade, como fácil, médio e difícil. Esse caso de uso ser refere à inclusão de um nível, exclusão de níveis já cadastrados, consulta de níveis cadastrados e alteração de dados de níveis cadastrados.
IECA bimestres	Um bimestre está relacionado ao período escolar, como, por exemplo, 1º, 2º, 3º e 4º bimestre. Esse caso de uso ser refere à inclusão, exclusão, consulta e alteração de bimestres.
IECA matérias	Uma matéria se refere a uma disciplina escolar. Por exemplo, Física, Química e Matemática. Esse caso de uso ser refere à inclusão de matérias, exclusão e alteração de matérias já cadastradas e consulta de matérias.
IECA turmas	Uma turma é um agrupamento de alunos. Assim, um aluno sempre pertence a uma turma. Esse caso de uso ser refere à inclusão de turmas, exclusão de turmas já cadastradas, consulta de turmas cadastradas e alteração de dados de turmas cadastradas.
Consulta cidades	As cidades foram incluídas diretamente no banco de acordo com um catálogo disponibilizado pelo IBGE. O usuário pode consultar uma cidade e vinculá-la ao seu cadastro. Manutenção de cidades existentes, inclusão de novas e exclusão são realizadas diretamente no banco de dados.
Compor turma	A partir de uma turma cadastrada e de alunos, o professor compõe uma turma, vinculando alunos à mesma. Esse caso de uso se refere a associar alunos a uma turma.
IECA perguntas	Uma lista que é um teste é composto por um conjunto de perguntas. Esse caso de uso ser refere à inclusão de perguntas, exclusão de perguntas cadastradas, consulta de perguntas cadastradas e alteração de dados de perguntas cadastradas. Na inclusão de uma pergunta é possível vincular uma imagem e deve ser indicado se a mesma é de acesso público ou exclusivamente ao usuário que a incluiu.
IECA usuários	Um usuário tem acesso ao sistema para responder um teste e acessar o seu resultado, se aluno. Para todas as funcionalidades de negócio do sistema, se professor. E incluindo alteração de nível de acesso, se administrador.
Alterar tipo de usuário	O usuário administrador altera o nível de acesso do usuário aluno ou professor cadastrados.

Quadro 1 – Listagem dos casos de uso

Além dos requisitos funcionais definidos como casos de uso, foram identificados os seguintes requisitos não funcionais:

a) Cada pergunta deve ter uma e somente uma alternativa correta;

b) Toda pergunta deve possuir cinco alternativas;

c) O sistema sorteará randomicamente as alternativas de cada pergunta no momento da composição de um teste;

d) Cada pergunta deverá ter definido se o seu acesso é público ou restrito ao usuário que a inseriu;

e) O usuário aluno tem acesso somente às respostas (correção) dos testes que ele realizou.

A Figura 12 apresenta o diagrama de classes do sistema.

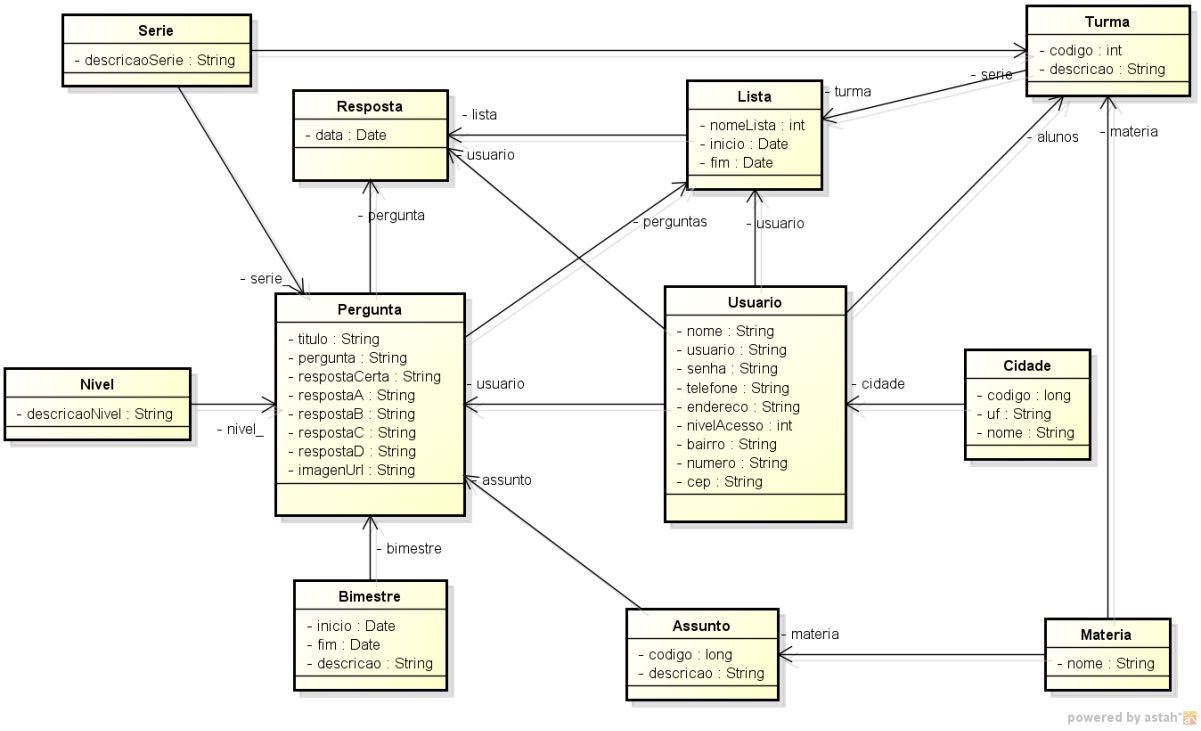


Figura 12 – Diagrama de classes

De acordo com a Figura 12 é possível identificar que as classes Pergunta e Lista são as principais, se considerado como parâmetro a quantidade de ligações que elas possuem. A classe Pergunta contém o título da pergunta, a descrição do enunciado da pergunta, as alternativas e a resposta correta. Essa classe está associada com as classes Bimestre, Nível (que estabelece o grau de dificuldade), Série escolar, Assunto e a Resposta correta. O assunto está associado a uma Matéria. Isso permite que o sistema seja utilizado simultaneamente por disciplinas escolares distintas, como Física, Química e Biologia.

A classe Lista está associada à Pergunta, Resposta, Usuário (um usuário responde uma lista) e Turma. A classe Usuário também está associada à Resposta para que seja possível o usuário visualizar o resultado da correção do teste que ele realizou. E, ainda, a turma, porque um usuário aluno pertence a uma turma. Assim, é possível realizar análises de resultados por turma. A classe cidade é utilizada para a inclusão do endereço do usuário.

Na Figura 13 está o diagrama de entidades e relacionamentos do banco de dados.

