**Skytech Consultoria e Sistemas Ltda**

**RT 002-01**

**Especificação dos Requisitos do Software**

**Sistema Gestão de Vídeo Aulas – SGV 1.0**

**Autor: Fábio Pereira de Vasconcelos**

**Uberlândia - MG**

**Janeiro de 2013**

**Aprovação**

**Aprovamos o documento de Especificação de Requisitos do projeto SGG 2013-03.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fábio Vasconcelos** |  | **31/01/13** |  |

***Versões revisadas anteriores***

**Não aplicável.**

**Especificação dos Requisitos do Software   
SGV 1.0**

**1 Introdução**

**Objetivos deste documento**

|  |
| --- |
| **O desenvolvedor de software, autor deste projeto, necessita de um sistema para web que seja capaz de gerar templates automáticos para o cadastro de cursos em outros sites de comércio eletrônico por leilão sendo o Mercado Livre o principal.**  **Público-alvo: desenvolvedor do projeto.** |

**Escopo do produto**

***Sistema Gestão de vídeo aulas***

|  |
| --- |
| **SGV (Componente único).** |

***Missão do produto***

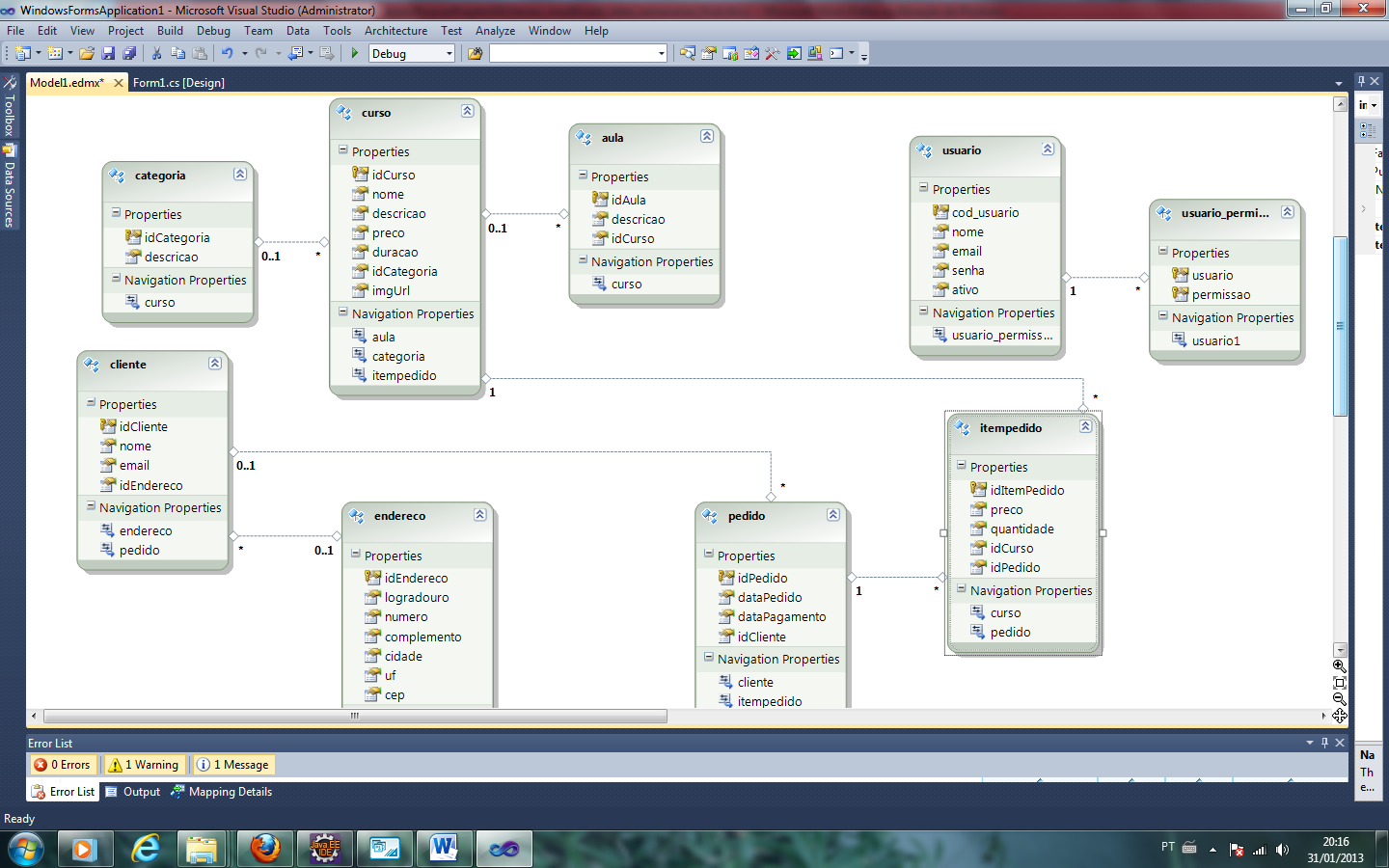
|  |
| --- |
| **Apoio informatizado ao controle de vendas de cursos em vídeo aulas, gerenciamento automático de anúncios para gerar templates de produtos.** |

