# 6 DELIMITAÇÃO DO SISTEMA

## 6.1 Regras de Negócios

Quadro 1 – Regras de negócio do sistema D2VS

|  |  |
| --- | --- |
| ***Código*** | ***Descrição*** |
| **RN01** | Para usar o sistema, o profissional autônomo deve se cadastrar |
| **RN02** | CNPJ não pode ser cadastrado |
| **RN03** | O profissional autônomo deve ter pelo menos um telefone para contato |
| **RN04** | O profissional autônomo precisa ter a idade mínima de 18 anos |
| **RN05** | O profissional autônomo pode avaliar o trabalho de outro profissional autônomo |
| **RN06** | O sistema não gerencia o serviço prestado pelo profissional autônomo |
| **RN07** | O profissional autônomo pode gerar relatório de visualização de perfil e de serviços prestados |
| **RN08** | O profissional autônomo pode recomendar outros profissionais |
| **RN09** | O profissional autônomo deve ter uma foto de perfil |
| **RN10** | O autônomo não pagante não terá acesso ao leilão |
| **RN11** | O autônomo pagante poderá participar de leilões ilimitadamente |
| **RN12** | O autônomo poderá inserir, ilimitadamente, fotos de seus serviços |
| **RN13** | O autônomo poderá adicionar links de vídeos do youtube |
| **RN14** | O contato feito pelo usuário do sistema deve ser respondido em até 2 dias, salvo em feriados e finais de semana |
| **RN15** | O administrador pode inativar usuários do sistema |
| **RN16** | O administrador pode ativar usuários do sistema |
| **RN17** | O cliente só poderá avaliar ou recomendar o autônomo após o término do leilão |
| **RN18** | O administrador do sistema pode cadastrar outros administradores |
| **RN19** | Cliente e autônomo podem abrir uma quantidade ilimitada de leilões |
| **RN20** | O leilão poderá ficar aberto por até 30 dias |
| **RN21** | O leilão deverá ter uma duração mínima de 1 dia |
| **RN22** | Cliente e autônomo que criaram um leilão serão notificados 2h antes do seu término |
| **RN23** | O cliente pode encerrar o leilão mesmo que este não tenha um vencedor |
| **RN24** | O leilão deve conter data de abertura, término, data de previsão de encerramento, título, descrição do serviço |
| **RN25** | O cliente será notificado de novos lances |
| **RN26** | O autônomo poderá fazer pesquisa de leilões. |
| **RN27** | Os lances mais novos do leilão devem ser exibidos no topo da lista dos lances |
| **RN28** | Autônomo não pode fazer lances em áreas de atuação que não sejam pertencentes a ele |
| **RN29** | O carrossel exibirá leilões aleatórios |
| **RN30** | No leilão, os lances mais novos serão exibidos primeiro para o cliente |
| **RN31** | Uma vez criado o leilão, este não poderá ter suas informações editadas |

## 6.2 Requisitos funcionais

TEXTO

Quadro 2 – Requisitos funcionais do sistema D2VS

|  |  |
| --- | --- |
| ***Código*** | ***Descrição*** |
| **RF01** | O sistema deve permitir a inscrição de usuários ao sistema |
| **RF02** | O sistema deve permitir ao usuário atualizar suas informações profissionais e/ou profissional |
| **RF03** | O sistema deve permitir ao autônomo incluir fotos, vídeos |
| **RF04** | O sistema deve permitir acesso ao sistema mediante e-mail e senha |
| **RF05** | Após login, o sistema direcionará o usuário para Index. |
| **RF06** | O sistema deve permitir ao usuário recuperar sua senha por meio do e-mail cadastrado |
| **RF07** | O sistema deve permitir ao usuário encerrar a sua sessão no sistema |
| **RF08** | O sistema permite ao usuário pesquisar trabalhadores autônomos por uma profissão ou especialidade |
| **RF09** | O sistema deve permitir filtrar a pesquisa por autônomos pelo estado e cidade |
| **RF10** | O sistema deve permitir ao usuário ordenar a pesquisa por autônomo mais buscado ou melhor avaliado |
| **RF11** | O sistema deve permitir ao autônomo agendar, atualizar ou excluir um serviço de sua agenda |
| **RF12** | O sistema deve permitir ao autônomo visualizar sua comanda mensal |
| **RF13** | O sistema deve permitir ao autônomo visualizar a quantidade de acessos ao seu perfil num período mensal |
| **RF14** | O sistema deve permitir ao cliente/autônomo criar um leilão de serviço, porém não sendo possível modificá-lo |
| **RF15** | O sistema permite ao cliente/autônomo encerrar um leilão antes do prazo final |
| **RF16** | O sistema permite ao cliente/autônomo filtrar lances por: nome, avaliação ou valor do lance |
| **RF17** | O sistema permite ao autônomo dar lance em leilões no qual ele participa |
|  | O sistema deve permitir |
| **RF18** | O sistema deve permitir a um administrador cadastrado cadastrar outros administradores |
| **RF19** | O sistema deve permitir a listagem de usuários cadastrador ao administrador |
| **RF20** | O sistema deve permitir a ordenação dos usuários cadastrador por tipo de usuário ou status de ativação |
| **RF21** | O sistema deve permite ao administrador ativar ou inativar usuários cadastrados no sistema |
| **RF22** | O sistema deve permitir o usuário cadastrado ou não entrar em contato com os administradores do sistema |
| **RF23** | O sistema deve permitir ao usuário, após o término do leilão, caso este tenha um vencedor, avaliar e recomendar um autônomo |

## 6.3 Modelagem do Sistema

### 6.3.1 Casos de Uso

Identificaram-se, durante a fase de levantamento de requisitos, os três atores que integram o sistema, o ator Administrador, Autônomo e o Cliente. Por meio do diagrama de caso de uso, descreveu-se o que cada ator faz no sistema facilitando a visão dos desenvolvedores. Por se tratar de um diagrama grande, ele foi divido por ator, entretanto foi mantida sua versão completa para que fique claro as funcionalidades dos atores. A figura 1 ilustra o caso de uso completo do sistema.

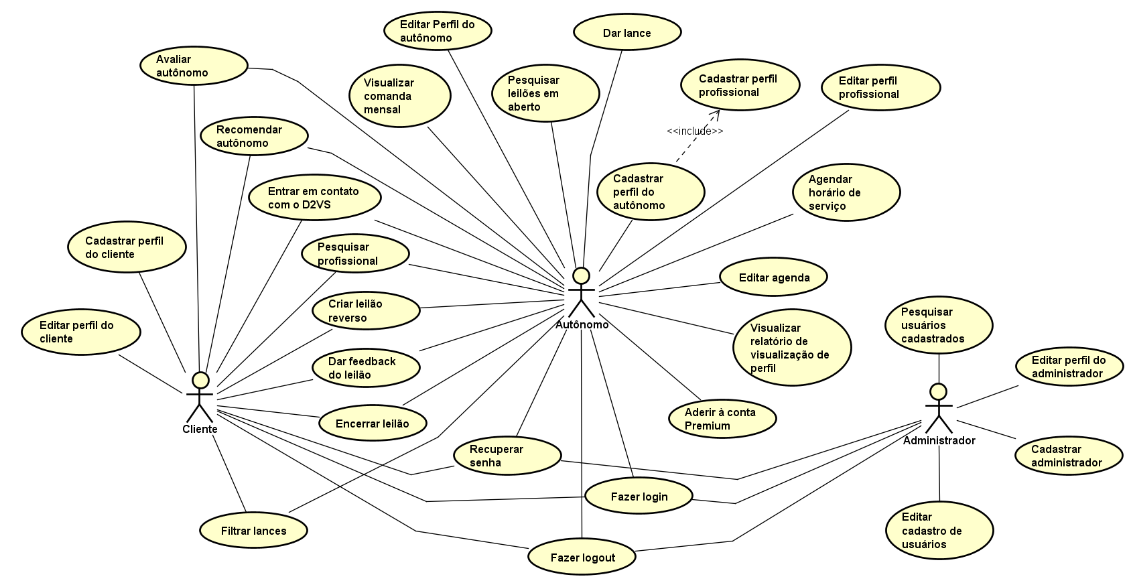


Figura 1 – Diagrama de caso de uso do sistema D2VS

As descrições completas de cada funcionalidade estão anexadas no Apêndice A.

#### 6.3.1.1 Administrador

O Administrador é o ator responsável por cadastrar novos administradores no sistema, assim como gerenciar as contas ativas e inativas. A figura 2 exibe todas as funcionalidades que o administrador pode executar no sistema:

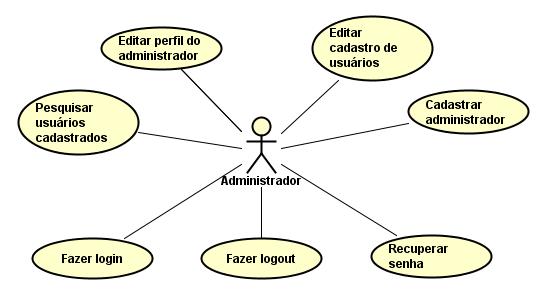


Figura 2 – Caso de uso do ator administrador

#### 6.3.1.2 Cliente

O Cliente é o usuário que acessa o sistema para procurar por trabalhadores autônomos. Além disso, ele pode, por meio do leilão reverso, requisitar um serviço e receber propostas por ele. A figura 3 exibe todas as funcionalidades que o cliente pode executar no sistema.

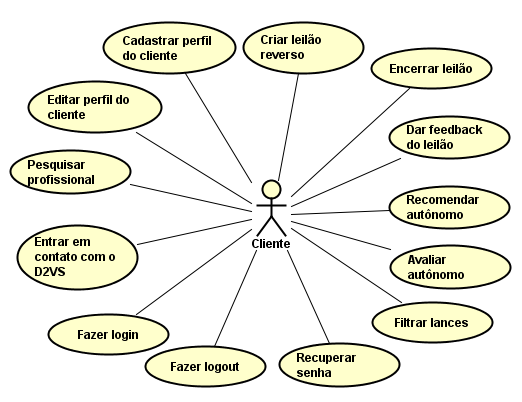


Figura 3 – Caso de uso do ator cliente

#### 6.3.1.3 Autônomo

O Autônomo, ator com o maior número de funcionalidades, é o trabalhador que se cadastra no sistema para divulgar seus serviços e participar de leilões. Além de conter a maior parte das funcionalidades do cliente, no caso as funcionalidades de leilão, há duas ferramentas de auxílio para sua rotina diária, a agenda de serviços e relatório de visualizações de perfil e comanda mensal. A figura 4 exibe todas as que o autônomo pode executar no sistema.

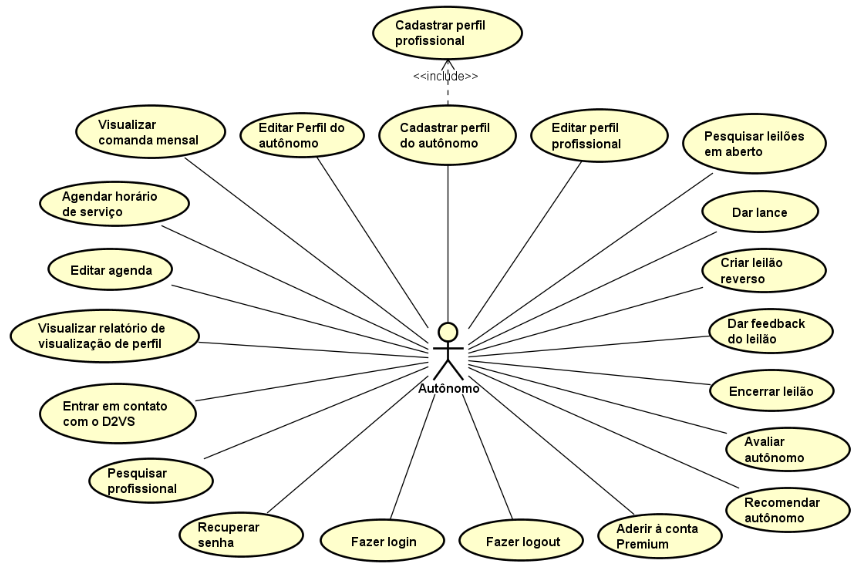


Figura 4 – Caso de uso do ator autônomo

### 6.3.2 Diagrama de Classe

Para definir a estrutura de classes identificadas para o sistema, elaborou-se o diagrama de classes. Nele estão contidos os atributos, métodos e os relacionamento entre as classes que trocam informações. A fim de reduzir o tamanho do diagrama e criar uma versão mais simples, os métodos chamados de Setters e Getters foram nomeados como metodosDeAcesso(). As relações que contém um “\*” representam a relação de 0 ou 1 elemento para vários elementos. Vale ressaltar que todas as figuras serão explicadas seguindo uma ordem de leitura, de cima para baixo e da esquerda para a direita. A figura 5 exibe o diagrama completo do sistema, porém ele será divido em partes e explicado devido ao seu tamanho. Mesmo embora a figura do diagrama completo não seja nítida, ela representa, claramente, todas as relações da classe.

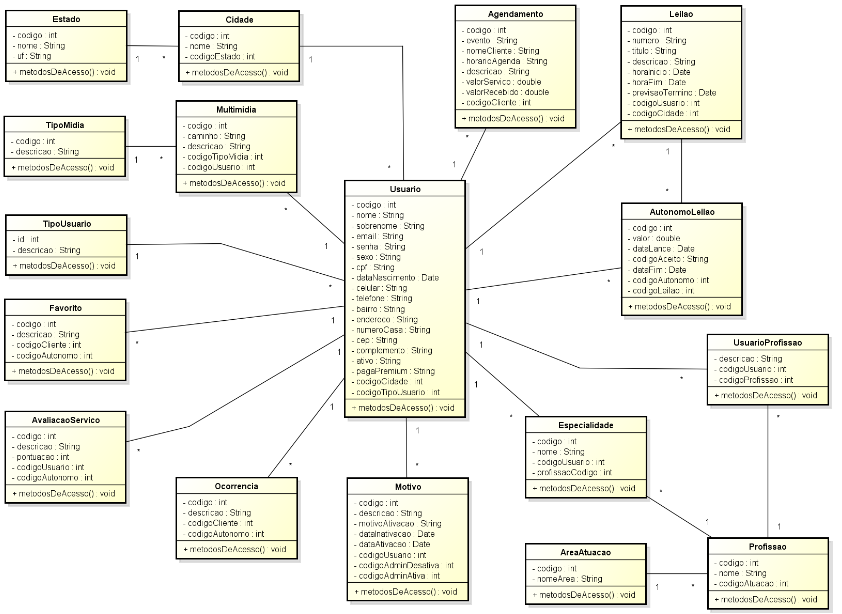


Figura 5 – Diagrama de classes do sistema D2VS

As classes Estado e Cidade representam a localização de um autônomo e também o local onde um leilão ocorre. Já a classe TipoMidia informa se uma mídia é uma foto ou um vídeo e a classe Multimidia informa o caminho no servidor e a legenda de cada mídia. A figura 6 ilustra todos os atributos dessas classes.