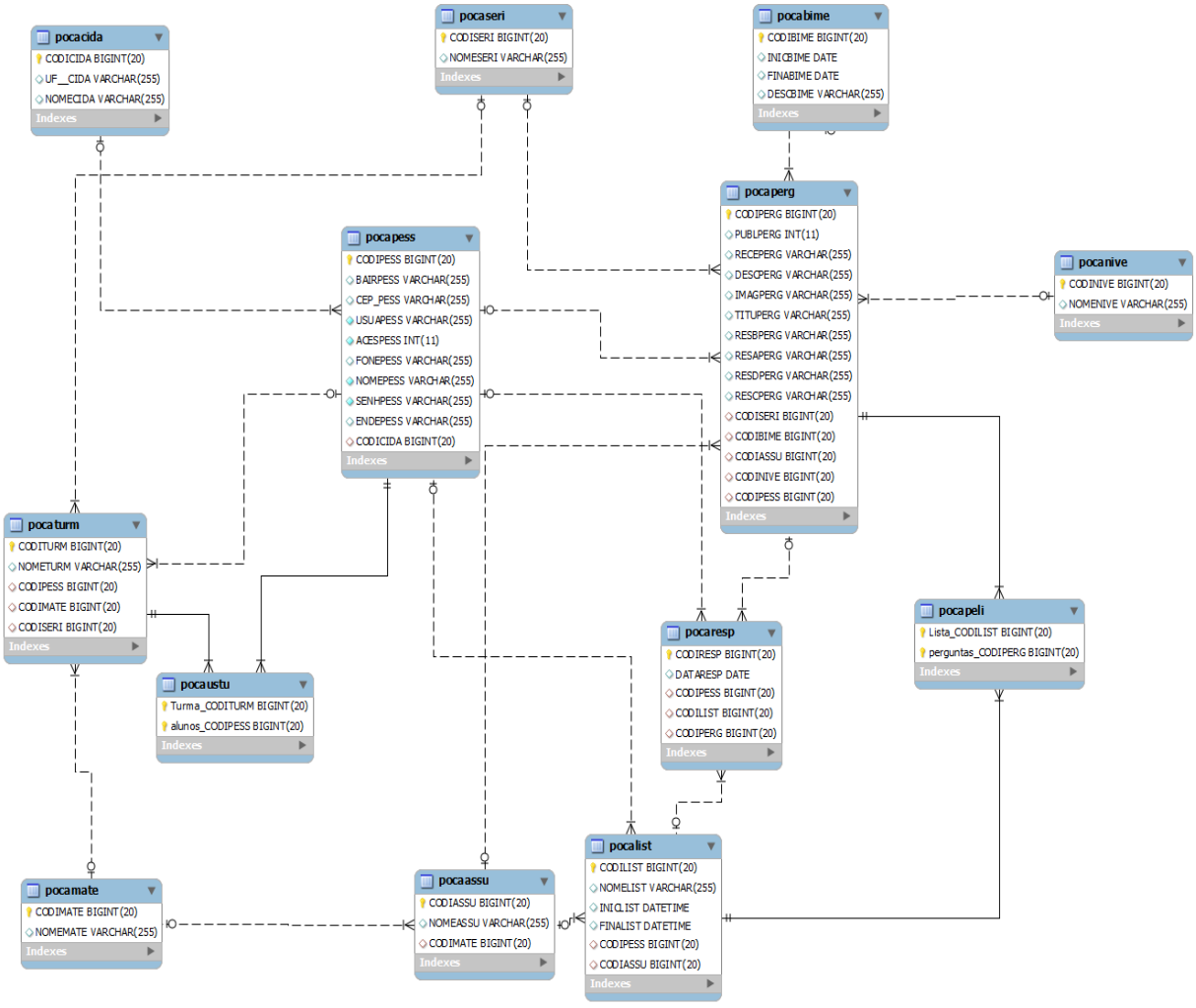


Figura 13 – Diagrama de entidades e relacionamentos do banco de dados

4.3 DESCRIÇÃO DO SISTEMA

A Figura 14 apresenta a organização do projeto. O SWF é compilado a partir do arquivo MXML. Na pasta “amfphp” estão as classes php e em “gateway.php” está o *gateway* AMFPHP.

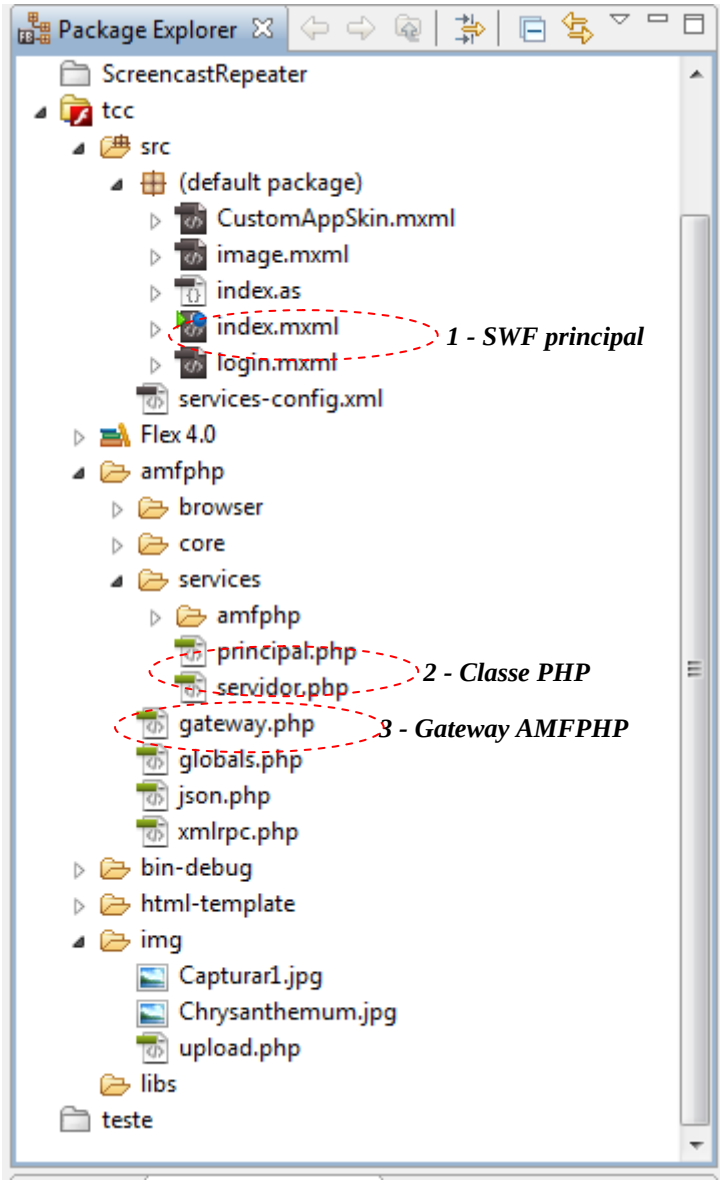
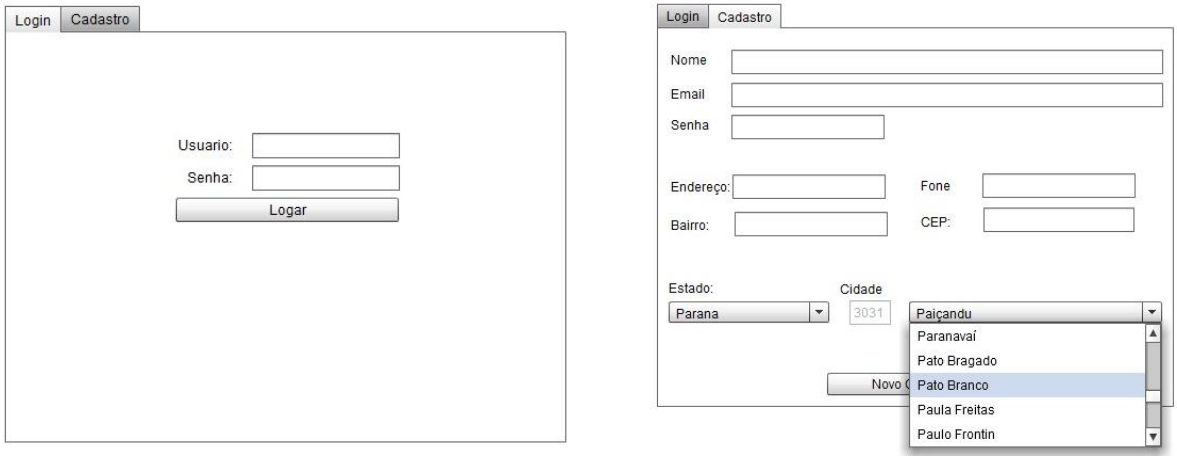