***Limites do produto***

|  |
| --- |
| **O SGV não fará vendas.**  **O SGV não tem como único objetivo gerar templates e relatórios para o próprio desenvolvedor.** |

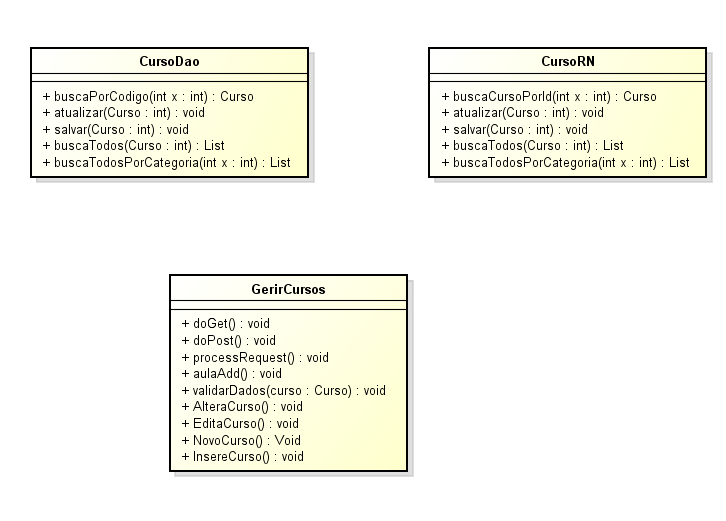
**2 Modelo de domínio**

**2.1 Modelo de domínio**



**Figura 1** – Diagrama de modelo de domínio

**3 Diagrama de classes**



**Figura 2** – Diagrama da arquitetura CRUD cursos

O esquema acima (Figura 2) ilustra a arquitetura do sistema, de um modo simplificado. A classe CursoDao efetua as operações no banco, enquanto o CursoRN apenas faz a passagem dos métodos para a camada de apresentação. Notemos também que a classe GerirCursos é o CRUD da classe Curso. Vários métodos foram criados a fim da melhor estruturação do sistema.

**4 Especificação dos métodos**

Será especificado apenas o CRUD Curso, visto que esta foi a base para criação desse projeto.