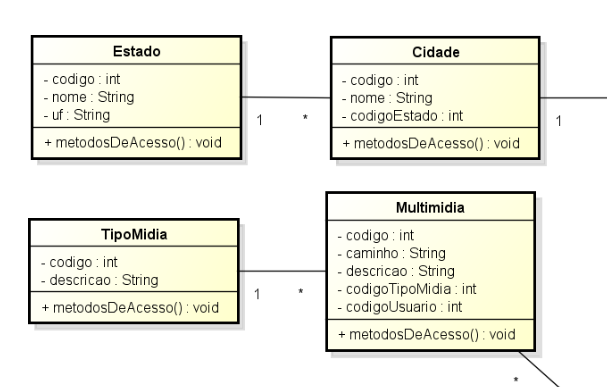


Figura 6 – Classes Estado, Cidade, TipoMidia e Multimidia

A classe TipoUsuario identifica o perfil do usuário no sistema podendo ser: cliente, autônomo ou administrador. A classe Favotiro contém informações para que um cliente possa guardar informações de um autônomo e acessar rapidamente na próxima vez que entrar no sistema. A AvaliacaoServico armazenara a avaliação do autônomo após o leilão. Essa avaliação será exibida em formas de estrelas em sua página de perfil. Por último, a Ocorrencia representa um registro de uma ocorrência feita por um usuário. A figura 7 ilustra todos os atributos dessas classes.

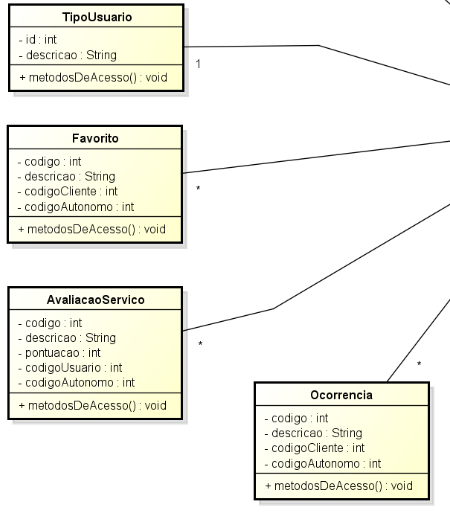


Figura 7 – Classes TipoUsuario, Favorito, AvaliacaoServico e Ocorrencia

A Especialidade representa todas as especialidades de uma profissão cadastrada por um autônomo. A classe Motivo contém as informações sobre o administrador que reativou um usuário do sistema e qual o motivo da reativação.

A classe AreaAtuacao mantém o registro de todas áreas que um autônomo pode estar inserido. Por último, a Profissao mantém todas as profissões cadastradas por um autônomo. A figura 8 ilustra todos os atributos dessas classes.

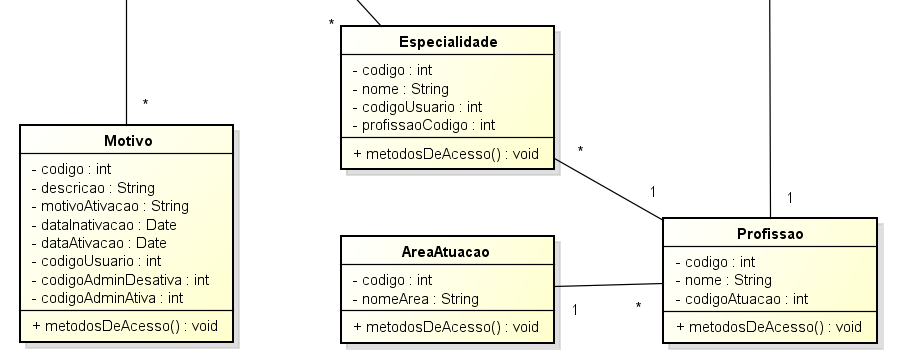


Figura 8 – Classes Especialidade, Motivo, AreaAtuacao e Profissao

A classe Agendamento representa os dados contidos na agenda do autônomo. O Leilao contém as informações necessárias para que o usuário crie um leilão e para que outros visualizem. Já a classe AutonomoLeilao armazena os lances dos autônomos. Por último, a classe UsuarioProfissao descreve cada profissão cadastrada do usuário. A figura 9 ilustra todos os atributos dessas classes.

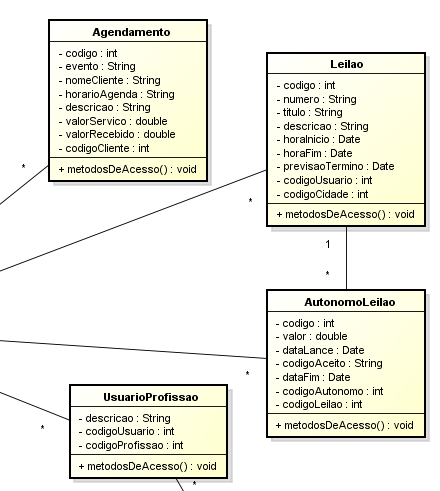


Figura 9 – Classes Agendamento, Leilao, AutonomoLeilao e UsuarioProfissao

Para finalizar a exibição das partes do diagrama de classes, a figura 10 ilustra a classe Usuario que representa todas as informações pertinentes aos atores do sistema, sendo que o autônomo terá de preencher a maior parte das informações.

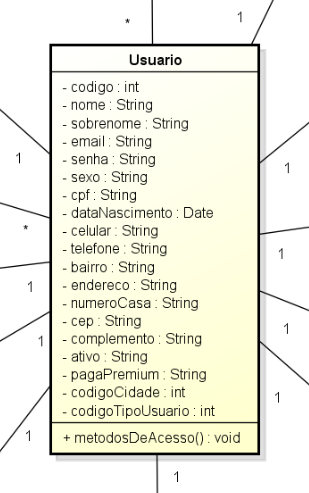


Figura 10 – Classe Usuario

### 6.3.3 Diagrama de Atividades

O diagrama de atividades pode ser usado para representar a modelagem de um processo de negócio assim como a sequência lógica de um único caso de uso. Sendo assim, utilizou-se esse diagrama para representar os principais processos de negócio do sistema D2VS. O primeiro processo é o cadastro de usuário. O usuário deve acessar o sistema e clicar em cadastrar devendo escolher entre cadastro de autônomo ou cliente. Se ele preencher corretamente o formulário, o acesso ao sistema será liberado. A figura 11 ilustra esse processo.

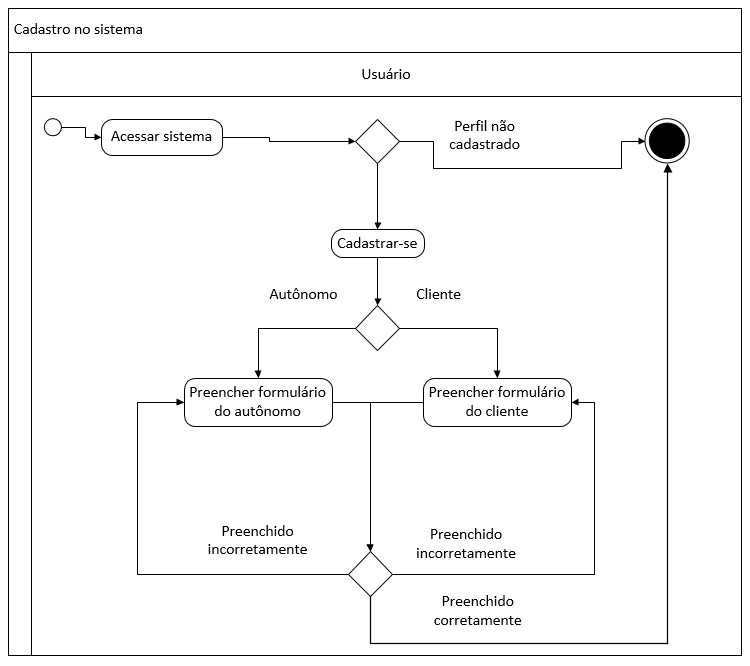


Figura 11 – Processo de cadastro no sistema D2VS

A criação de leilão reverso, principal processo do sistema D2VS, acontece seguindo as seguintes etapas; primeiro, o cliente deve acessar o sistema e realizar o login. Após acesso, deve-se navegar até a área de criação de leilão e clicar em criar leilão. Será solicitado ao usuário um formulário o qual deve ser preenchido corretamente para que o leilão seja exibido aos demais usuários do sistema. Caso tenha sido preenchido incorretamente o usuário deverá preenchê-lo novamente. A figura 12 ilustra o processo completo.

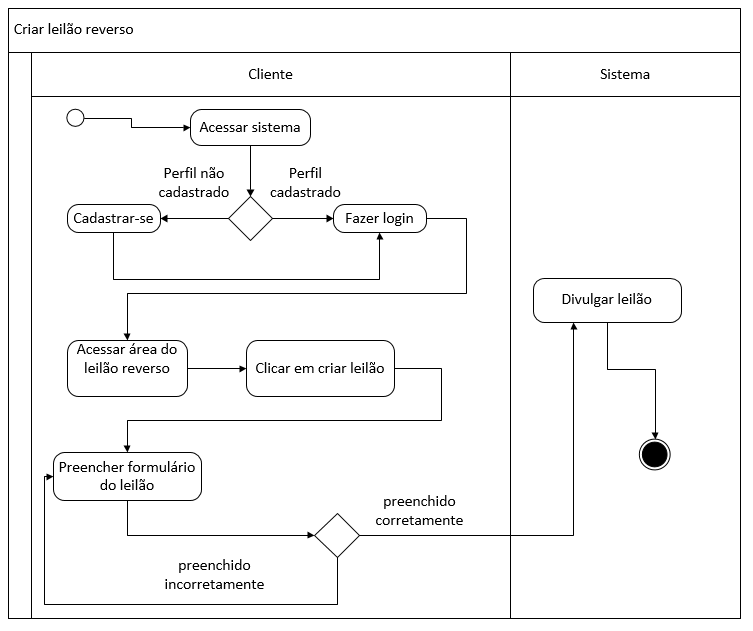


Figura 12 – Processo de criação de leilão no sistema

Outro processo muito importante e que dá continuidade ao leilão reverso é a oferta de lance do autônomo. O trabalhador deverá acessar o sistema, que verificará se os dados são válidos. Após validação, ele deverá navegar até a área do leilão reverso. No campo de busca, deverá digitar o leilão que busca. Caso seja encontrado ele deve acessá-lo, caso contrário, deverá realizar uma nova pesquisa. Juntamente com as informações do leilão, haverá um botão chamado “fazer oferta” que deverá ser clicado e passado um valor para o leilão. A figura XX ilustra as etapas pertinentes a esse processo.

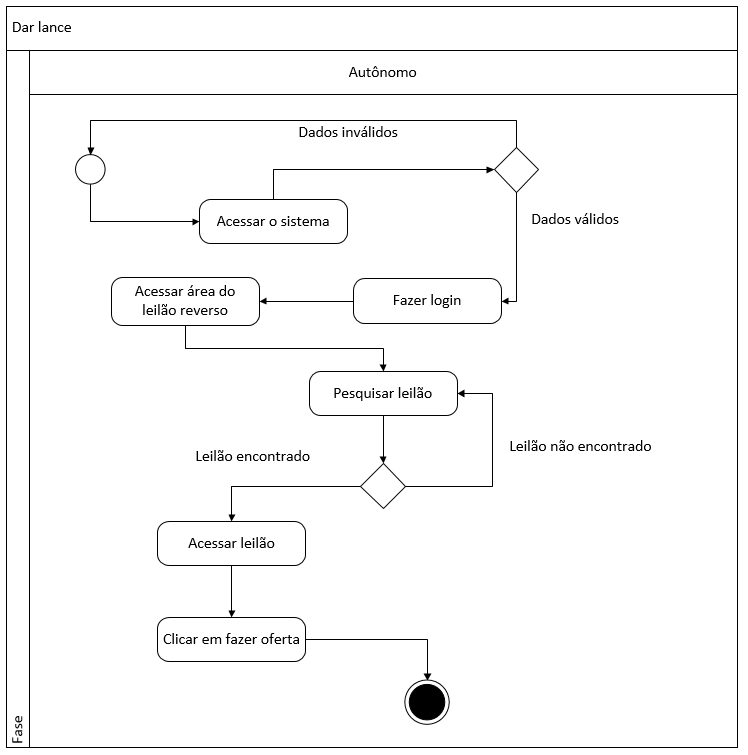


Figura 13 – Processo de fazer oferta por um leilão

## 6.4 Modelagem do Banco de Dados

### 6.4.1 Modelo Entidade e Relacionamento

Para que fosse desenvolvido o script do banco de dados do sistema D2VS, a equipe projetou o MER contendo os dados que devem persistir no sistema e suas relações. Em relação ao diagrama de classes apresentado no capítulo 6.3, subtópico 6.3.2, a única diferença presente é a ausência do metodoDeAcesso(). No modelo apresentado na figura 14 são apresentados, em forma de quadros com um cabeçalho azul, os objetos, seus atributos e seus relacionamento igual ao diagrama de classes. Tendo em vista que os atributos e os relacionamentos já foram explicados, esse MER não será divido.