Figura 14 – Arquivos fontes da aplicação

O sistema é dividido em duas partes. A parte administrativa e a parte usuários. Para ter acesso ao sistema, o aluno deve estar previamente cadastro. A Figura 15 apresenta a tela de *login* ao sistema e de cadastro de usuários. Nesse cadastro há uma caixa de combinação para selecionar o Estado. A partir de um Estado selecionado é alimentada uma caixa de combinação com os Municípios do respectivo Estado. Cada Cidade (Município) possui associado um código numérico padronizado pelo Instituto Brasileiro de Geografia e Estatística. Esse código é apresentado em um campo não editável a partir do Município selecionado.



(a) Tela de cadastro (b) Tela de login

Figura 15 – Tela de cadastro e de login

Depois de efetuado o *login*, o sistema direciona o usuário para a tela de atividades em aberto e calendário das próximas atividades (Figura 16). Essa tela permite selecionar uma avaliação e respondê-la.

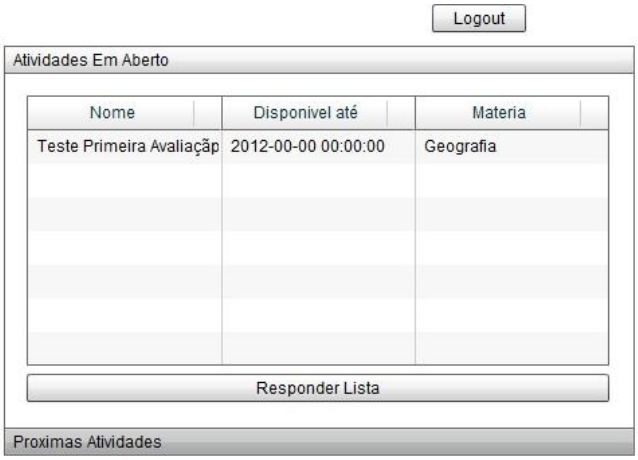


Figura 16 – Tela inicial com a lista de atividades em aberto e próximas atividades

Na Figura 17 é possível visualizar uma questão da maneira em que é apresentada ao usuário. O questionamento é apresentado e permite ao aluno selecionar uma resposta salvando a questão, antes de passar para a próxima.

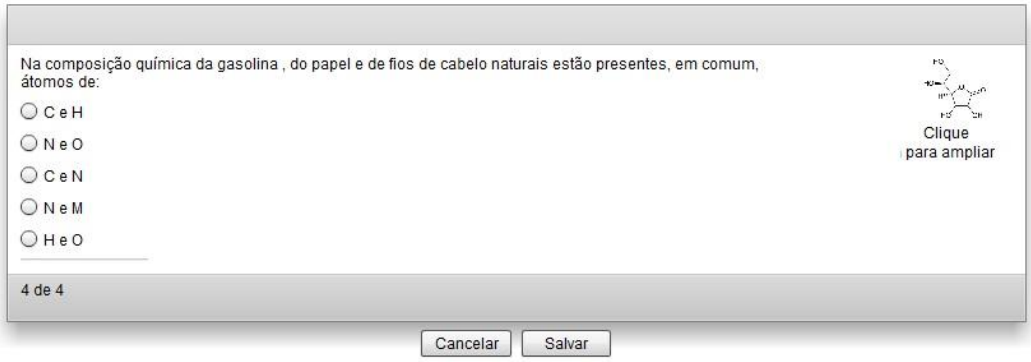


Figura 17 – Tela de avaliação

A apresentação da questão possui um de *zoom* da imagem (Figura 18).

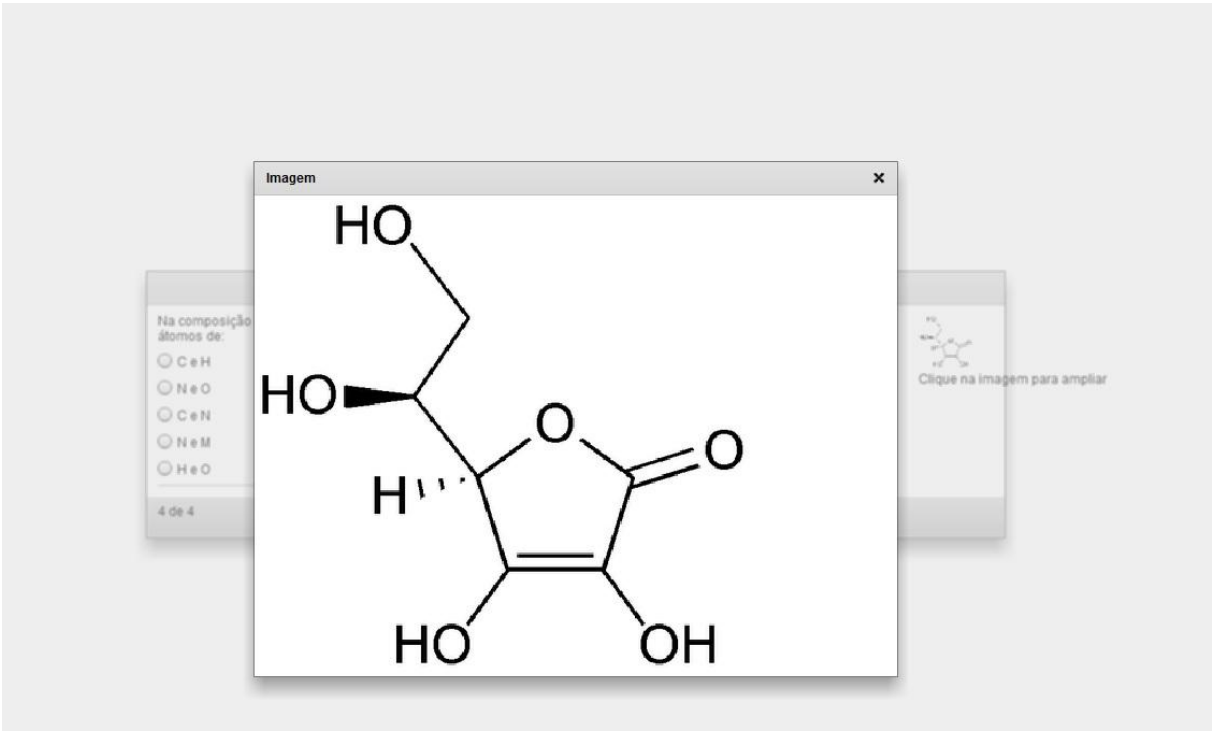


Figura 18 – Zoom da imagem na tela de avaliação

Após a conclusão da avaliação o usuário recebe um alerta, notificando-o que sua avaliação chegou ao fim. A interface de usuários permite ao aluno responder as questões. Outro recurso do sistema é a interface administrativa, que permite aos usuários incluir atividades e todas as informações relacionadas e gerenciar usuários e avaliações. Além de consultar, imprimir e salvar resultados das avaliações realizadas.

A Figura 19 apresenta a tela de inclusão de novas perguntas.