**Classe GerirCursos**

|  |  |
| --- | --- |
| Assinatura | doGet(HttpServletRequest, HttpServletResponse response): void; |
| Efeito | Padrão do Servlet para transportar informações via protocolo HTTP via Get |

|  |  |
| --- | --- |
| Assinatura | doPost(HttpServletRequest request,  HttpServletResponse response) : void |
| Efeito | Padrão do Servlet para transportar informações via protocolo HTTP via Get |

|  |  |
| --- | --- |
| Assinatura | processRequest( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Processa as requisições que vem do cliente, pegando os parâmetros de entrada (via get ou post) e repassando para os métodos doGet e doPost. É necessário encaminhar a requisição para o método correto. |

|  |  |
| --- | --- |
| Assinatura | validarDados(Curso): List<String> |
| Efeito | Valida os dados passados por parâmetro para conferir senão é nulo e retornar uma mensagem de erro caso seja. |

|  |  |
| --- | --- |
| Assinatura | AlteraCurso ( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Envia uma lista de categorias para a página AlterarCursos.jsp e o objeto Curso para ser modificado. |

|  |  |
| --- | --- |
| Assinatura | EditaCurso ( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Responsável por fazer a atualização propriamente dita e listar todos os cursos. |

|  |  |
| --- | --- |
| Assinatura | NovoCurso ( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Encaminha para a página InserirCursos.jsp e envia uma lista de categorias para ser exibida no formulário. |

|  |  |
| --- | --- |
| Assinatura | InsereCurso ( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Faz a inserção propriamente dita lista todos os cursos. |

|  |  |
| --- | --- |
| Assinatura | IntanciaCurso (HttpServletRequest request): Curso |
| Efeito | Instancia um objeto curso, pegando os parâmetros da requisição e montando o objeto. É usado para alterar e inserir novos cursos. |

|  |  |
| --- | --- |
| Assinatura | ExcluiCurso ( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Pega o código do curso pela requisição e exclui o curso. |

|  |  |
| --- | --- |
| Assinatura | ListaCursos ( HttpServletRequest request, HttpServletResponse response): void |
| Efeito | Encaminha uma lista de cursos através da requisição. |

**5 Protótipo de telas**



**Figura 3** – Protótipo da tela de exibir detalhes do curso

Para efeito de teste, foi criado uma tela relativa à como seria o sistema adaptado para vendas online. Novamente, não é o objetivo deste projeto, mas pode ser facilmente adaptado para uma necessidade posterior.



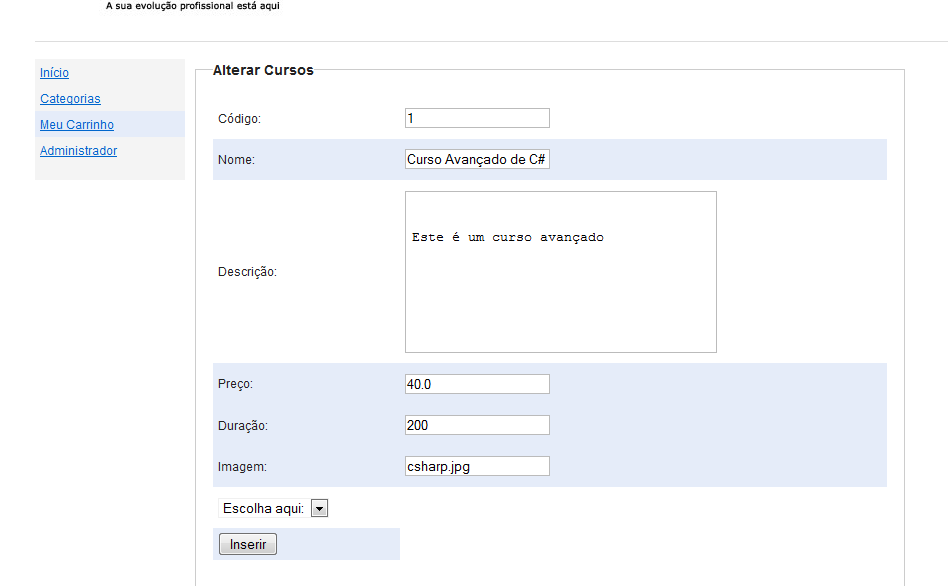
**Figura 4** – Protótipo da tela de listar cursos para o administrador gerir o sistema.