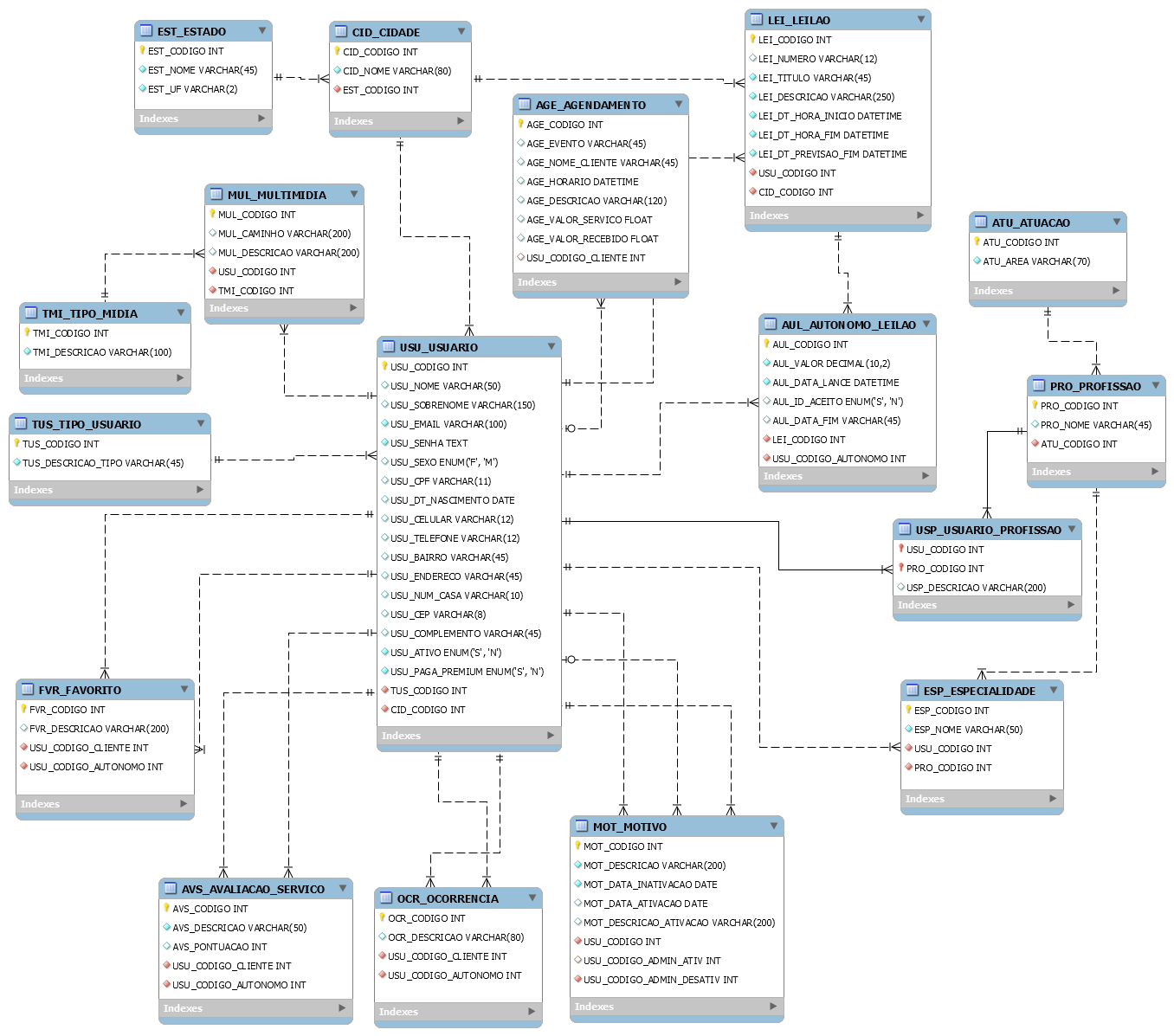


Figura 14 – Modelo Entidade Relacionamento do sistema D2VS

### 6.4.2 Scripts do Banco de Dados

Baseado no MER, gerou-se os comandos necessários para a criação do banco de dados. Para uma melhor visualização dos comandos, as palavras estão escritas em cores diferentes. A cor preta representa o nome do banco de dados, o nome das tabelas e dos campos contidos nela. Em cor azul estão as declarações SQL para criar o banco, as tabelas e definir o tipo de dado dos campos. A cor laranja identifica o tamanho dos campos definidos como varchar, ou seja, um conjunto de strings. Por último, identificado na cor verde, estão os campos do tipo enum, objeto string que aceita valores definidos pelos desenvolvedores. A seguir, serão apresentados os comandos utilizados para criar cada tabela do banco de dados do sistema D2VS.

CREATE DATABASE D2VS DEFAULT CHARACTER SET utf8;

USE D2VS;

CREATE TABLE EST\_ESTADO (

EST\_CODIGO INT PRIMARY KEY,

EST\_NOME VARCHAR(45) NOT NULL,

EST\_UF VARCHAR(2) DEFAULT NULL);

CREATE TABLE CID\_CIDADE(

CID\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

CID\_NOME VARCHAR(80) NOT NULL,

EST\_CODIGO INT NOT NULL,

FOREIGN KEY (EST\_CODIGO) REFERENCES EST\_ESTADO (EST\_CODIGO));

CREATE TABLE TMI\_TIPO\_MIDIA (

TMI\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

TMI\_DESCRICAO VARCHAR(100) NOT NULL);

CREATE TABLE TUS\_TIPO\_USUARIO (

TUS\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

TUS\_DESCRICAO\_TIPO VARCHAR(45) NOT NULL);

CREATE TABLE USU\_USUARIO (

USU\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

USU\_NOME VARCHAR(50) NOT NULL,

USU\_SOBRENOME VARCHAR(150) NULL,

USU\_EMAIL VARCHAR(100) NOT NULL UNIQUE,

USU\_SENHA TEXT NOT NULL,

USU\_SEXO ENUM('F', 'M') NULL,

USU\_CPF VARCHAR(11) NULL UNIQUE,

USU\_DT\_NASCIMENTO DATE NULL,

USU\_CELULAR VARCHAR(12) NULL,

USU\_TELEFONE VARCHAR(12) NULL,

USU\_BAIRRO VARCHAR(45) NULL,

USU\_ENDERECO VARCHAR(45) NULL,

USU\_NUM\_CASA VARCHAR(10) NULL,

USU\_CEP VARCHAR(9) NULL,

USU\_COMPLEMENTO VARCHAR(45) NULL,

USU\_ATIVO ENUM('S', 'N') NOT NULL,

USU\_PAGA\_PREMIUM ENUM('S', 'N') NOT NULL,

CID\_CODIGO INT NOT NULL,

TUS\_CODIGO INT NOT NULL,

FOREIGN KEY (TUS\_CODIGO) REFERENCES TUS\_TIPO\_USUARIO (TUS\_CODIGO),

FOREIGN KEY (CID\_CODIGO) REFERENCES CID\_CIDADE (CID\_CODIGO));

CREATE TABLE MUL\_MULTIMIDIA (

MUL\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

MUL\_CAMINHO VARCHAR(200) NULL,

MUL\_DESCRICAO VARCHAR(200) NULL,

USU\_CODIGO INT NOT NULL,

TMI\_CODIGO INT NOT NULL,

FOREIGN KEY (USU\_CODIGO) REFERENCES USU\_USUARIO (USU\_CODIGO),

FOREIGN KEY (TMI\_CODIGO) REFERENCES TMI\_TIPO\_MIDIA (TMI\_CODIGO));

CREATE TABLE FVR\_FAVORITO (

FVR\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

FVR\_DESCRICAO VARCHAR(200) NULL,

USU\_CODIGO\_CLIENTE INT NOT NULL,

USU\_CODIGO\_AUTONOMO INT NOT NULL,

FOREIGN KEY (USU\_CODIGO\_CLIENTE) REFERENCES USU\_USUARIO (USU\_CODIGO),

FOREIGN KEY (USU\_CODIGO\_AUTONOMO) REFERENCES USU\_USUARIO

USU\_CODIGO));

CREATE TABLE AVS\_AVALIACAO\_SERVICO (

AVS\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

AVS\_DESCRICAO VARCHAR(50) NOT NULL,

AVS\_PONTUACAO INT NULL,

USU\_CODIGO\_CLIENTE INT NOT NULL,

USU\_CODIGO\_AUTONOMO INT NOT NULL,

FOREIGN KEY (USU\_CODIGO\_CLIENTE) REFERENCES USU\_USUARIO (USU\_CODIGO),

FOREIGN KEY (USU\_CODIGO\_AUTONOMO) REFERENCES USU\_USUARIO

(USU\_CODIGO));

CREATE TABLE OCR\_OCORRENCIA (

OCR\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

OCR\_DESCRICAO VARCHAR(80) NULL,

USU\_CODIGO\_CLIENTE INT NOT NULL,

USU\_CODIGO\_AUTONOMO INT NOT NULL,

FOREIGN KEY (USU\_CODIGO\_CLIENTE) REFERENCES USU\_USUARIO (USU\_CODIGO),

FOREIGN KEY (USU\_CODIGO\_AUTONOMO) REFERENCES USU\_USUARIO

(USU\_CODIGO));

CREATE TABLE MOT\_MOTIVO (

MOT\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

MOT\_DESCRICAO VARCHAR(200) NOT NULL,

MOT\_DATA\_INATIVACAO DATE NOT NULL,

MOT\_DATA\_ATIVACAO DATE NULL,

MOT\_DESCRICAO\_ATIVACAO VARCHAR(200) NULL,

USU\_CODIGO INT NOT NULL,

USU\_CODIGO\_ADMIN\_DESATIV INT NOT NULL,

USU\_CODIGO\_ADMIN\_ATIV INT NULL,

FOREIGN KEY (USU\_CODIGO) REFERENCES USU\_USUARIO (USU\_CODIGO),

FOREIGN KEY (USU\_CODIGO\_ADMIN\_DESATIV) REFERENCES USU\_USUARIO

USU\_CODIGO),

FOREIGN KEY (USU\_CODIGO\_ADMIN\_ATIV) REFERENCES USU\_USUARIO

(USU\_CODIGO));

CREATE TABLE ESP\_ESPECIALIDADE (

ESP\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

ESP\_NOME VARCHAR(50) NOT NULL,

USU\_CODIGO INT NOT NULL,

PRO\_CODIGO INT NOT NULL,

FOREIGN KEY (PRO\_CODIGO) REFERENCES PRO\_PROFISSAO (PRO\_CODIGO),

FOREIGN KEY (USU\_CODIGO) REFERENCES USU\_USUARIO (USU\_CODIGO));

CREATE TABLE ATU\_ATUACAO (

ATU\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

ATU\_AREA VARCHAR(70) NOT NULL);

CREATE TABLE PRO\_PROFISSAO (

PRO\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

PRO\_NOME VARCHAR(45) NULL,

ATU\_CODIGO INT NOT NULL,

FOREIGN KEY (ATU\_CODIGO) REFERENCES ATU\_ATUACAO (ATU\_CODIGO));

CREATE TABLE USP\_USUARIO\_PROFISSAO (

PRO\_CODIGO INT NOT NULL,

USU\_CODIGO INT NOT NULL,

USP\_DESCRICAO VARCHAR(200) NULL,

PRIMARY KEY (PRO\_CODIGO, USU\_CODIGO),

FOREIGN KEY (PRO\_CODIGO) REFERENCES PRO\_PROFISSAO (PRO\_CODIGO),

FOREIGN KEY (USU\_CODIGO) REFERENCES USU\_USUARIO (USU\_CODIGO));

CREATE TABLE AGE\_AGENDAMENTO (

AGE\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

AGE\_EVENTO VARCHAR(45) NULL,

AGE\_NOME\_CLIENTE VARCHAR(45) NULL,

AGE\_HORARIO DATETIME NULL,

AGE\_DESCRICAO VARCHAR(120) NULL,

AGE\_VALOR\_SERVICO FLOAT NULL,

AGE\_VALOR\_RECEBIDO FLOAT NULL,

USU\_CODIGO\_CLIENTE INT NULL,

FOREIGN KEY (USU\_CODIGO\_CLIENTE) REFERENCES USU\_USUARIO

USU\_CODIGO));

CREATE TABLE LEI\_LEILAO (

LEI\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

LEI\_NUMERO VARCHAR(12) NOT NULL,

LEI\_TITULO VARCHAR(29) NOT NULL,

LEI\_DESCRICAO VARCHAR(250) NOT NULL,

LEI\_DT\_HORA\_INICIO DATE NOT NULL,

LEI\_DT\_HORA\_FIM DATE NULL,

LEI\_DT\_PREVISAO\_FIM DATE NOT NULL,

USU\_CODIGO INT NOT NULL,

ESP\_CODIGO INT NOT NULL,

CID\_CODIGO INT NOT NULL,

FOREIGN KEY (USU\_CODIGO) REFERENCES USU\_USUARIO (USU\_CODIGO),

FOREIGN KEY (ESP\_CODIGO) REFERENCES ESP\_ESPECIALIDADE (ESP\_CODIGO),

FOREIGN KEY (CID\_CODIGO) REFERENCES CID\_CIDADE (CID\_CODIGO));

CREATE TABLE AUL\_AUTONOMO\_LEILAO (

AUL\_CODIGO INT PRIMARY KEY AUTO\_INCREMENT,

AUL\_VALOR DECIMAL(10,2) NOT NULL,

AUL\_DATA\_LANCE DATETIME NOT NULL,

AUL\_DATA\_FIM DATETIME NOT NULL,

AUL\_ID\_ACEITO ENUM('S', 'N') NULL,

USU\_CODIGO\_AUTONOMO INT NOT NULL,

LEI\_CODIGO INT NOT NULL,

FOREIGN KEY (USU\_CODIGO\_AUTONOMO) REFERENCES USU\_USUARIO

(USU\_CODIGO),

FOREIGN KEY (LEI\_CODIGO) REFERENCES LEI\_LEILAO (LEI\_CODIGO));

# 7 INTERAÇÃO HUMANO COMPUTADOR

## 7.1 Aplicação de práticas de acessibilidade no sistema

A fim de garantir que essas pessoas tenham acesso a informação e promover o desenvolvimento da sociedade, adotaram-se boas práticas de acessibilidade na web promovidas pela W3C. A W3C, consórcio de empresas governamentais e não governamentais, que tem a finalidade de desenvolver padrões para web, desenvolve diretrizes, recursos e estratégias para ajudar pessoas com deficiência[[1]](#footnote-1).

Baseado numa lista desses recursos promovido pelo consórcio[[2]](#footnote-2), utilizou-se 5 aplicações práticas para promover acessibilidade no sistema D2VS. Essas práticas são apresentadas logo abaixo, em formato de quadro, constando o número da prática e o nome na primeira coluna, uma breve definição na segunda coluna e como ela foi utilizada no sistema na última coluna.

O quadro 3 apresenta a primeira prática que beneficia cegos e deficientes auditivos, pois provê alternativas textuais para contextos não textuais, por exemplo, imagens.

Quadro 3 - Alternativa textual para conteúdo não textual

|  |  |  |
| --- | --- | --- |
| **Nome da prática** | **Definição** | **Utilização** |
| 1.1.1 Conteúdo não textual | Fazer com que as informações transmitidas através de conteúdo não-textual sejam acessíveis através de uma alternativa de texto. | Uso do atributo html "alt" em imagens. |

A segunda prática, apresentada no quadro 4, tem por objetivo promover uma visualização agradável do conteúdo textual para pessoas com pouca visão e pessoas idosas.

Quadro 4 - Taxa de contraste apropriada para conteúdos textuais

|  |  |  |
| --- | --- | --- |
| **Nome da prática** | **Definição** | **Utilização** |
| 1.4.3 Contraste (mínimo) | Prover contraste entre cor da frente e cor de fundo para ser lido por idosos e pessoas com baixa visão. | Apresentação visual do texto com taxa de contraste de luminosidade de 4.5:1 |

Beneficiando a todos os usuários, o quadro 5 traz informações sobre o uso de título nas páginas para que os usuários possam se orientar no sistema.

Quadro 5 - Título na página para facilitar a atual localização no sistema.

|  |  |  |
| --- | --- | --- |
| **Nome da prática** | **Definição** | **Utilização** |
| 2.4.2 Título na página | Orientar os usuários sobre sua localização no sistema. | Inserir o título da página na tag html “<title>” |

O uso de rótulos ao lado ou acima de campos de preenchimento em formulários previne erros e otimiza o uso do tempo dos usuários. O quadro 6 refere-se ao uso de rótulos como prática de acessibilidade.

Quadro 6 - Uso de rótulos em formulários

|  |  |  |
| --- | --- | --- |
| **Nome da prática** | **Definição** | **Utilização** |
| 3.3.2 Instruções e rótulos | Identificar tipos de formulários para que os usuários saibam o tipo de dado a ser inserido. | Formulário indicado por rótulos acima de seus campos de preenchimento |

Por último, sem necessidade de recorrer a lentes de aumento, pessoas com pouca visão utilizam pistas visuais para localizar conteúdo rapidamente. A rápida localização de conteúdo ocorre devido ao uso padronizado de elementos que se repetem por um conjunto de páginas do sistema. O quadro 7 traz a definição e a utilização dessa prática.