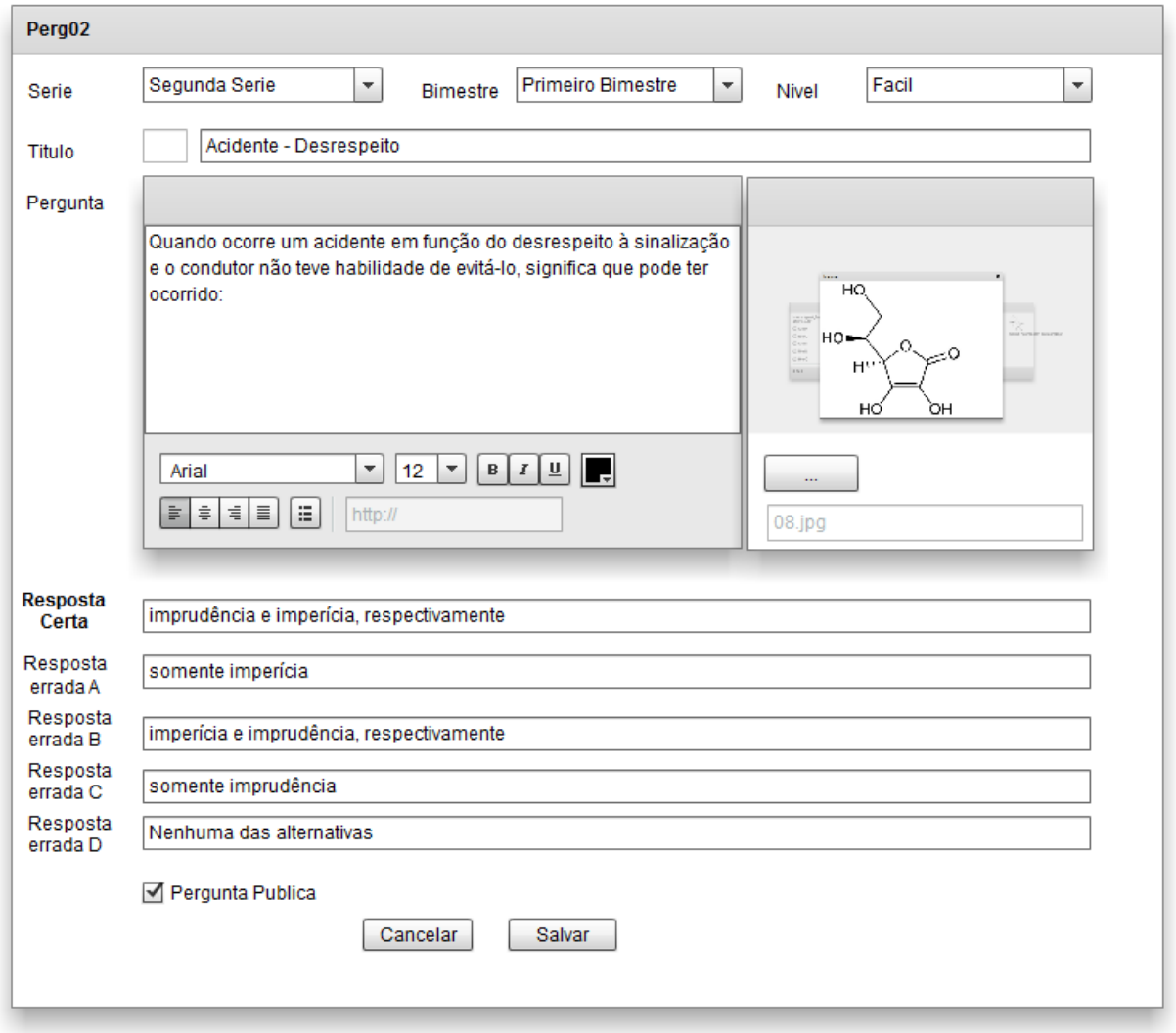


Figura 19 – Tela inclusão de perguntas

A interface administrativa permite a inclusão, alteração e exclusão de séries, bimestres, níveis de dificuldade, matérias, assuntos, usuários, turmas, avaliações e perguntas. Na Figura 20 apresenta a vinculação dos alunos a uma turma.

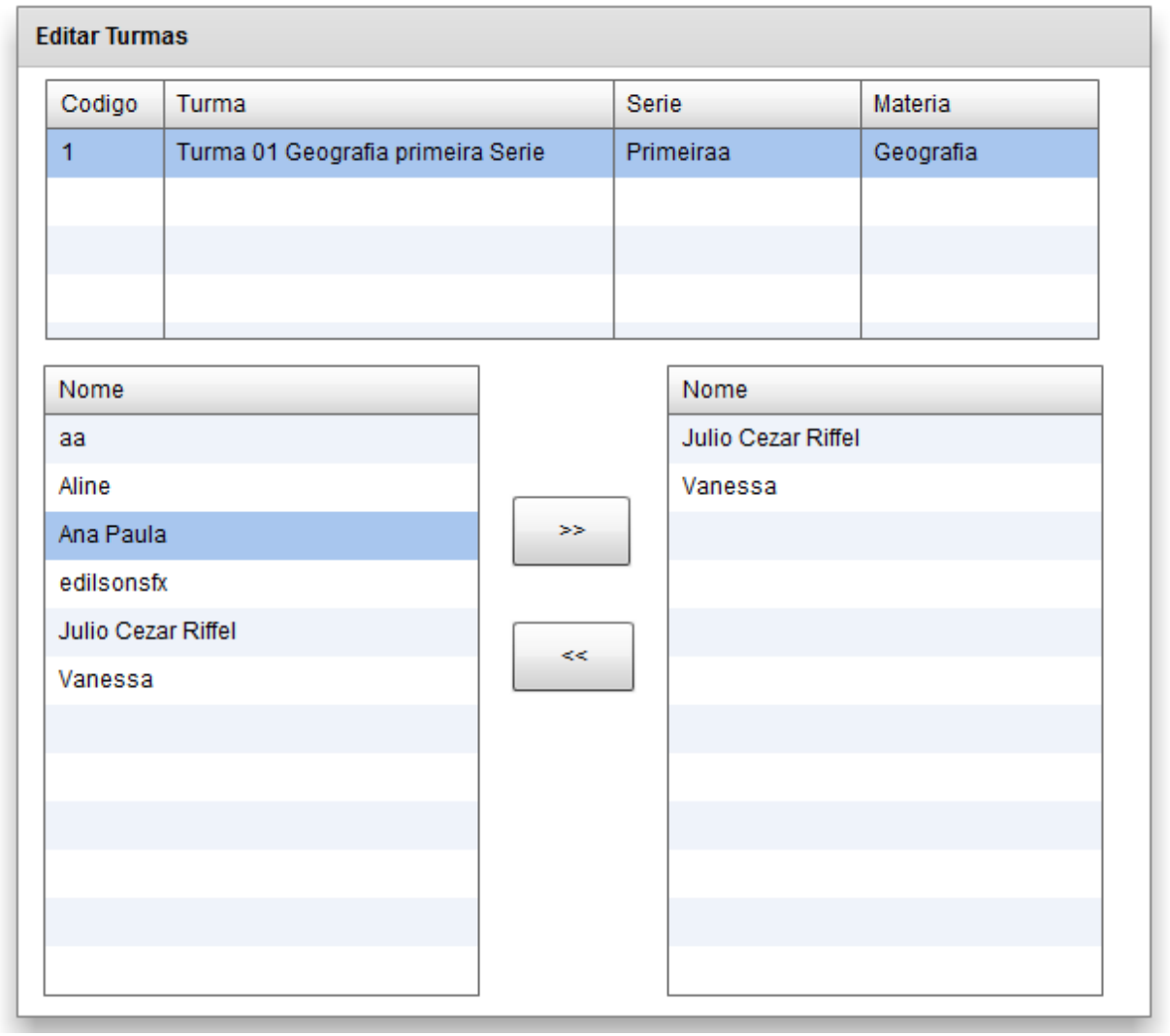


Figura 20 – Tela inclusão de aluno em turmas

Para uma mesma turma é possível geral várias listas (avaliações), agendando a data de disponibilização e fechamento de cada lista de questões (Figura 21).