Nesta tela (Figura 4), é exibida a lista de cursos para ser modificado, excluído e adicionar novas aulas (verão posterior). Note que os dados relevantes dos cursos também são listados. A tela contém link para adicionar um novo curso e para exibir relatório por faixa de preço.



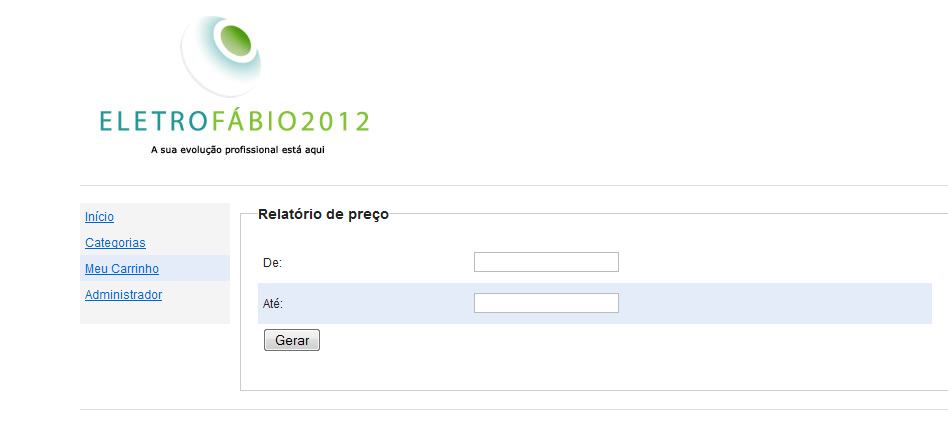
**Figura 5** – Formulário para cadastrar um novo curso

Tela relativa ao formulário de cadastro de curso (Figura 5). Notemos a caixa de combinação com a listagem das categorias.