Quadro 7 - Navegação consistente para rápida interação com conteúdo

|  |  |  |
| --- | --- | --- |
| **Nome da prática** | **Definição** | **Utilização** |
| 3.2.3 Navegação consistente | Uso consistente de elementos os quais o usuário interage repetidamente num conjunto de páginas. | Padronização da posição de elementos nas páginas do sistema. |

## 7.2 Tecnologia assistiva e reconhecimento de Voz

De acordo com o CAT, Comitê de Ajudas Técnicas (2009, p.9), Tecnologia Assistiva é uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social.

Um exemplo de tecnologia assistiva são os reconhecedores de voz, software que recebe e interpreta a voz de uma pessoa e o transforma em texto. Visto que o D2VS é um sistema de busca, optou-se por usar essa tecnologia. A intenção é facilitar a pesquisa de profissionais autônomos para que estes não tenham dificuldade de acesso a eles. A aplicação da tecnologia foi feita através de uma API[[3]](#footnote-3) JavaScript. A API testa se há suporte por parte do navegador do usuário. Além disso, a privacidade do usuário é garantida, uma vez que este deverá permitir acesso ao microfone.

## 7.3 Paleta de Cores

Para a construção visual do sistema, elaborou-se uma cartela com 6 cores que serão aplicadas do sistema, assim como seu código em hexadecimal. A cor principal irá compor o cabeçalho e o rodapé das páginas do sistema e algumas vezes será usado como hover, efeito que muda a cor de um elemento HTML. A cor secundária servirá como contraste da cor principal para dar destaque aos elementos do cabeçalho, por exemplo, o logotipo e botões. A cor base é o background do sistema e será um leve tom de cinza. O texto terá a cor preta e será usado para descrição do serviço do autônomo assim como em outros momentos de escrita. O alerta representa a cor de mensagens de erro que os usuários possam ter durante a utilização do sistema. Por último, o sucesso representa as mensagens que confirmam que determinada etapa foi concluída sem problemas. A figura 15 representa a paleta de cores aplicada no sistema.

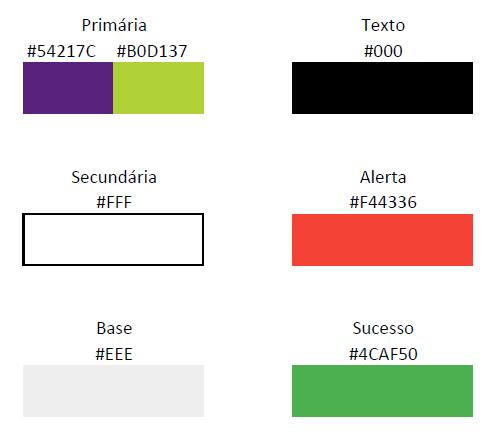


Figura 15 - Paleta de cores aplicada no sistema

## 7.4 Logotipo

O logotipo simboliza um acordo entre um cliente e um autônomo por meio de um aperto de mão. A cor roxa estimula a imaginação, altos ideais e a inspira a fé, comportamento e características essenciais para quem busca prestar o melhor serviço possível. A figura 16 a ilustra



Figura 16 – Logotipo do sistema D2VS

## 7.5 Cartela de ícones

Para a construção do sistema, identificou-se a utilização de 24 ícones. Ter-se-ão 5 ícones de serviços para representar as áreas de atuação que o autônomo pode ser inserido, são elas: serviços de gastronomia, serviços de tecnologia da informação, serviços domésticos, serviços de construção e serviços de educação. Têm-se também ícones para que os usuários compartilhem o sistema nas redes sociais, são eles: facebook, twitter e google plus. Na área do autônomo, estão presentes ícones que representam funcionalidades específicas desse ator: serviços do autônomo, relatório de serviços, calendário de serviços e editar perfil. Na área de cadastro de serviço estarão presentes os ícones: área de imagem, que representa um espaço de adicionar imagens de serviços, link para vídeo e adicionar, que também estará no calendário de serviços. A estrela de avaliação representa a avaliação do autônomo no sistema e poderá ser visualizada por qualquer usuário. Para o administrador, o único ícone específico será cadastrar administrador. Comum para todos os atores, estarão os ícones perfil de usuário, ícone de pesquisa, menu de navegação, login e logout, que dependerá do estado do usuário no sistema. O ícone comanda servirá para exibir ao autônomo os seus ganhos com serviços. O leilão reverso terá um ícone de martelo. Os botões de confirmação terão um sinal de confirmação, os botões de cancelar, um “X”. O autônomo terá um ícone de telefone em sua página para que seus clientes vejam seus contatos. Por último, as opções do leilão terão ícones em forma de caixa de diálogo. A fim de exibir os ícones e suas definições, elaborou-se uma cartela deles exibidas na figura 17.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Serviço de gastronomia | Serviço de Tecnologia da informação | Serviços domésticos | Serviços de construção | Confirmar |
| Serviço de educação | Perfil do usuário | Ícone de pesquisa | Menu de navegação | Cancelar |
| Área de imagem | Link para vídeo | Estrela de avaliação | Adicionar | Contato |
| Login | Logout | Serviços do autônomo | Editar perfil | Criar leilão |
| Relatório de serviço | Calendário de serviços | Cadastrar administrador | http://simpleicon.com/wp-content/uploads/facebook-2-128x128.png  Facebook | Leilão em andamento |
| http://simpleicon.com/wp-content/uploads/twitter-2-128x128.png  Twitter | https://cdn3.iconfinder.com/data/icons/picons-social/57/80-google-plus-512.png  Google plus | Comanda | Leilão reverso | Histórico de leilão |

Figura 17 - Cartela de ícones do sistema D2VS

## 7.6 Tipologia

Escolheu-se a fonte calibri para o desenvolvimento do sistema. Por não ter serifas, pequenos prolongamentos de linha nas extremidades das letras, facilita a legibilidade de textos online tanto em computadores desktop como em dispositivos móveis, reforça a confiança e a credibilidade dos leitores em relação ao conteúdo, valorizando o sistema.

# 8 IMPLEMENTAÇÃO

## 8.1 Linguagens de Desenvolvimento

Para que o sistema D2VS fosse desenvolvido, utilizaram-se alguns padrões de desenvolvimento e linguagens de desenvolvimento sendo elas: C#, HMTL5, CSS3, JavaScript e MySQL. A seguir, será apresentada uma breve descrição desses padrões e linguagens.

### 8.1.1 SQL

Antes de falar sobre SQL, é necessário definir o que é um banco de dados. Banco de dados pode ser definido como “um local no qual é possível armazenar informações para consulta ou utilização, quando necessário” (CARVALHO, 2015, p.3). Essas informações são armazenadas em tabelas com colunas e linhas. As tabelas representam um objeto do qual se guarda dados, as colunas identificam o tipo de dado que é guardado e as linhas representam as informações sobre cada registro. Cada banco é um conjunto de tabelas relacionadas.

Sendo assim, para que um banco de dados exista, é necessária uma linguagem para definir, criar e manipular os dados. Está linguagem chamada de *Structured Query Language* (SQL) “é a linguagem padrão utilizada pelos bancos de dados relacionais” (CARVALHO, 2015, p.5).

### 8.1.2 C#

Por se tratar de uma linguagem simples e poderá, toda a parte do desenvolvimento das funcionalidades foi feita com a linguagem C# (pronuncia-se C sharp). Nas palavras de LIMA (2002), é uma linguagem poderosa como o C++ e simples como o Visual Basic. Além disso, ela é completamente orientada a objetos e fortemente tipada o que ajuda evitar erros de manipulação de tipos e atribuições incorretas.

### 8.1.3 HTML5

É comum nos dias de hoje que o projeto de desenvolvimento de aplicações locais ou remotas tenham desenvolvedores trabalhando na parte *front-end* do projeto. Mazza (2012, p.2) define o termo *front-end* como “um dos termos usados para se referenciar a interface de uma aplicação”. E para o desenvolvimento *front-end,* utilizou-se o *HTML*. HyperText Markup Languagem (HTML), conhecida como Linguagem de Marcação de Hipertexto, é uma linguagem baseada em marcas em que palavras-chave indicam o início e o fim de cada elemento hipertexto (BISPO et al., 2012). Esta é apropriada para o projeto de interfaces e, portanto, sua versão mais nova foi utilizada para o desenvolvimento do sistema.

### 8.1.4 CSS3

Para dar uma aparência refinada e moderna ao sistema, utilizou-se o padrão CSS. Folhas de Estilo em Cascata são um mecanismo utilizado para se controlar a aparência das páginas como a cor, posição dos elementos, efeitos de negrito, tamanho e estilo da letra (BISPO et al, 2012).

### 8.1.5 JavaScript

JavaScript é uma linguagem interpretada pelo navegador, isso significa que um script é executado quando acessado por ele (AIRES; RIBEIRO; ORLOVSKI 2013). Essa linguagem traz diversos recursos para tornar as páginas mais interativas como, por exemplo, mensagens de alerta e caixas de interação. Ainda, algumas de suas funcionalidades são: adicionar efeitos visuais baseadas em condições, busca de informações sem a necessidade de um recarregamento da página e modificação de conteúdo textual. Além de trazer diversos recursos, o uso dela acrescenta usabilidade e acessibilidade para que o usuário tenha a melhor experiência possível.

## 8.2 Padrão de Desenvolvimento

### 8.2.1 Web Forms

Tendo em vista que o presente produto é um sistema Web, decidiu-se por utilizar o Web Forms. Web Forms é uma página baseada na tecnologia ASP.NET para criar a interface do usuário para aplicações Web. Ele apresenta informações para o usuário em qualquer navegador e implementa a lógica do aplicativo usando o código no lado do servidor (server-side). Um arquivo Web Forms pode conter HTML estático e também controles de servidor ASP.NET facilitando o desenvolvimento. A lógica para a página reside num arquivo chamado de code-behind podendo ser escrita em Visual Basic ou em C#.

## 8.3 Ferramentas Utilizadas

Todas as ferramentas mencionadas nos subtópicos do presente capítulo foram utilizadas no Sistema Operacional (S.O) Microsoft Windows. Embora a versão do S.O instalada no computador dos desenvolvedores fossem diferentes, havia um versão compatível para todos sem que o projeto fosse prejudicado.

### 8.3.1 Pencil Project

Foram feitos *wireframes*, protótipo inicial das telas sem uso de cores, das telas do sistema D2VS para que os desenvolvedores tivessem uma base de como fazê-las. Para isso, utilizou-se a ferramenta a Pencil Project. Além de ser é gratuita, essa ferramenta de prototipação possui diversos recursos como uma coleção de formas prontas para desenvolvimento de aplicativos móveis. Ainda, o resultado do trabalho pode ser exportado para um arquivo de imagem .png ou mesmo como uma página *WEB.*

### 8.3.2 MySQL Workbench

O MySQL Workbench é uma ferramenta de gerenciamento de bancos de dados (SGBD) usada por desenvolvedores. Ele permite aos desenvolvedores criar um modelo visual (MER) para que, antes de executar um script, todos os objetos e relacionamentos do banco pudessem ser criados. O MySQL Workbench fornece um ambiente para configurar servidores, administrar usuários e realizar backup de bancos de dados.

### 8.3.3 Visual Studio 2013

Microsoft Visual Studio 2013 é um Ambiente de Desenvolvimento Integrado (IDE) para o desenvolvimento de aplicativos para o sistema operacional Windows e também aplicações Web. Ele inclui um editor de código com suporte ao IntelliSense, técnica que completa parte do código digitado pelo desenvolver otimizando o tempo de desenvolvimento. Além disso, fornece também um depurador, ferramenta que ajuda os desenvolvedores a encontrar problemas com o código.

### 8.3.4 Astah Community

Durante toda a fase de levantamento de requisitos, utilizou-se a Linguagem de Modelagem Unificada (UML). Nas palavras de Guedes (2011), UML é uma linguagem visual para modelar softwares auxiliando os engenheiros de software a definirem as características do sistema.

A equipe precisou criar diagrama de classes e de caso de uso e para criá-los, decidiu-se optar pelo software Astah Community. Astah Community é um software gratuito de modelagem UML, criada por uma empresa Japonesa, muito útil no desenvolvimento de sistemas.

### 8.3.5 Trello

Todo o projeto foi gerenciado por meio do Trello, ferramenta on-line de colaboração que organiza o projeto em quadro. O quadro possui quatro divisórias indicando em qual etapa está uma determinada tarefa, representada por um cartão semelhante as notas adesivas *posts-its*. A primeira divisória, chamada Backlog, contém todas as tarefas do projeto. A segunda divisória, *Sprint* X, em que X representa o número da *sprint,* estão os cartões que serão desenvolvidos pela equipe num determinado período de, geralmente, de 20 a 30 dias. A terceira divisória contém os cartões que estão de fato sendo desenvolvido e qual membro da equipe está trabalhando nele. Por último, a divisória Feito contém todos os cartões que a equipe desenvolveu.

### 8.3.6 Git e SourceTree

SourceTree é uma Interface gráfica do usuário (GUI) para o Git, sistema de controle de versão de arquivos. O Git permite controlar a versão do código fonte do sistema bem como todos os documentos. Esse sistema permite que cada desenvolvedor trabalhe numa parte do projeto e, posteriormente, integrá-las. Cada alteração de código é submetida com uma mensagem sucinta sobre o que foi feito. Deste modo, todos os desenvolvedores podem acompanhar o progresso geral do projeto. Além disso, se uma submissão recente conter erros, facilmente ela pode ser revertida. Por ter uma interface amigável e permitir permite realizar todas essas operações com poucos cliques.

## 8.4 Prototipação e Desenvolvimento

### 8.4.1 Protótipo

Neste tópico, serão mostradas as telas de protótipo que foram desenvolvidas para que pudesse ser desenvolvido as telas reais do sistema D2VS. A figura 18 representa a Index, página inicial do sistema, em que um usuário pode pesquisar por um autônomo ou leilão utilizando a barra de pesquisa na parte superior da tela. Ao lado da barra de pesquisa, o usuário tem a opção de se cadastrar ou entrar no sistema, caso já seja cadastrado. Abaixo da barra de pesquisa, há um carrossel, espécie de slide automático, que exibe leilões em destaque. Por último, abaixo do carrossel, há uma lista de autônomos em destaque por área de atuação.