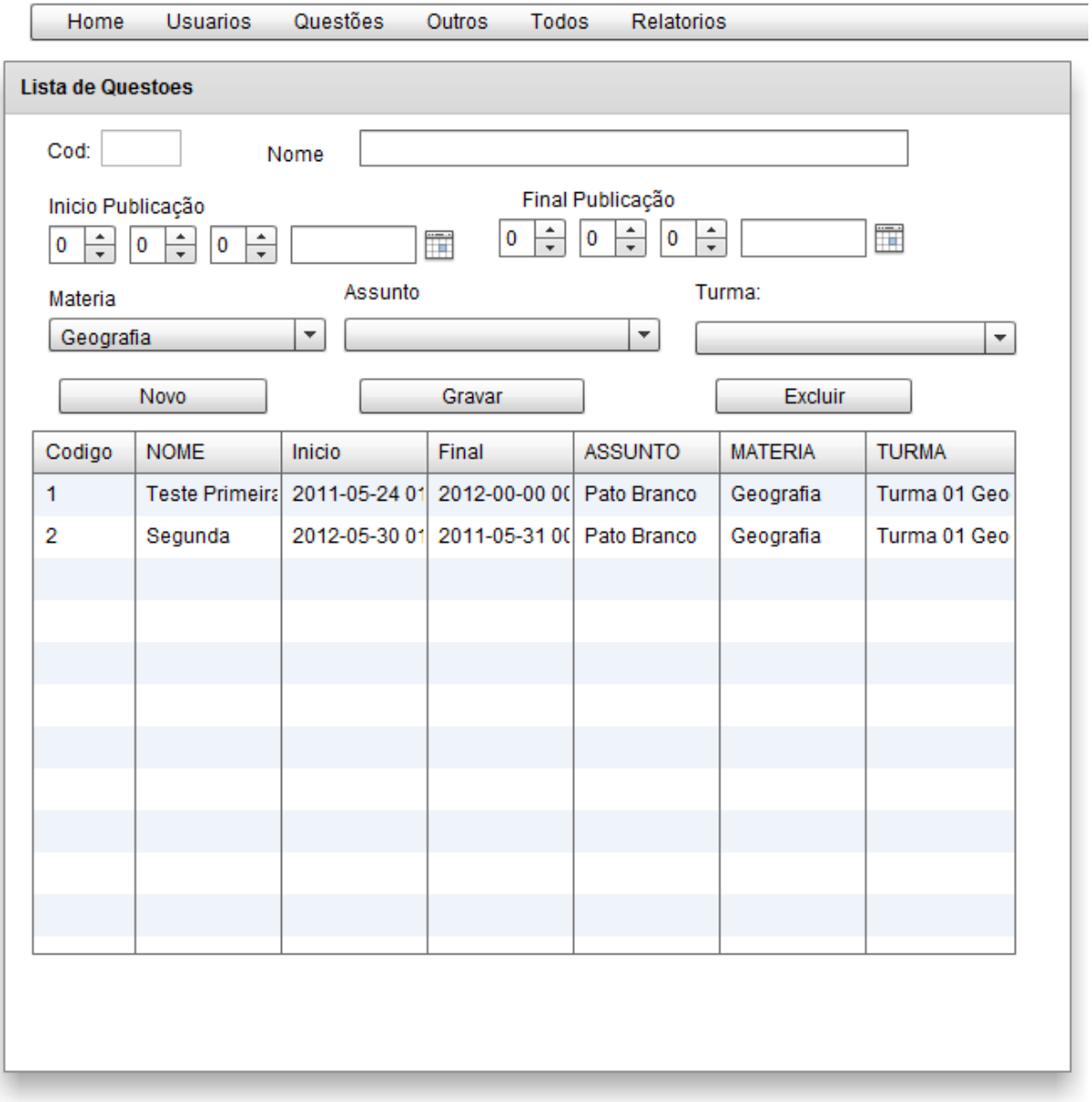
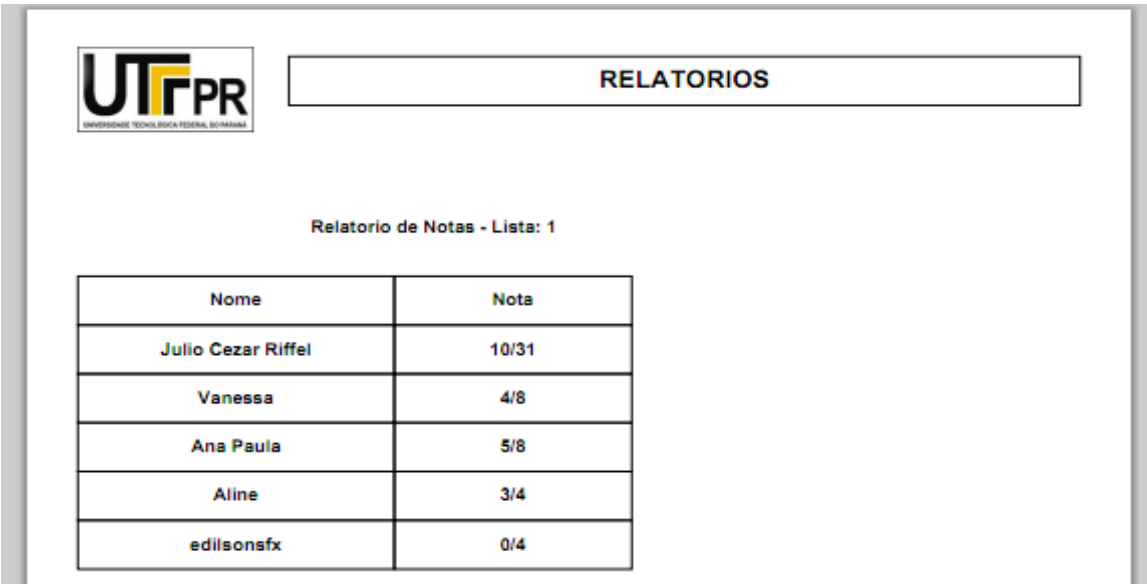


Figura 21 – Tela de listas de questões

O relatório apresenta os resultados parciais caso a lista ainda esteja disponível para os alunos e final, depois de fechar o período disponível para os alunos. O relatório é gerado em PDF (*Portable Document Format)* classificado por lista de questões. Foi optado pelo PDF devido a possibilidade de salvar, imprimir e a compatibilidade com sistemas operacionais. A Figura 22 exemplifica um relatório de notas de uma determinada lista.



RELATORIOS

Relatorio de Notas - Lista: 1

Nome	Nota
Julio Cezar Riffel	10/31
Vanessa	4/8
Ana Paula	5/8
Aline	3/4
edilsonsfx	0/4

Figura 22 – Relatório de notas em PDF

4.4 IMPLEMENTAÇÃO DO SISTEMA

A implementação do sistema iniciou com o desenvolvimento das tabelas do banco de dados MySQL por meio da ferramenta MySQL Workbench. Após a conclusão do banco de dados foi criado um novo projeto no Adobe Flash Builder 4 e em seguida instalado o pacote AMFPHP. A criação do projeto definiu o ambiente inicial de desenvolvimento. A Figura 23 contém uma visualização do desenvolvimento da interface de manutenção de cadastros.