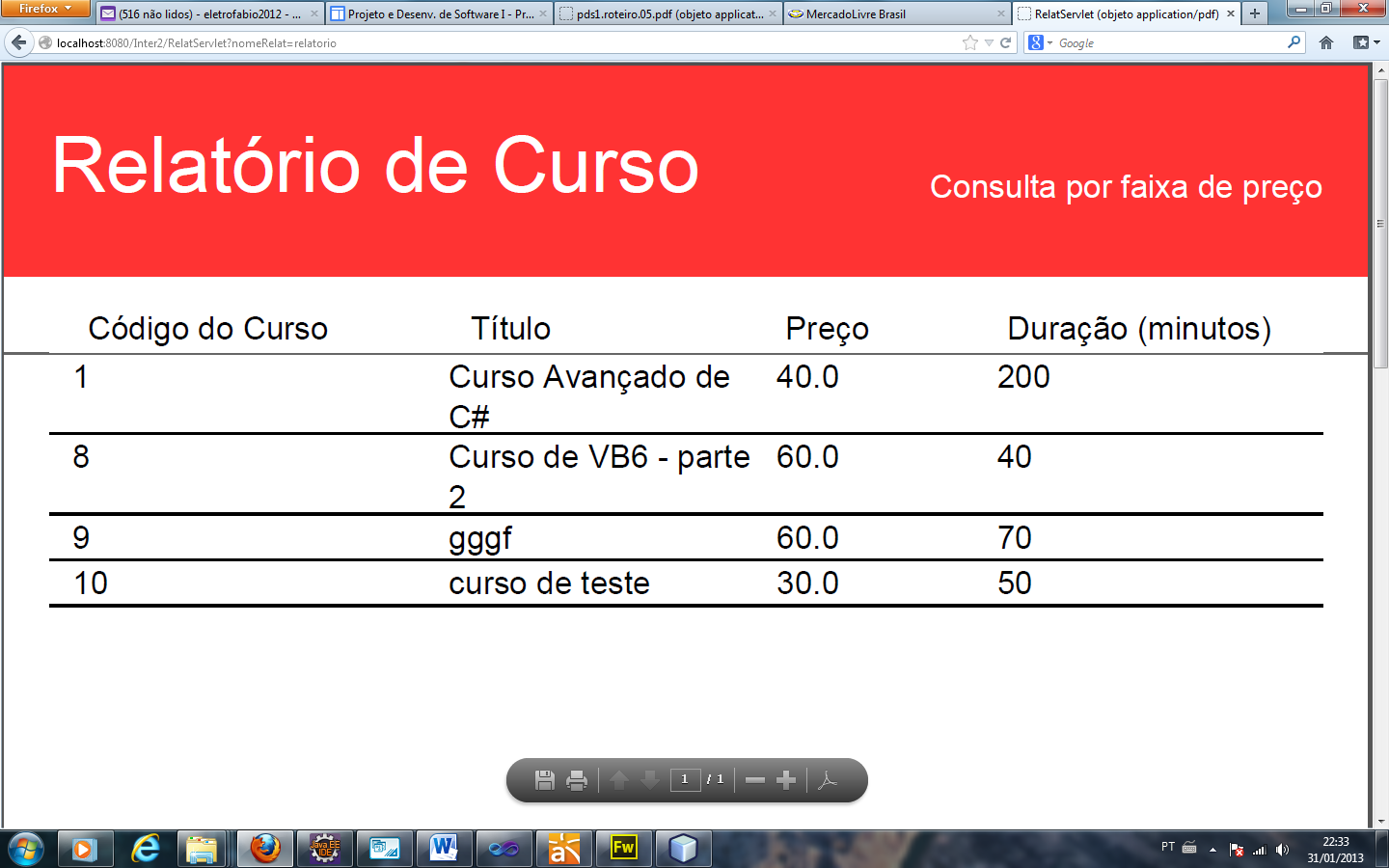
**Figura 6** – Formulário para Alterar um curso

Tela relativa ao formulário para alterar o curso. O sistema é capaz de recuperar os dados do banco para serem exibidos para o cliente (administrador).



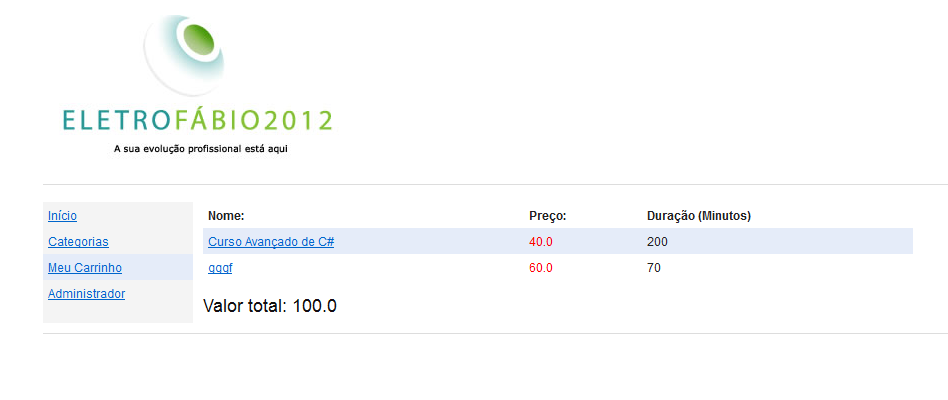
**Figura 7** – Formulário para gerar um relatório entre faixas de preço.

Tela relativa ao formulário para gerar relatório de uma faixa de prelo. O sistema é capaz de criar um arquivo pdf com o relatório.



**Figura 8** – Relatório de consulta de preços por faixa.

Nesta tela (Figura8) é exibido o relatório. Notemos que esse é um arquivo pdf gerado pelo sistema.