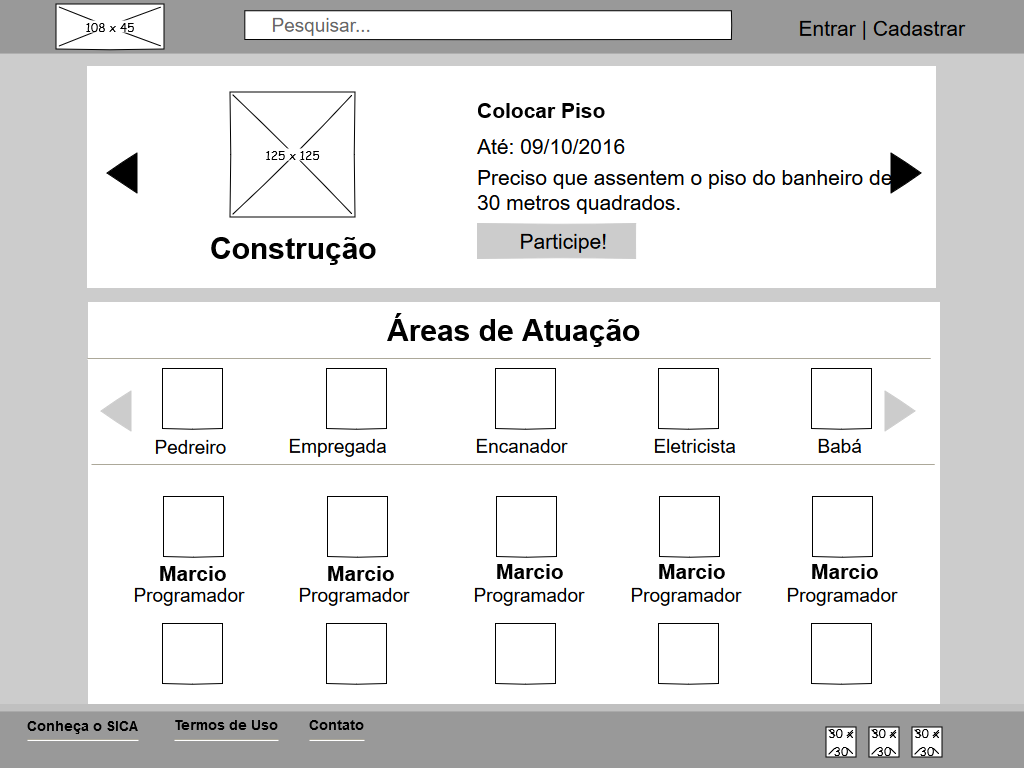


Figura 18 - Tela de protótipo da Index

A figura 19 ilustra o protótipo da tela de dados pessoais do autônomo. Essa tela contém um formulário em que ele deve inserir seus dados pessoais. A função desse cadastro é gerar um perfil para que ele seja encontrado por clientes. Além disso, o administrador gerencia melhor o sistema podendo ativar e inativar autônomos. Pelo fato do cadastro ser um processo extenso, a tela foi dividida em três partes: cadastro de dados pessoais, cadastro de perfil profissional e cadastro de mídia.

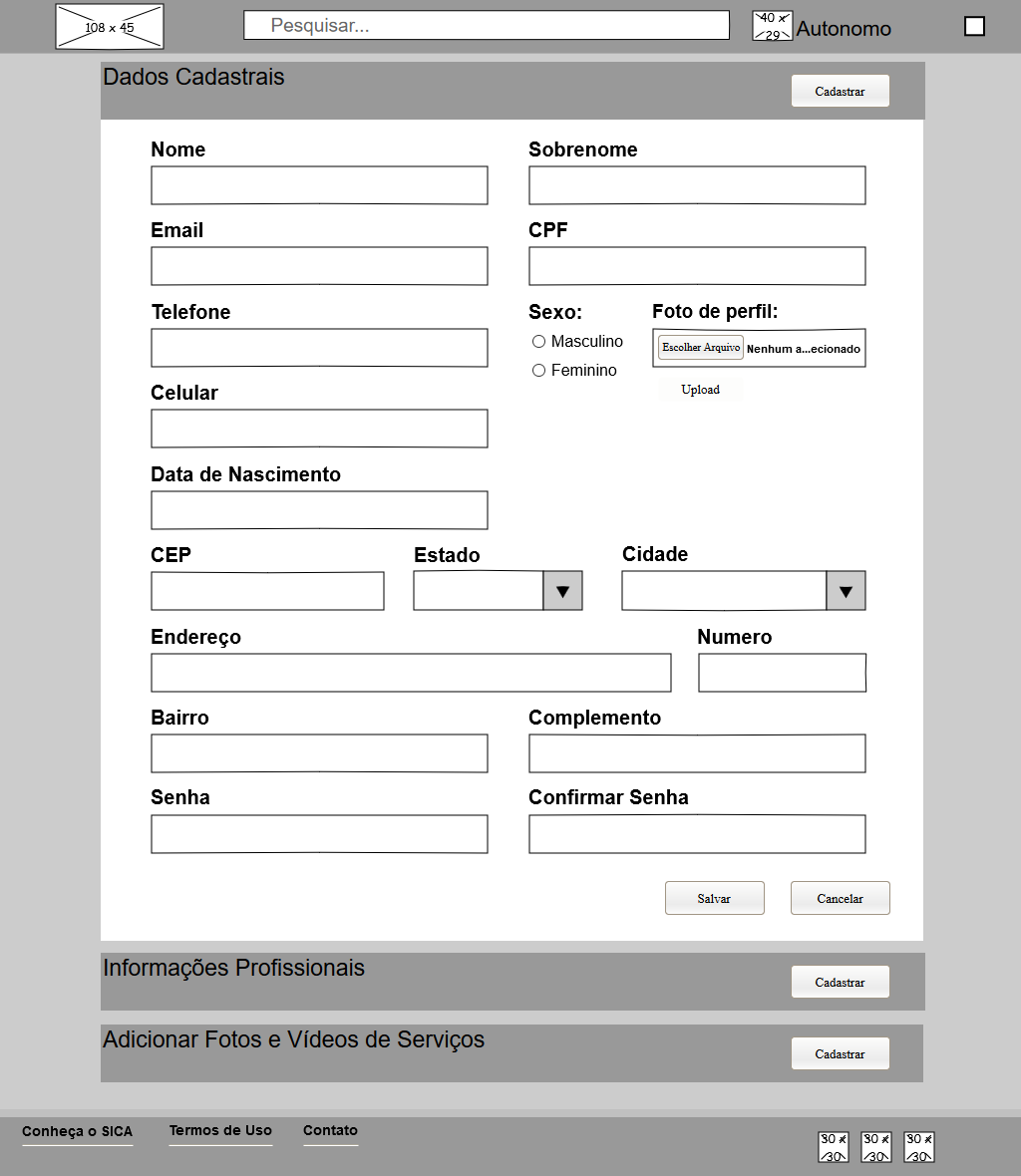


Figura 19 - Tela de protótipo de cadastro de dados pessoais

A figura 20 ilustra a continuação do cadastro do perfil profissional, o cadastro de perfil profissional. Neste momento, é necessário cadastrar a área de atuação, a profissão e suas especialidades. A área de atuação é um conjunto de profissões e o autônomo deverá escolher uma para que os clientes consigam filtrar o resultado de suas pesquisas.

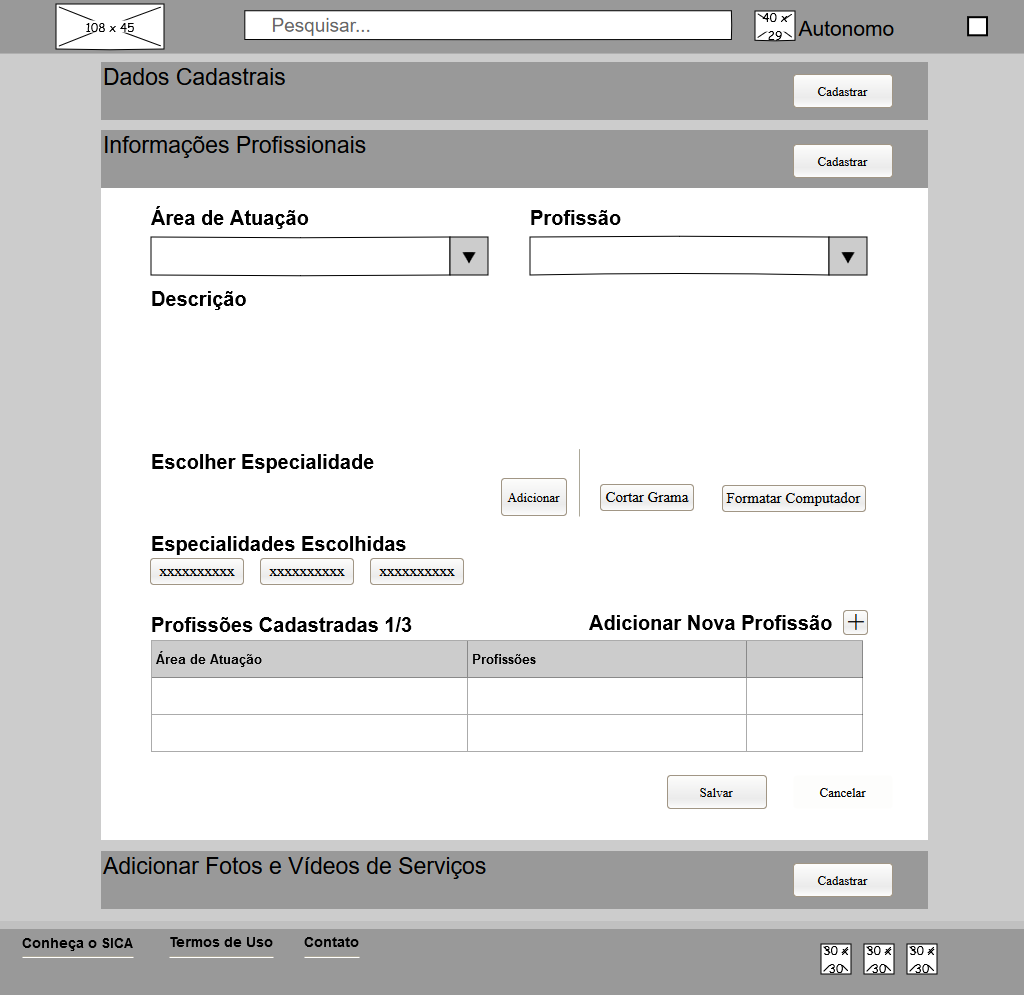


Figura 20 - Tela de protótipo de cadastro de profissional

Por último, a figura 21 ilustra o cadastro de mídia. Quando todas as informações já estiverem preenchidas, o autônomo poderá incluir no seu perfil dois tipos de mídia, foto e vídeo. Essas mídias estarão disponíveis no perfil do autônomo e poderão ser editadas posteriormente.

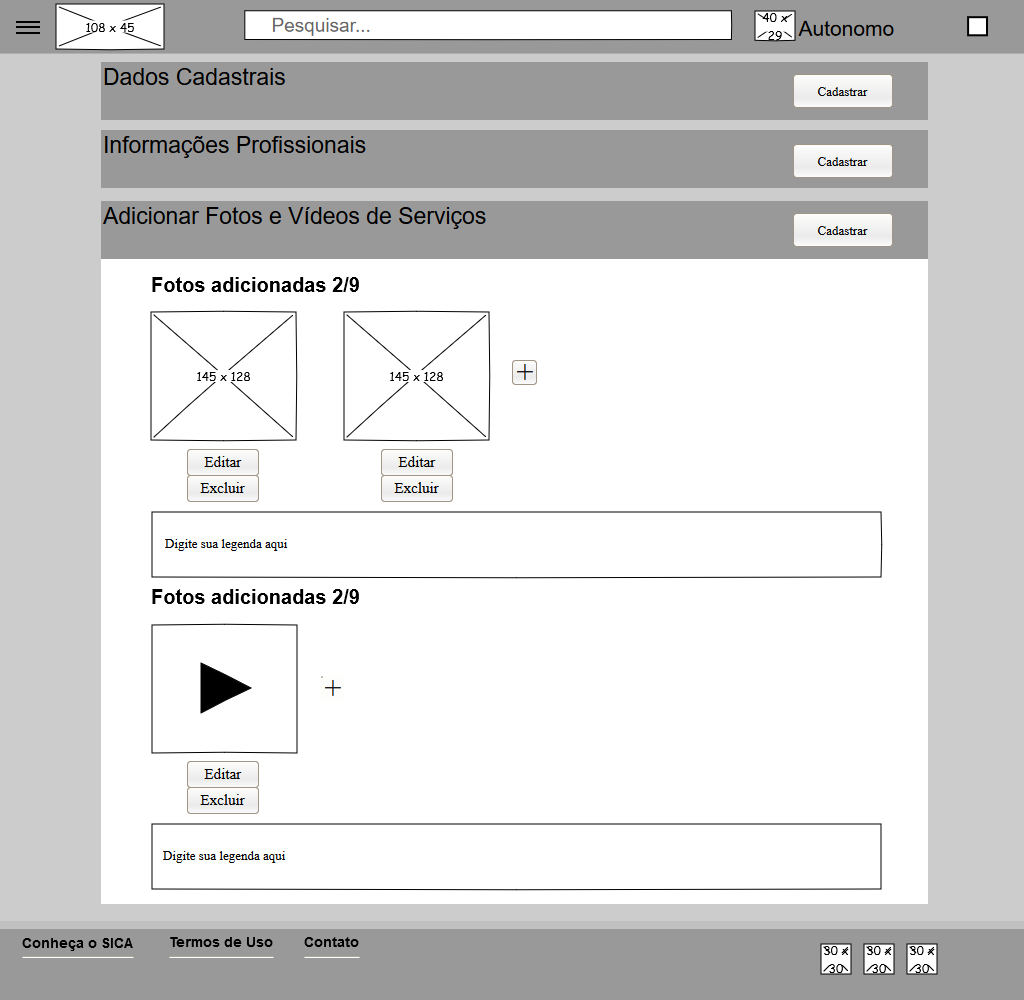


Figura 21 - Tela de protótipo de cadastro de mídia

A figura 22 ilustra a área de leilão, espécie de painel de controle do usuário que permite que ele faça busca por leilões, crie, visualize quais ele está participando e todos que ele já participou. Clicando-se na opção “leilão reverso”, opção encontrada no menu lateral, ilustrada na figura 27, o autônomo pode acessar essa área e executar a operação que deseja.



Figura 22 - Tela de protótipo da Área de Leilão

A figura 23 mostra como o usuário criará seu leilão. Deve-se preencher um formulário contendo um título do leilão, um prazo final, a área de atuação, seu estado e cidade e, por fim, uma breve descrição de sua real necessidade.