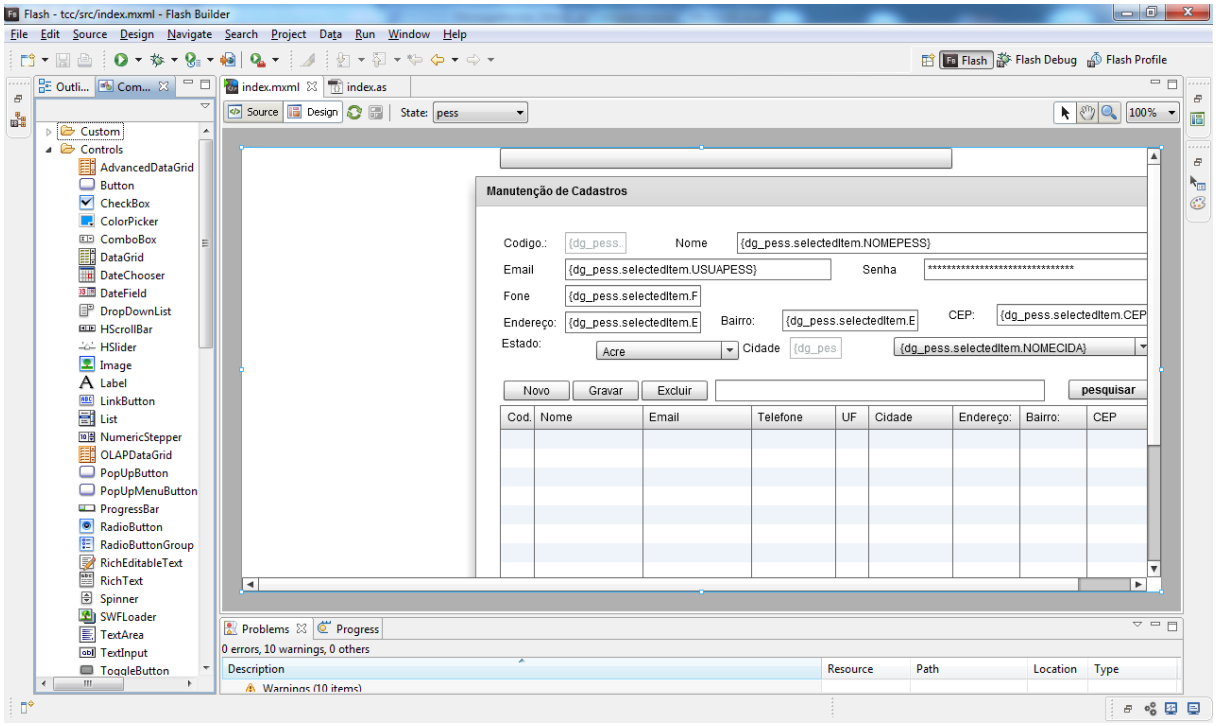


Figura 23 – Desenvolvimento da interface de manutenção de cadastros

Depois da Primeira tela composta, foi alternando entre o desenvolvimento da interface gráfica, a implementação do código fonte de lógica e conexão e a implementação da classe de conexão em PHP.

Para cada tela implementada foi utilizado o seguinte procedimento: finalizar a sua implementação e realizar as correções necessárias antes de seguir para a próxima tela. Desta maneira a interface gráfica e lógica do sistema foram implementadas simultaneamente em ciclos iterativos.

As linguagens de marcação se tornaram um padrão no desenvolvimento de *software* para Internet, e são oferecem muitos recursos para especificar uma interface com o usuário. Baseada em XML, a MXML é composta de várias marcações definindo a aplicação, sendo mais completas que HTML, por prover uma quantidade maior de componentes de programação, como menus e elementos de controle de dados.

Em uma marcação MXML é possível definir os estilos, atributos e eventos do objeto, de maneira que os eventos podem fazer chamadas a funções ActionScript semelhante ao que HTML faz com funções JavaScript.

A seguir serão apresentados trechos principais dos códigos fontes de interface e lógica do sistema desenvolvido.

Namespace (Listagem 1) define tudo o que o aplicativo precisa para acessar os componentes e as classes originais do *framework*. No Flex é definido um *namespace* para cada parte do SDK. Os novos componentes do Flex 4 ficam no *namespace* com prefixo “S”, e os elementos *core* do SDK ficam no prefixo “FX”. Já os componentes do Flex 3 continuaram no pacote de prefixo “mx”.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
3                 xmlns:s="library://ns.adobe.com/flex/spark"
4                 xmlns:mx="library://ns.adobe.com/flex/mx"
5                 creationComplete="init()"
6                 xmlns:local="*"
7                 currentState="index"
8                 height="100%" width="100%"
9                 skinClass="CustomAppSkin">
312
313    //*.....*//
314
315 </s:Application>
316
```

Listagem 1 – Exemplo de definição de namespaces

ActionScript (Listagem 2) é responsável por toda a lógica do sistema. O ActionScript

é utilizado para criar todas as interatividades vistas na aplicação. Nessa linguagem foram implementadas validações como a verificação de tipos de dados em tempo de compilação e em tempo de execução. Essa linguagem permite a implementação efetiva de herança baseada em classes orientadas a objeto.

```
11
12      <fx:Script source="index.as" />
13
14      <fx:Script>
15          <![CDATA[
16              import flash.profiler.showRedrawRegions;
17
18              import mx.controls.Alert;
19              import mx.managers.PopUpManager;
20              import mx.rpc.Fault;
21              import mx.rpc.events.FaultEvent;
22
23              private function resultLogin (event:Object):void    {
24                  if (event.result.login == "OK"){
25                      PopUpManager.removePopUp(this);
26
27                  }else{
28                      Alert.show("Senha/Usuario Invalido");
29                      lb_erro.text="Tente Novamente";
30                  }
31              }
32
33              }
34
35          ]]>
36      </fx:Script>
37
```

Listagem 2 – Exemplo de ActionScript elaborado

O *state* (Listagem 3) define um estado de visão, uma visão particular de telas. No sistema foi definida uma *state* para cada tela. Quando o sistema é carregado é mostrado o *state index* em que está disponível o formulário de *login*. Quando é realizado o *login* com sucesso o *state* é alterado para *home* no qual está disponível o menu principal que da acesso a todos as telas do sistema.

```
41      <s:states>
42          <s:State name="index"/>
43          <s:State name="pess"/>
44          <s:State name="home"/>
45          <s:State name="mate"/>
46          <s:State name="seri"/>
47          <s:State name="bime"/>
48          <s:State name="nive"/>
49          <s:State name="turm"/>
50          <s:State name="ustu"/>
51          <s:State name="assu"/>
52          <s:State name="list"/>
53          <s:State name="perg"/>
54          <s:State name="perg01"/>
55          <s:State name="perg02"/>
56          <s:State name="peli"/>
57          <s:State name="nota"/>
58      </s:states>
59
```

Listagem 3 – Exemplo de states definidos

A codificação RemoteObject utiliza AMF, e permite o acesso a objetos em Java, PHP ou qualquer outra linguagem suportada, através do AMF/SOAP. Porém, os dados são serializados em um formato binário e desserializados assim que retornam para o Flash Player.

Na Listagem 4 está um exemplo de invocação do método AMFPHP. Primeiro, o RemoteObject é declarado como *driver* de destino o AMFPHP. Em seguida é declarado o método que se conectara a função "servidor" na classe PHP do *service* AMFPHP. Todo método de comunicação deve possuir os métodos de manipulação, *result* para tratar a resposta da invocação e o método *fault* caso ocorra alguma falha na comunicação.