**Figura 9**– Tela Meu Carrinho.

Nesta tela (Figura 9) é exibido um protótipo de carrinho de compras. A nível de sistema, esses dados estão em uma sessão e não definitivamente gravadas no banco. Esta operação é necessária para finalizar um pedido. Novamente, esta tela foi criada apenas para fins de testes, podendo ser utilizada posteriormente.

**6 Script do banco de dados**

create database inter;

use inter;

CREATE TABLE endereco (

idEndereco int(11) NOT NULL PRIMARY KEY auto\_increment,

logradouro varchar(45) NOT NULL,

numero int(11) NOT NULL,

complemento varchar(45) DEFAULT NULL,

cidade varchar(45) NOT NULL,

uf char(2) DEFAULT NULL,

cep varchar(30) DEFAULT NULL

);

CREATE TABLE cliente (

idCliente int(11) NOT NULL PRIMARY KEY,

nome varchar(45) NOT NULL,

email varchar(45) DEFAULT NULL,

idEndereco int(11) DEFAULT NULL,

FOREIGN KEY (idEndereco) REFERENCES endereco(idEndereco) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE categoria (

idCategoria int PRIMARY KEY auto\_increment,

descricao varchar(500) DEFAULT NULL

);

CREATE TABLE curso (

idCurso int(11) PRIMARY KEY auto\_increment,

nome varchar(500) DEFAULT NULL,

descricao varchar(45) DEFAULT NULL,

preco double DEFAULT NULL,

duracao int DEFAULT NULL,

idCategoria int,

imgUrl varchar(50) DEFAULT NULL,

foreign key(idCategoria) references categoria(idCategoria)

);

insert into categoria (descricao) value ('Curso de C#');

create table aula(

idAula varchar(20) primary key,

descricao varchar(60),

idCurso int,

foreign key(idCurso) references curso(idCurso)

);

CREATE TABLE pedido (

idPedido int(11) PRIMARY KEY,

dataPedido date DEFAULT NULL,

dataPagamento date DEFAULT NULL,

idCliente int(11) DEFAULT NULL,

FOREIGN KEY (idCliente) REFERENCES cliente(idCliente) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE itempedido (

idItemPedido int(11) NOT NULL PRIMARY KEY,

preco double NOT NULL,

quantidade int(11) NOT NULL,

idCurso int(11) NOT NULL,

idPedido int(11) NOT NULL,

FOREIGN KEY (idCurso) REFERENCES curso(idCurso) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (idPedido) REFERENCES pedido(idPedido) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE usuario (

cod\_usuario INT NOT NULL AUTO\_INCREMENT,

nome VARCHAR(80) NOT NULL,

email VARCHAR(255) NOT NULL,

senha VARCHAR(30) NOT NULL,

ativo TINYINT(1) NOT NULL,

PRIMARY KEY (cod\_usuario),

UNIQUE KEY (email)

);

CREATE TABLE usuario\_permissao (

usuario INT NOT NULL,

permissao VARCHAR(20) NOT NULL,

FOREIGN KEY (usuario)

REFERENCES usuario (cod\_usuario)

ON DELETE CASCADE

ON UPDATE CASCADE

);

LOCK TABLES usuario WRITE;

ALTER TABLE usuario DISABLE KEYS;

INSERT INTO usuario(cod\_usuario, nome, email, senha, ativo) VALUES

(1,'Joao Silva', 'joao@gmail.com', '123', 1);

ALTER TABLE usuario ENABLE KEYS;

UNLOCK TABLES;

INSERT INTO usuario\_permissao(usuario, permissao) VALUES

(1,'ROLE\_OPERADOR');

insert into categoria (descricao) value ('Curso de C#');

insert into curso (nome, descricao, preco, duracao, idCategoria, imgUrl) value ('Curso Avançado de C#', 'Este é um curso avançado', 40, 200, 1, 'csharp.jpg');