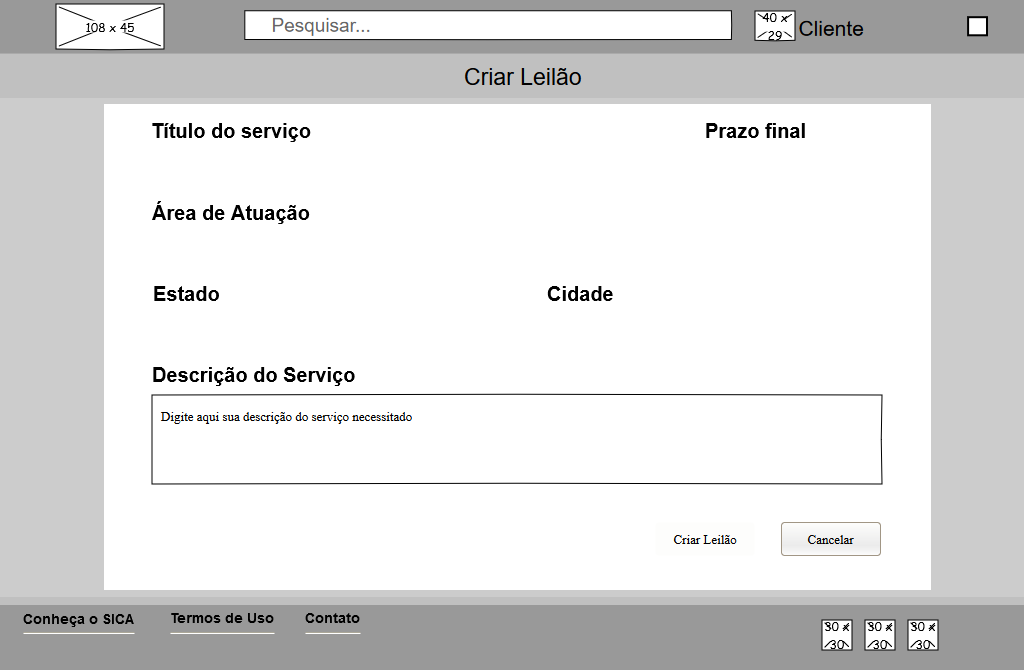


Figura 23 - Tela de protótipo de Criação de Leilão

A figura 24 ilustra a tela de histórico de leilão. Nessa tela, o usuário pode filtrar os leilões em aberto, os encerrados ou mesmo os dois juntos.

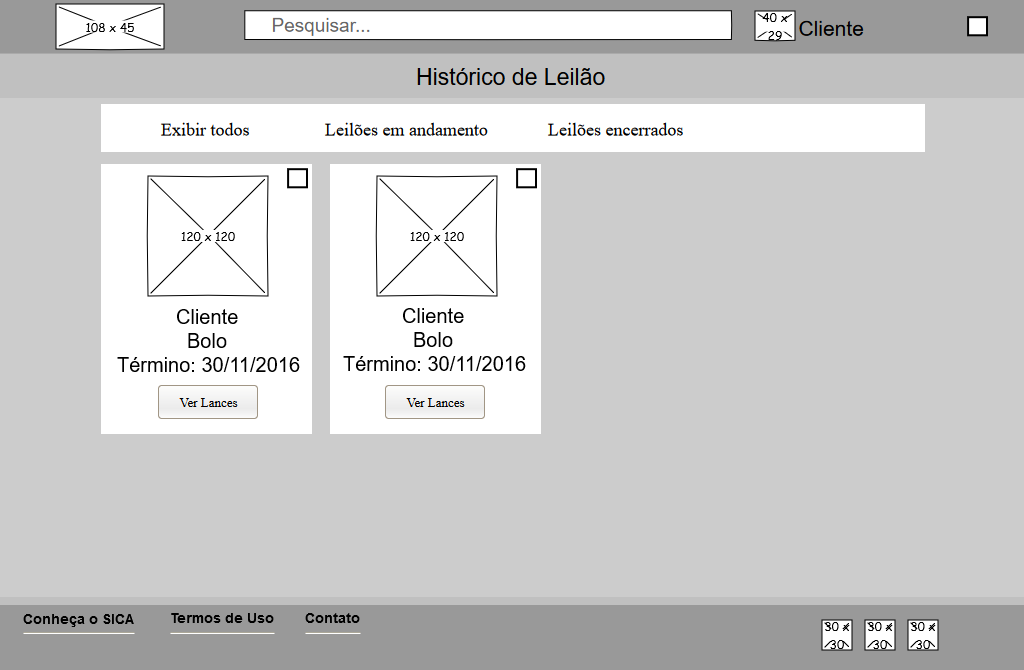


Figura 24 - Tela de protótipo de Histórico de Leilão

Na figura 25, apresenta-se a tela de dar lance em um leilão. Após o autônomo pesquisar o leilão e escolher o leilão que o agrade, ele poderá enviar um lance. Uma vez que o lance foi dado, ele poderá ser atualizado.



Figura 25 - Tela de protótipo de Enviar Lance

Na figura 26, o usuário Cliente e o Autônomo poderão visualizar os lances dados pelos autônomos. Para cada lance enviado é gerado um cartão contendo o nome, a avaliação e o valor do lance. Esse cartão pode ser visto na figura 26 logo abaixo da escrita “Ordenar por”, espécie de filtro disponível ao usuário, e assim aceitar o lance mais adequado.

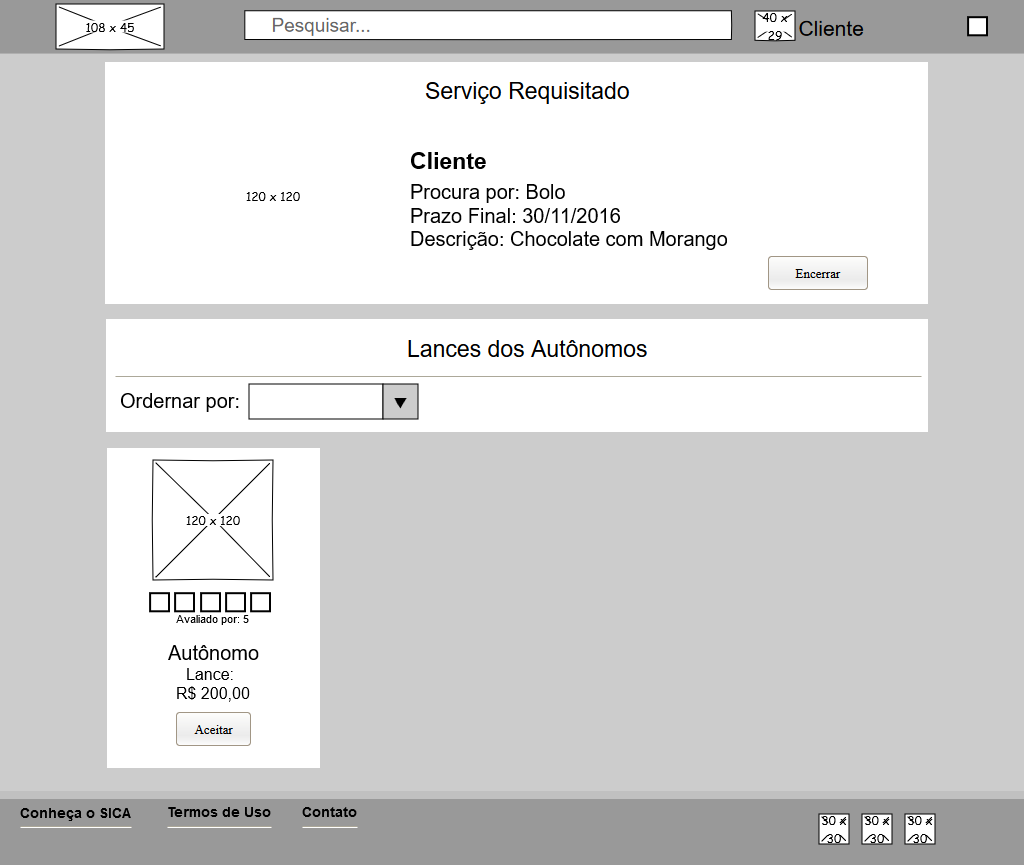


Figura 26 - Tela de protótipo de Visualizar Lances

Por último, a figura 27 ilustra o menu do autônomo. Esse menu facilita a sua navegação para todas as partes do sistema. Todo usuário do sistema terá um menu e, dependendo do seu perfil, outras opções estarão disponíveis. Por exemplo, o usuário Administrador tem acesso a uma área exclusiva de gerenciamento de usuários e, portanto, não poderá ser vista no menu do usuário Cliente ou Autônomo.

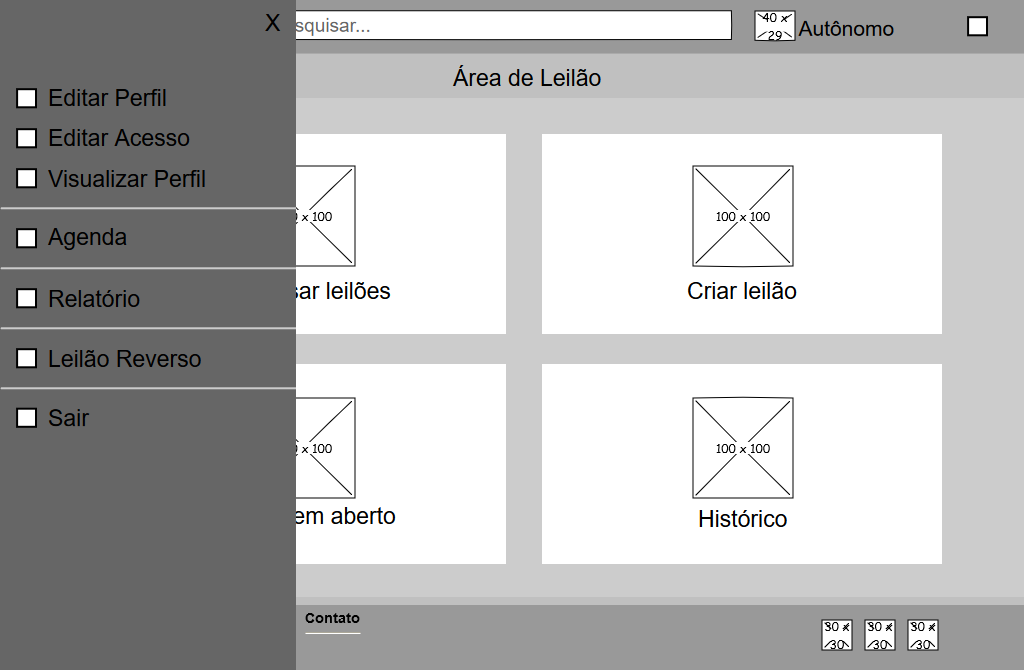


Figura 27 - Tela de protótipo do Menu Lateral do Autônomo

### 8.4.2 Desenvolvimento

Serão explicadas no subtópico 8.4.2 as principais funcionalidades, das principais telas do sistema, e como foi feita sua programação. Vale ressaltar que as funcionalidades se encontram no Apêndice A deste documento.

A seguir, ilustrado na figura 28, está a tela a index, página inicial do sistema, que poderá ser visualizada por qualquer usuário que estiver acessando o sistema, independente se este é cadastrado ou não. Nessa tela, ao lado do logo D2VS, está a barra de pesquisa com o ícone de uma lupa. Essa barra permite pesquisar autônomos e leilões filtrado por estado e cidade. Será explicado, pois, como foi feita a sua programação.

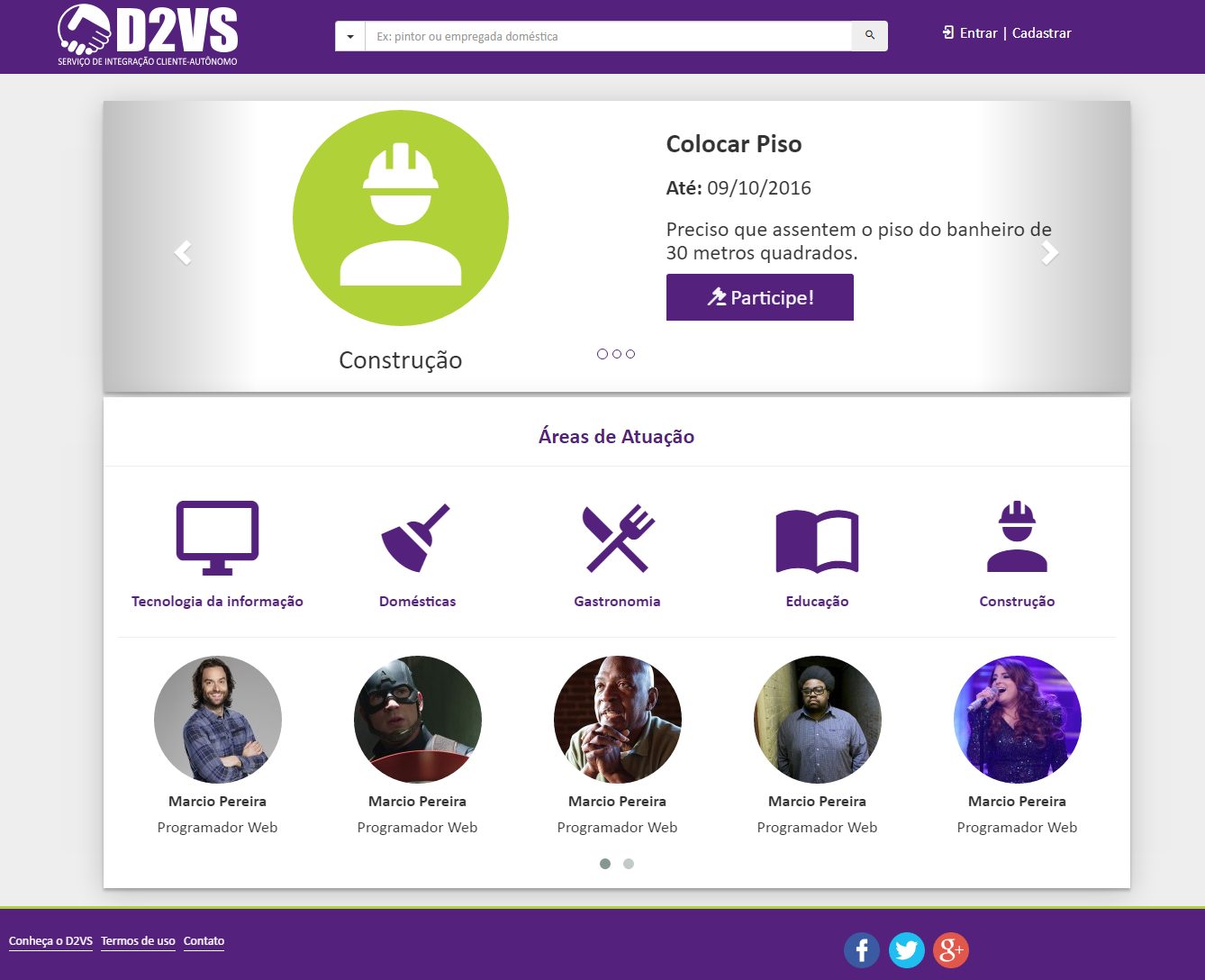


Figura 28 - Index do sistema D2VS

Ao clicar no ícone da lupa, é chamado um método que valida o tipo de pesquisa, pesquisa por autônomo ou por leilão conforme ilustra a figura 29. Se a pesquisa por autônomo for escolhida, o sistema irá direcionar o usuário para a tela de buca de profissionais, caso contrário, o usuário será redirecionado para a pesquisa de leilões.

protected void lbtPesquisar\_Click(object sender, EventArgs e)

{

if (rblTipoPesquisa.SelectedValue == "1")

{

Response.Redirect("BuscaDeProfissional.aspx?prof=" + txtPesquisa.Text + "&cid=" + ddlCidade.SelectedValue + "&t=" + rblTipoPesquisa.SelectedValue);

}

else

{

Response.Redirect("AutonomoOferta.aspx?l=" + txtPesquisa.Text + "&c=" + ddlCidade.SelectedValue + "&t=" + rblTipoPesquisa.SelectedValue);

}

}

Figura 29 – Validação do tipo de pesquisa

A figura 30 identifica a tela de cadastro de dados pessoais autônomo.