```
50
51      <fx:Declarations>
52          <s:RemoteObject id="ro_servidor"
53              source="servidor"
54              destination="amfphp"
55              showBusyCursor="true">
56
57  <!--LOGIN -->
58              <s:method name="login_sessionresult"
59                        result="{resultLogado(event)}"
60                        fault  = "mostrarLogin()"/>
61
62              <s:method name="login_logout"/>
```

Listagem 4 – Exemplo de inovação do método AMFPHP

O sistema apresenta várias telas, a navegação entre elas é feita por meio do menu. A

Listagem 5, corresponde ao código fonte do menu principal do sistema, carregado a partir de um XMLList interno composto no próprio ActionScript pelo comando dataProvider. Esse componente representa um menu, com vários submenus. O menu principal é incluído em todas as telas pelo comando IncludeIn.

```
456
457     <mx:MenuBar labelField="@label" itemClick="menuHandler(event);"
458                 dataProvider="{menuBarCollection}" width="50%"
459                 id="menubar1" x="273" y="0"
460 includeIn="assu,bime,home,list,mate,nive,nota,peli,perg,
461 perg01,perg02,pess,seri,turm,ustu"/>
462
463
```

Listagem 5 – Código fonte do menu principal do sistema

O *panel* é um dos componentes mais utilizados para organização de componentes em uma tela, possui um TitleBar para o titulo e uma área para adicionarmos conteúdo. A Listagem 6 corresponde ao código fonte do *panel* de manutenção de matérias. Neste *panel* há outros componentes como *label*, *textinput*, *button* e *datagrid*. Cada *panel* pode ser incluso em um ou mais *states*.

```
578 <s:Panel includeIn="mate" x="273" y="30" width="585" height="470" creationComplete="initmate()"
579          title="Materias">
580     <s:Label x="36" y="30" text="Cod:"/>
581     <s:TextInput x="77" y="34" id="ti_mate_cod" text="{dg_mate.selectedItem.CODIMATE}"
582                  enabled="false" width="53"/>
583     <s:TextInput x="192" y="33" id="ti_mate_nome" text="{dg_mate.selectedItem.NOMEMATE}" width="354"/>
584     <s:Label x="149" y="28" text="Nome&#xd;"/>
585
586     <mx:DataGrid id="dg_mate" x="36" y="107" dataProvider="{array_mate}" width="510" height="266" >
587                 <mx:columns>
588                         <mx:DataGridColumn headerText="Codigo" dataField="CODIMATE" width="60"/>
589                         <mx:DataGridColumn headerText="Nome" dataField="NOMEMATE"/>
590
591                 </mx:columns>
592         </mx:DataGrid>
593
594         <s:Button x="368" y="69" label="Excluir" click="mate_delete()" width="90"/>
595         <s:Button x="245" y="69" label="Gravar" click="ro_servidor.mate_gravar.send()" width="94"/>
596         <s:Button x="135" y="69" label="Novo" click="novo_mate()" width="91"/>
597 </s:Panel>
```

Listagem 6 – Exemplo de código de um panel

O remoteObject chama a função na classe PHP transmitindo os dados necessários de forma binária. Esse modelo permite enviar e receber objetos entre o PHP e o Adobe Flex. É possível criar um objeto no Flex e enviá-lo diretamente ao PHP, mantendo a sua composição intacta. No sistema desenvolvido, o Adobe Flex acessa, via RemoteObject, um método de uma classe que está localizado no servidor por meio do protocolo AMF.

O AMFPHP é baseado em “Convenção sobre Configuração”, ou seja, para que tudo ocorra corretamente basta colocar os arquivos nos locais certos. E respeitar as convenções de manter a classe na pasta amfphp/service. Além disso, o nome da classe deve ser o mesmo nome do arquivo.

A Listagem 7 apresenta o início da classe principal que realiza a conexão com o banco de dados na função construtora e verifica o usuário e a senha para realizar o *login*. Após a consulta ao banco de dados, essa classe retorna um objeto com as informações do *login* para o cliente.

```php
<?php
session_start();
class principal{

    function principal() {
        $servidor = "localhost";
        $usuario_bd = "utfpr_tcc";
        $senha_bd = "********";
        $banco = "utfpr_tcc";
        $con = mysql_connect($servidor, $usuario_bd, $senha_bd) or die(
            "Erro de conexão com localhost, o seguinte erro ocorreu -> " . mysql_error());
        mysql_select_db($banco) or die("Erro de conexão com banco de dados,
                o seguinte erro ocorreu -> " . mysql_error());
    }

//LOGIN
    function login_fazer($usuario, $senha) {

        $ssql = "SELECT usuapess, senhpess, codipess, nomepess FROM pocapess
        WHERE usuapess='$usuario' && senhpess='$senha'";
        $rs = mysql_query($ssql);