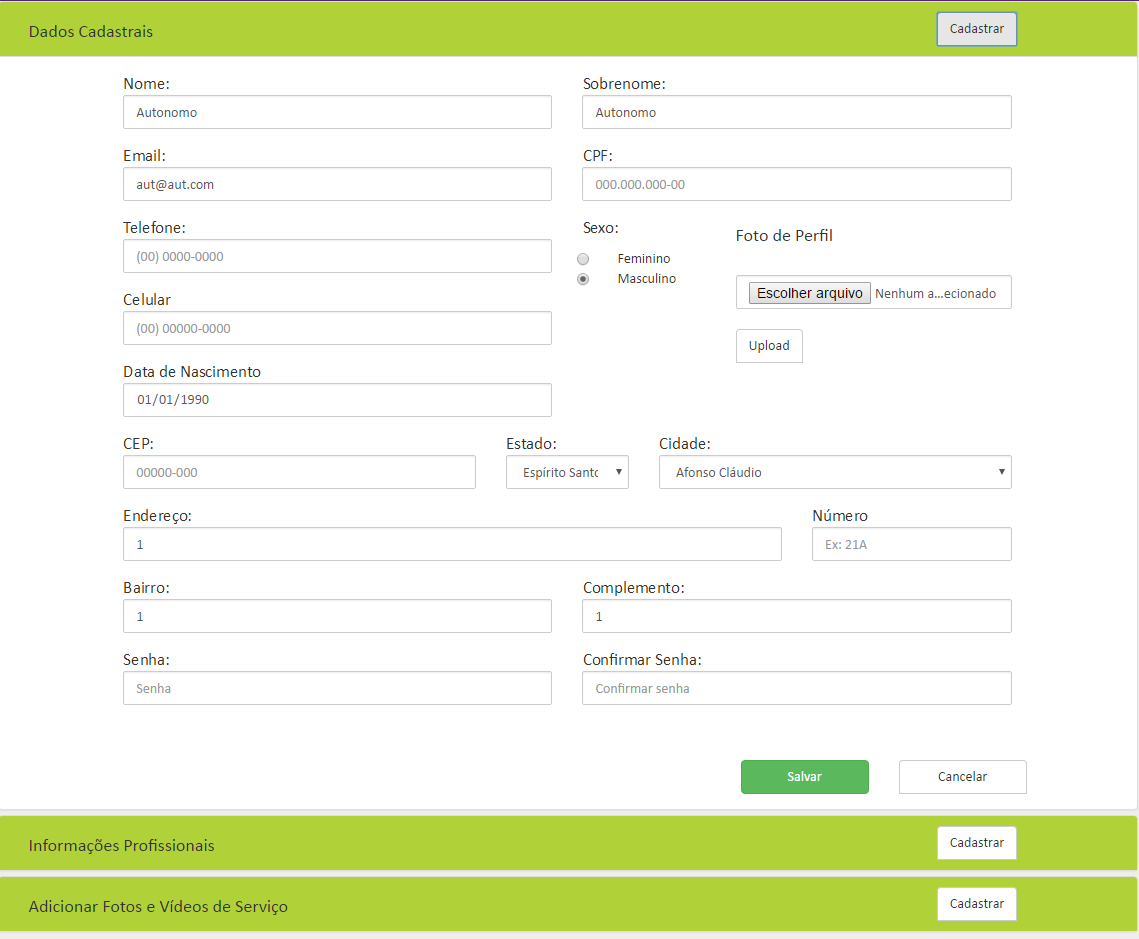


Figura 30 - Tela de protótipo de cadastro de dados pessoais

A figura 31 representa a segunda parte do cadastro profissional do autônomo.

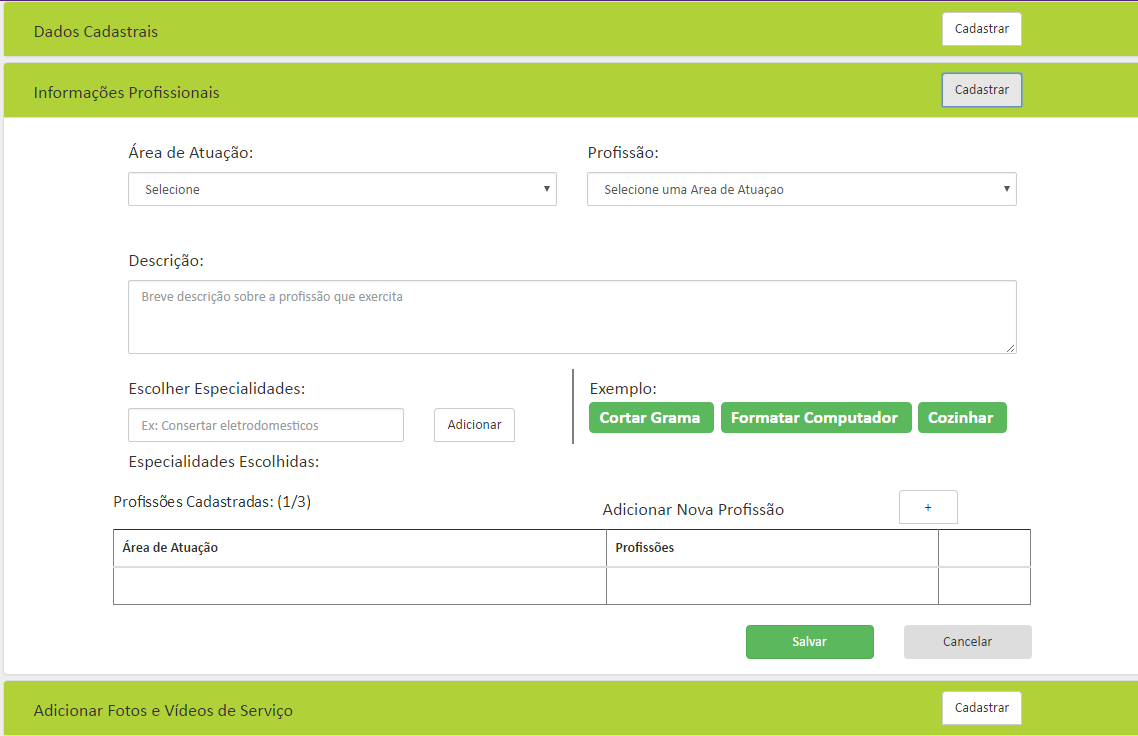


Figura 31 - Tela de cadastro profissional do autônomo

A figura 32 ilustra a última parte do cadastro do autônomo, o cadastro de mídia.

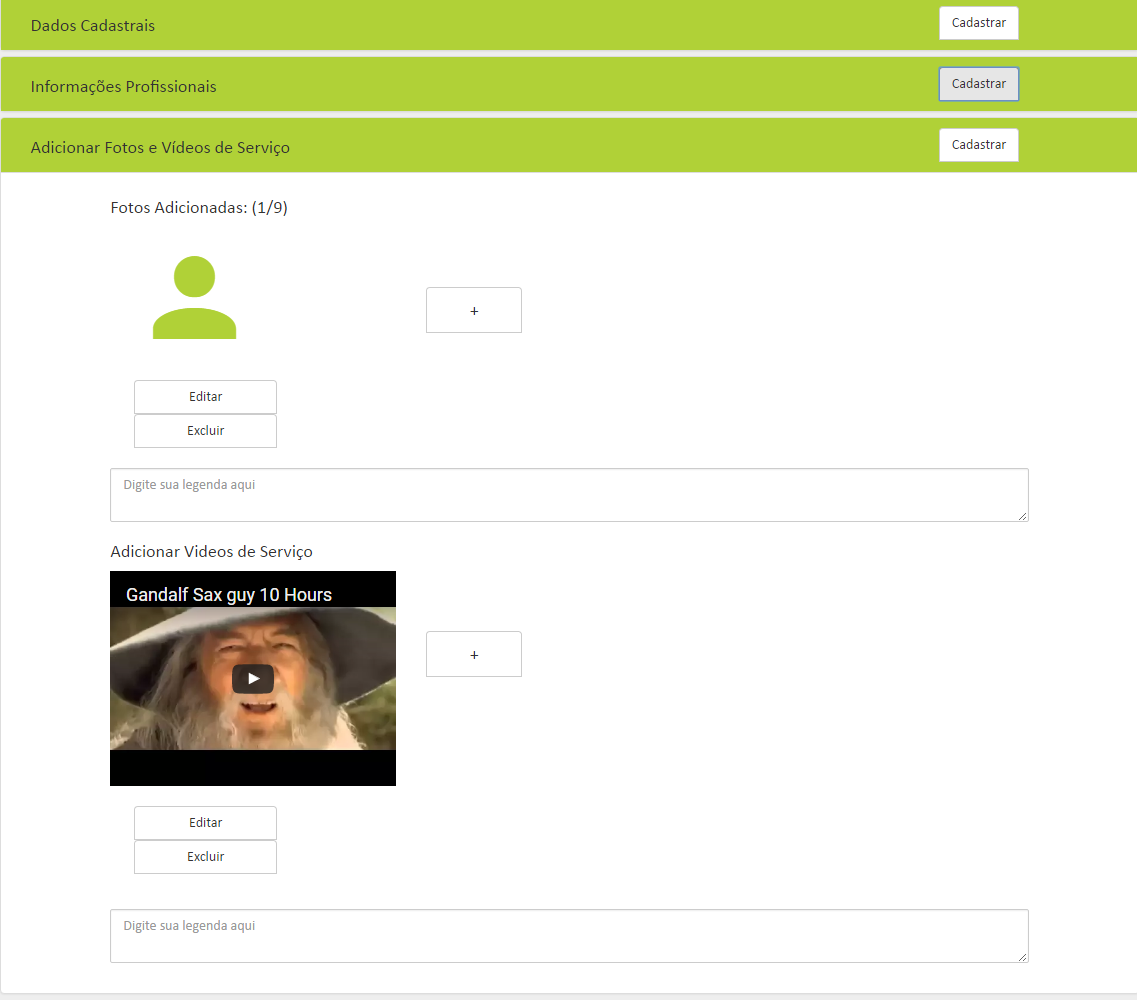


Figura 32 - Tela de cadastro de mídia

Todo esse desenvolvimento cadastral do autônomo teve uma programação bem semelhante durante as três fases ilustradas nas figuras 30, 31 e 32. Houve mudanças apenas em nome de objetos e nos seus atributos, portanto, será explicado apenas o trecho do cadastro de dados profissionais do autônomo, especificamente, de suas especialidades.

Ao ser carregada, a página de cadastro de dados pessoais do autônomo verifica se o autônomo está logado, caso estiver, três funções são chamadas: CarregarFormulario(), CarregarEspecialidade() e CarregarProfissoes. Seus dados pessoais, profissionais e as mídias serão preenchidas automaticamente nos campos dos formulários facilitando, no futuro, a atualização de alguma informação. A figura 33 ilustra como esse processo foi feito.

if (Session["LOGIN"] != null) {

Usuario usuario = (Usuario)Session["LOGIN"];

if (usuario.TipoUsuario.Codigo== 1) {

CarregarFormulario();

CarregarEspecialidades();

CarregarProfissoes();

}

Figura 33 – Verificação de usuário e preenchimento de formulário

Caso o usuário estiver fazendo um cadastro, o processo será diferente. Será criado um objeto “especialidade” do tipo “Especialidade” que receberá todas as informações do formulário. Após o objeto especialidade ter sido preenchido, um método da classe de persistência, que é um método responsável por inserir as informações no banco de dados do sistema. Destacado em amarelo na primeira linha da figura 34 está a criação do objeto especialidade, na última, a chamada do método de persistência.

Especialidade especialidade = new Especialidade();

especialidade.Nome = txtEscolherEspecialidade.Text;

especialidade.Autonomo = autonomo;

Profissao profissao = new Profissao();

profissao.Codigo = Convert.ToInt32(ddlProfissao.SelectedValue);

especialidade.Profissao= profissao;

EspecialidadeBD.Insert(especialidade);

Figura 34 – Cadastro de especialidade do autônomo

Pelo fato do módulo leilão reverso ser a principal funcionalidade do sistema, decidiu-se por explicar as principais etapas do desenvolvimento. A seguir, ilustrado na figura 35, verifica-se que os usuários Cliente e Autônomo têm acesso através de uma área às pesquisas de leilões, criação, visualizar leilões abertos e seus históricos. Vale ressaltar que a área de leilão oferece todas as funcionalidades para ambos os usuários, salvo “Dar lance”, funcionalidade exclusiva do Autônomo.

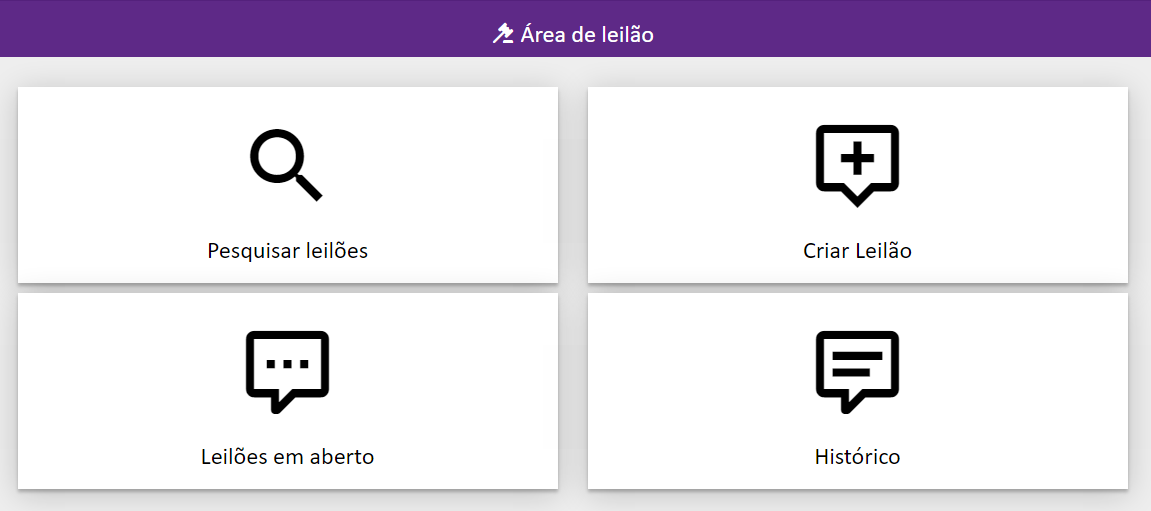


Figura 35 - Tela da área de leilão dos usuários Cliente e Autônomo

Uma das principais funcionalidades do Cliente e do Autônomo é a capacidade de criar leilão reverso preenchendo um formulário como ilustra a figura 36.



Figura 36 – Formulário de criação de leilão

No desenvolvimento *front-end*, não só na criação do leilão, mas em todas as telas que contenham formulário, utilizou-se o controle *RequiredFieldValidator*. Esse controle aplicado em campos específicos do formulário garante que este não pode estar vazio. Desse modo, o usuário preenche corretamente as informações prevenindo erros de preenchimento. Destacado em amarelo, a linha do código ilustrada na figura 37 mostra a aplicação desse controle sendo aplicado no campo “Título do Serviço” do formulário da figura 18. Parte interessante desse controle é o uso do atributo “*ErrorMessage”* que permite definir uma mensagem de erro ao usuário.

<div class="col-lg-6 col-lg-offset-1">

<asp:ValidationSummary ID="vlsSumario" runat="server" CssClass="alert alert-danger" HeaderText="Erros:" />

<asp:Label ID="lblServicoReq" runat="server" Text="Título do serviço:" Font-Size="Large"></asp:Label>

<asp:RequiredFieldValidator ID="rfvServicoReq" runat="server" ErrorMessage="Título obrigatório" ControlToValidate="txtServicoReq" Text="\*" ForeColor="Red"></asp:RequiredFieldValidator>

<div class="input-group">

<asp:TextBox ID="txtServicoReq" CssClass="form-control col-xs-12" runat="server" placeholder="Ex: pintar casa, formatar computador" MaxLength="29" Width="350"></asp:TextBox>

</div>

</div>

Figura 37 – Componente validador de campos

No *back-end,* as informações preenchidas e validadas no formulário estão sendo coletadas e armazenadas num banco de dados. Quando o usuário finaliza o preenchimento e clica no botão “Criar leilão”, botão verde localizado ao final do formulário, é ativado uma sequência de etapas para fazer a inserção dos dados no banco. O código da figura 38 mostra como foi feita essa sequência de etapas. Ao clicar no botão “Criar leilão”, o método chamado lbtCriarLeilao\_Click é acionado. Esse método instancia um objeto chamado “lei” do tipo “Leilao”. Esse objeto recebe os valores vindo do formulário. Destacado em amarelo, a primeira linha mostra como é feita a instanciação de um objeto, a segunda, a atribuição do valor a ele.

Na primeira linha destacada em verde, está a chamada de um método de uma classe de persistência. Inserido numa estrutura de decisão “if”, o método valida se as informações foram corretamente inseridas. Caso tudo esteja correto, o método retorna o valor 0. Ao final da estrutura de decisão será exibida uma mensagem ao usuário informando que o leilão começou como mostra a segunda linha destacada em verde.

protected void lbtCriarLeilao\_Click(object sender, EventArgs e)

{

Leilao lei = new Leilao();

lei.Titulo = txtServicoReq.Text;

lei.DataHoraInicio = DateTime.Now;

lei.DataPrevisao = DateTime.Now.AddMonths(1);

lei.DataHoraFim = Convert.ToDateTime(txtPrazoFinal.Text);

lei.Descricao = txtDescricaoCompleta.Text;

lei.Encerrado = 'N';

Atuacao atu = new Atuacao();

atu.CodigoAtuacao = Convert.ToInt32(ddlAtuacao.SelectedValue);

lei.Atuacao = atu;

Cidade cid = new Cidade();

cid.CodigoCidade = Convert.ToInt32(ddlCidade.SelectedValue);

lei.CodigoCid = cid;

Usuario usu = (Usuario)Session["LOGIN"];

lei.CodigoUser = usu;

if (LeilaoDB.InsertLeilao(lei) == 0)

{

txtServicoReq.Text = string.Empty;

txtDescricaoCompleta.Text = string.Empty;

txtPrazoFinal.Text = string.Empty;

ddlEstado.SelectedIndex = 0;

ddlCidade.SelectedIndex = 0;

ddlAtuacao.SelectedIndex = 0;

Response.Write("<script>alert('Começou o leilão!');</script>");

Response.Write("<script>window.location.href='HistoricoLeilao.aspx';</script>");

}

}

Figura 38 – Desenvolvimento *back-end* para a criação do leilão

Após a criação do leilão, tanto o Autônomo e o Cliente terão acesso aos leilões criados por meio da tela “Histórico de leilão”. A figura 39 ilustra essa com dois leilões criado.

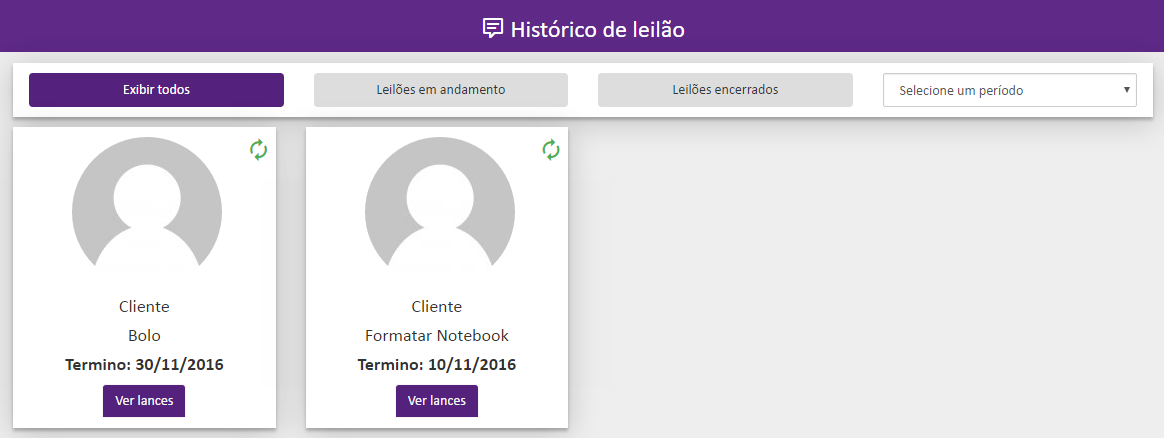


Figura 39 - Tela de histórico de leilão

Outra funcionalidade importante e exclusiva do autônomo é dar lance em leilões em aberto. A figura 40 ilustra a tela em que um autônomo envia um lance a um leilão.

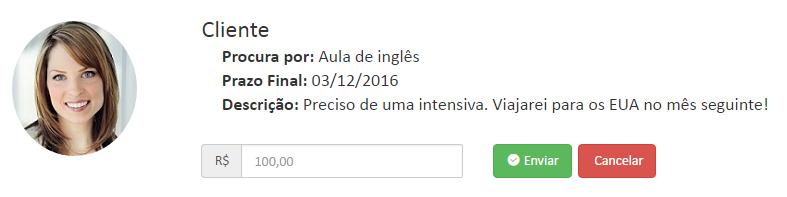


Figura 40 – Tela de envio de lances

No desenvolvimento *back-end*, todas as informações sobre o leilão como, por exemplo, o serviço procurado e a descrição estão sendo resgatadas do banco de dados do sistema.

O código para gerar a tela da figura 21 foi divido em duas partes. A primeira parte, ilustrada na figura 41, refere-se às informações do leilão como a imagem do cliente e todas as informações escritas a direita dela. As informações resgatadas do banco de dados foram armazenadas num objeto “lei” do tipo Leilao e colocadas entre um código HTML, destacado em amarelo na figura, para que a tela fosse gerada corretamente.

lblLeilao.Text =

"<div class='col-xs-2 col-xs-offset-1'>" +

"<img src = 'img/imgOfertaServicoCliente.jpg' class='img1' />" +

"</div>" +

"<div class='col-xs-8'>" +

"<h3 class='textos2'>" + lei.CodigoUser.Nome + "</h3>" +

"<h4 class='textos2'><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Procura por: </b> " + lei.Titulo.ToString() + "</h4>" +

"<h4 class='textos2'><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Prazo Final: </b>" + lei.DataHoraFim.ToShortDateString().ToString() + "</h4>" +

"<h4 class='textos2'><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Descrição: </b>" + lei.Descricao.ToString() + "</h4>" +

"</div>";

Figura 41 – Código para gerar as informações sobre o leilão

A segunda parte, ilustrada na figura 42, refere-se ao campo em que um valor será passado e aos dois botões, Enviar e Cancelar em verde e vermelho respectivamente. Esses elementos, a princípio, ficam escondido do autônomo até que sejam validados. É necessário verificar se o autônomo está tentando dar lance no próprio leilão, não há sentido nisso.

string darLance =

"<input id='ValorLance' name='ValLance' class='form-control' aria-label='Amount (to the nearest dollar)' placeholder='100,00' />" +

"<button id='btnEnviarOferta' class='btn btn-success' onclick='lanceEnvia(); return false;'\"><i class='mdi mdi-check-circle' style='color: white'></i>&nbsp;Enviar</button>&nbsp;&nbsp;" +

"<button id='btnCancelarOferta' class='btn btn-danger' data-dismiss='modal' onclick='limpaLance(); return false;'>&nbsp;Cancelar</button>" +

Figura 42 – Código para gerar o campo do lance e os botões

Uma vez que o lance foi enviado, ele será exibido para o cliente ou autônomo que iniciou o leilão em forma de um cartão que pode ser visto ao lado esquerdo da seta vermelha na figura 43.

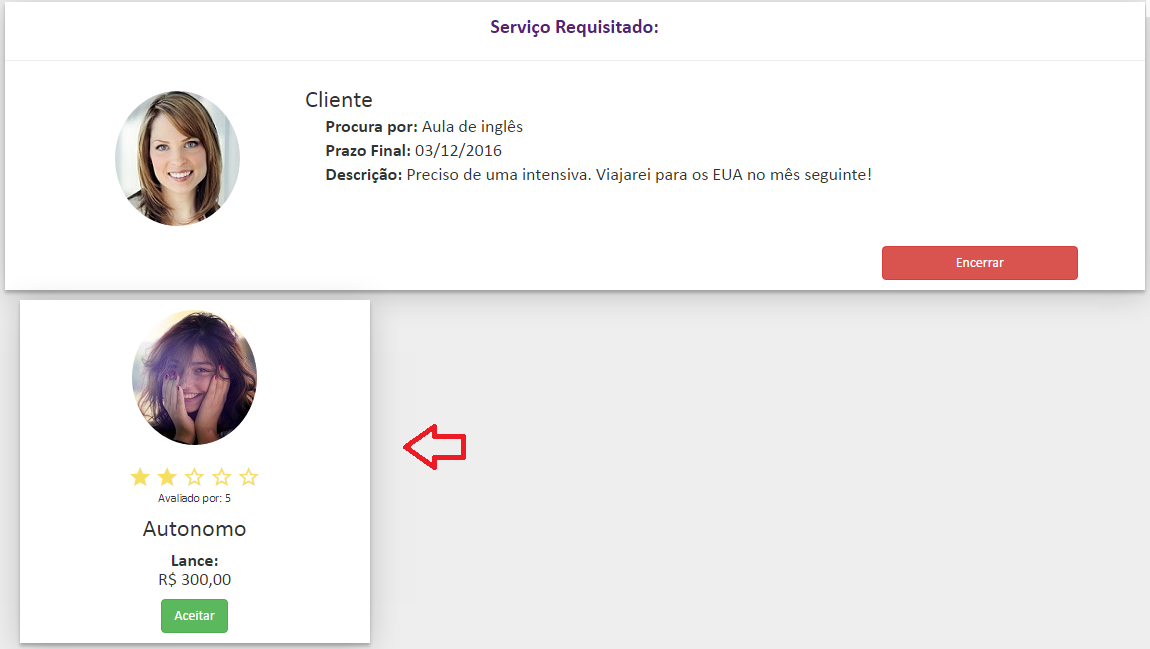


Figura 43 – Cartão gerado a partir de um lance

Todos os lances estão sendo armazenado num *DataSet* chamado ds, objeto que armazena um conjunto de dados vindo do banco de dados. Os dados armazenados nele são percorridos por um uma estrutura de repetição *foreach*. No código destacado em amarelo, na figura 44, por meio do comando dr[“USU\_NOME”].toString(), o nome do autônomo é acessado no DataSet e exibido numa estrutura HTML.

DataSet ds = LeilaoDB.SelecionarLeilaoUsuarioPorCodigoLeilao(leilao);

foreach (DataRow dr in ds.Tables[0].Rows)

"<h3 style='margin-top: 0px'><div id='nomeAut'>" + dr["USU\_NOME"].ToString() + "</div></h3> " +

"<h4><b>Lance:</b></br> <div id='lanceAut'>R$ " + dr["AUL\_VALOR"].ToString() + "</div> </h4> " + "</div> " +

"<div class='row'> " +

botao +

"</div> " +

Figura 44 – Lance de um autônomo em forma de cartão

Por último, a figura 45 ilustra o menu lateral do autônomo fornecendo-o acesso a todas as funcionalidades do sistema disponível para esse perfil de usuário, neste caso, Autônomo.

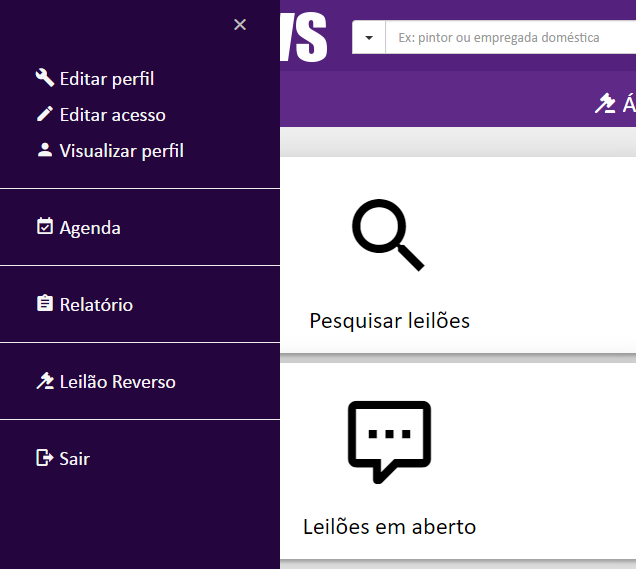


Figura 45 - Tela de menu lateral

Por se tratar de um desenvolvimento semelhante, a figura 46 ilustra um exemplo de como é feito a programação de um item do menu, a opção “Leilão Reverso”. Esse botão foi feito usando o “*LinkButton*”, componente *asp* pois permite usar ícone e outras funcionalidades. A figura 46 ilustra a aplicação desse botão na interface gráfica do usuário. Destacado em amarelo, entre a tag <asp:LinkButton> </asp:LinkButton>, está a tag <i>, marcação HTML para uso de ícones.

<asp:LinkButton ID="btnLeilao" runat="server" CssClass="btn btn-menu" OnClick="btnLeilao\_Click" CausesValidation="false"><span style="font-size:20px"><i class="mdi mdi-gavel"></i>&nbsp;Leilão Reverso </span></asp:LinkButton>

Figura 46 – LinkButton na interface gráfica do usuário

Quando o botão “Leilão reverso” é clicado, um método chamado “btnLeilao\_Click*”* executa uma ação de redirecionamento para a página “AreaLeilao.aspx” por meio do comando *Response.Redirect* ilustrado na figura 47.

protected void btnLeilao\_Click(object sender, EventArgs e) {

Response.Redirect("AreaLeilao.aspx");

}

Figura 47 – Redirecionamento do botão para área de leilão

1. https://www.w3.org/WAI/gettingstarted/Overview.html [↑](#footnote-ref-1)
2. Guia de boas práticas de acessibilidade: https://www.w3.org/TR/UNDERSTANDING-WCAG20/Overview.html#contents [↑](#footnote-ref-2)
3. APIs (*application programming interfaces*) conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web [↑](#footnote-ref-3)