        if ((mysql_num_rows($rs)) == 1) {
            $retorno['login'] = "OK";
            $id_encontrada = mysql_fetch_object($rs);
            $retorno['id'] = $id_encontrada->codipess;
            $retorno['nome'] = $id_encontrada->nomepess;
            $_SESSION['CODIPESS'] = $id_encontrada->codipess;
            $_SESSION['SENHPESS'] = md5($id_encontrada->senhpess);
        } else {
            $retorno['login'] = "falha";
            $retorno['mensagem'] = "Usuario/senha não confere";
        }
        return $retorno;
    }
```

Listagem 7 – Exemplo de classe em PHP definida

4.5 DISCUSSÃO

A discussão aqui apresentada é decorrente do mesmo sistema ter sido implementado em uma versão para *web* e uma versão para ambiente *desktop*. Essa discussão se refere, de certo modo, à comparação entre a implementação do sistema em versão *web* e *desktop*. O que

está colocado nesta seção tem como base o ponto de vista do programador. Assim, são considerados aspectos relacionados à implementação decorrentes de uma análise informal, que tem o objetivo de apenas colocar o ponto de vista de uso de tecnologias.

A versão *desktop* do sistema foi implementada utilizando a linguagem em Java e a versão *web* foi implementada utilizando Adobe Flex e a linguagem PHP.

Há uma grande diferença no método de utilização de banco de dados. Com a linguagem Java é possível conectar diretamente o banco. Enquanto que com Flex isso não é possível. É necessário uma linguagem *backend* para a conexão.

O Flex apresenta a vantagem de possuir mais componentes prontos que a linguagem Java, que podem ser utilizados ou editados conforme a necessidade.

Para facilitar e agilizar o desenvolvimento em flex existem várias bibliotecas *open source* com componentes já desenvolvidos que são muito semelhantes as APIs (*Application Programming Interface)* disponíveis para Java. Ambas são totalmente orientados a objetos, facilitando a manutenção e permitindo reaproveitamento de componentes e código.

A plataforma de execução deve apenas possuir uma JVM (*Java Virtual Machine*) no caso do Java ou Flash Player no caso do Flex. Ambas executam da mesma maneira em qualquer plataforma e permitem desenvolvimento de aplicações *desktop*, *web* e *mobile*.

O Flex é uma linguagem nova, lançada em 2004, e que está conquistando progressivamente espaço entre os programadores. Enquanto que a linguagem Java é uma linguagem mais madura que já conquistou o seu espaço como uma das linguagens mais utilizadas para desenvolvimento. Em termos de futuro, o Flex tem fortes perspectivas de ganhar espaço, pois possui muitos recursos, é fácil de usar e proporciona alta produtividade. Ambas as tecnologias possuem o SDK aberto, produzindo aplicativos compatíveis com grande parte dos usuários, por executar por meio de uma máquina virtual (*flash player*) instalado quase na totalidade dos navegadores, como mostra a Figura 24.

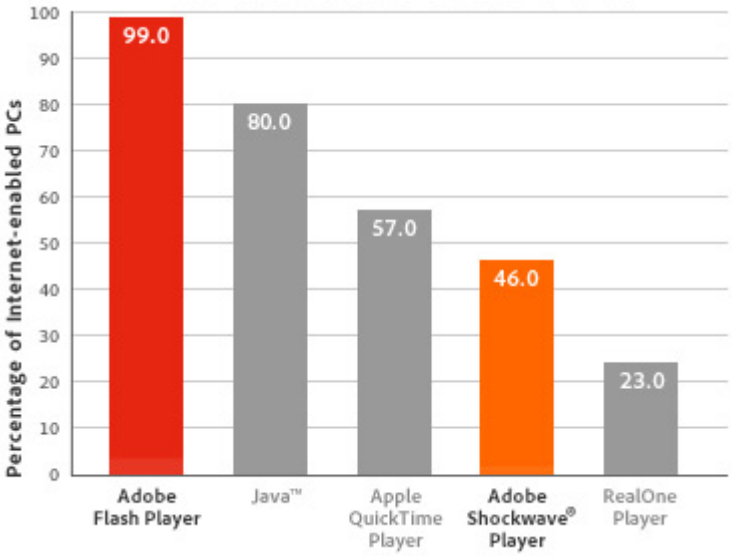


Figura 24 – Difusão das tecnologias

Fonte: http://www.adobe.com/products/player_census/flashplayer/

5 CONCLUSÃO

A utilização da informática como ferramenta de apoio ao aprendizado é uma realidade há tempos e não pode ser desconsiderada. Pode ser utilizada para auxiliar e agilizar o processo de transmissão de conhecimento. Os novos recursos tecnológicos têm despertado interesses pela possibilidade de tornar mais atrativo e dinâmico o processo de ensino e aprendizagem. Nesta perspectiva, a informática ganha atenção, inclusive pela possibilidade de integrar recursos multimídia.

Por meio deste trabalho foi desenvolvido um sistema de avaliação de conteúdos (testes de múltipla escolha) utilizando uma linguagem de programação, conhecida por ActionScript 3, em o ambiente de desenvolvimento Adobe Flash Builder 4. Além de conceitos no desenvolvimento de software *web* com interface ricas, conhecidos por RIAs.

A utilização desses conceitos e tecnologias para desenvolver soluções na *web* se mostrou eficaz. Isso é decorrente da facilidade proporcionada pelo ambiente de desenvolvimento e pela praticidade de implementação que são obtidas, inclusive, pelo uso de orientação a objetos tanto para a interface quanto para a lógica de negócio. Desta forma, o sistema desenvolvido apresenta uma alternativa para implementar aplicativos com interfaces mais intuitivas e similares aos aplicativos *desktop*.

Atualmente o Flex é considerado um dos ambientes mais completos para o desenvolvimento de aplicações hipermídia. Isso se deve pela sua integração com o Flash para publicação de conteúdo interativo com interfaces gráficas avançadas e amplamente compatíveis.

O presente trabalho foi construtivo. Principalmente por possibilitar, por meio de extensa pesquisa, o aprofundamento de conhecimento em conceitos de desenvolvimento adquiridos durante a graduação e de uma nova linguagem que não foi aprendida durante o curso.

Os objetivos de desenvolver um sistema para auxiliar o professor nas avaliações utilizando conceito de RIA com o Flex foram alcançados, considerando as funcionalidades e a finalidade da proposta e permitindo o acompanhamento do desempenho dos alunos. A validação foi feita pelas alunas do curso de Química em situação real.

Este trabalho apresenta apenas as funções básicas de um sistema de avaliação. Assim, existem muitas outras adições que poderiam ser feitas. Como proposta futura, o sistema desenvolvido possui diversas possibilidades de expansão e melhorias, dentre elas:

a) Inclusão de novas funcionalidades e interações (ações e eventos);

b) Melhorias na usabilidade;

c) Implementar um *log* de acessos, controlando todas as atividades;

d) Elaborar estatísticas de acesso individual em cada página;

e) Desenvolver relatórios avançados que permitam identificar dificuldades (lacunas) no aprendizado dos alunos que são usuários do sistema.

REFERÊNCIAS

ADOBE. **Adobe Flex 3**. **Now building on flex**. Disponível em: <http://www.adobe.com/products/flex/>. Acesso em: 12 de maio de 2011.

AMFPHP. **AMPFPHP**. Disponível em http://amfphp.sourceforge.net/. Acesso em 20 de abril de 2011.

ASTAH. **Astah community**. Disponível em <http://astah.change-vision.com/en/product/astah-community.html>. Acesso em 08 de abril de 2011.

CHO, E. S.; KIM, S. D.; RHEW, S. Y.; LEE, S. D.; KIM, C. G. **Object-oriented web application architectures and development strategies**. Asia Pacific Software Engineering Conference and and International Computer Science Conference 1997 (APSEC '97/ICSC '97), 1997, p. 322 – 331.

DUHL J., **Rich internet applications**, IDC white papers, 2003. Disponível em http://www.idc.com. Acesso em 20 de novembro de 2010.

FRATERNALLI, P., PAOLINI, P., **A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications**, In: Extending Database Tecnology (EDBT 98), p. 421-435, 1998.

FUKUDA, H., YAMAMOTO, Y. **A system for supporting development of large scaled Rich Internet Applications**. In: 23rd IEEE/ACM International Conference on Automated Software *Engine*ering (ASE 2008), 2008, p. 459-462.

LOOSLEY, C. **Rich internet applications: design, measurement and management chalenges** 2006. Disponível em: <http://www.keynote.com/docs/whitepapers/>. Acesso em: 5 de abril de 2011.

MILANI, A. **MySQL – guia do programador**. São Paulo: Novatec, 2007.

MULLET, K. **The essence of effective rich internet applications**. Macromedia Whitepapers. 2003.

MYSQL. **MySQL.** Disponível em: <http://www.mysql.com>. Acesso em 29 de março de 2011.

NETCRATF. **Netcraft**. Disponível em http://news.netcraft.com/archives/category/web-server-survey/. Acesso em 14 de maio de 2011.

OLIVEIRA, E., PEREIRA, M. J. V., HENRIQUES, P. R. **Compreensão de aplicações web: o processo e as ferramentas**. 6a. Conferência da Associação Portuguesa de Sistemas de Informação. Bragança, 2005, p.1-14.

PHP. **Página web oficial da linguagem PHP**. Disponível em http://www.php.net/. Acesso em 12 de março de 2011.

PRECIADO, J.C. et al. **Necessity of methodologies to model rich internet applications.** In 2005 Seventh IEEE International Symposium on *Web* Site Evolution (WSE'05), 2005, p. 7-13.

PRESSMAN, R. **Engenharia de software**, 5ª ed., Rio de Janeiro: McGrawHill, 2002.

WORKBENCH, **MySQL workbench**. Disponível em http://wb.mysql.com/. Acesso em 15 de março de 2011.