**Projeto de Software**

**Equipe:**

Francisco Anderson Furtado Sarmento

Francisco César da Silva Filho

João Malvino Júnior

Curso: Bacharelado em Ciência da Computação

Instituição: Faculdade de Filosofia Ciências e Letras de Cajazeiras - FAFIC Rua Padre Ibiapina, s/n - Centro, Cajazeiras - PB, 58900-000

**SUMÁRIO**

1. INTRODUÇÃO……………………………………………………………...…….…..Pag 02

1.1. DEFINIÇÃO DO PROBLEMA………………………………………..…...…….…..Pag 02

1.2. JUSTIFICATIVA…………………………………………………………….…….…..Pag 02

1.3. OBJETIVOS ……………………………………………………………….……..…..Pag 02

1.3.1. OBJETIVO GERAL ……………………………………………………….……..…..Pag 02

1.3.2. OBJETIVOS ESPECÍFICOS.. ………………………………………………….…..Pag 02

1.4. ORGANIZAÇÃO DO TRABALHO …………………..…………...…………….…..Pag 03

2. IDENTIFICAÇÃO DO PROJETO …………………………………..…………..…..Pag 03

2.1. IDENTIFICAÇÃO DA INSTITUIÇÃO …………………………………..…..…..…..Pag 03

2.3. IDENTIFICAÇÃO DAS DISCIPLINAS………………………………………….…..Pag 03

2.2. IDENTIFICAÇÃO DOS ALUNOS …………………………………………...….…..Pag 03

3. O SISTEMA ……………………………………………………..……………..….…..Pag 04

3.1. O CONTEXTO …………………………………………….……………...……….…..Pag 04

3.1.1. SISTEMA …………………………………………………………………………..…..Pag 04

3.1.1.1. JUSTIFICATIVA …………………………………………………………………...…..Pag 05

3.1.1.2. CONTEXTO DE NEGÓCIO ……………………………………………..……….…..Pag 05

3.1.1.3. DURAÇÃO DO PROJETO ………………………………………………..…….…..Pag 05

3.2. ARTEFATOS E TECNOLOGIAS ……………………….……………………….…..Pag 05

3.2.1. RECURSOS HARDWARE ………………………………..………………….….…..Pag 05

3.2.2. RECURSOS DE SOFTWARE……………………………......………………….…..Pag 05

3.2.3. RECURSOS HUMANOS ……………………………….……………………….…..Pag 05

3.3. REQUISITOS …………………………………………………………..………….…..Pag 06

3.3.1. FUNCIONAIS …………………………………………………………..………….…..Pag 06

3.3.2. NÃO-FUNCIONAIS …………………………………………………...………….…..Pag 06

3.4. ANÁLISE E DESIGN………………………………………………………..…….…..Pag 08

3.4.1. DIAGRAMA DE CASOS DE USO ………………………..……………………..…..Pag 08

3.4.2 DIAGRAMA DE ATIVIDADE…………………………………..………………….…..Pag 09

3.4.3. DIAGRAMA DE CLASSES …………………………………...………………….…..Pag 10

3.4.4. DESCRIÇÃO CONCEITUAL (Minimundo) …………………………………….…...Pag 13

3.4.5. LINGUAGENS E FERRAMENTAS ……………………………………………..…..Pag 13

3.4.6. DIAGRAMA ENTIDADE-RELACIONAMENTO (DER)…………………………….Pag 14

3.4.7. MODELO RELACIONAL…………………………………………………….……..…..Pag 15

3.4.8. CRIAÇÃO FÍSICA DO MODELO DE DADOS - SQL……………………...…....…..Pag 16

3.4.9. TELAS ……………………………………………………………………...….....….…..Pag 28

4. CONCLUSÃO……………………………………………………..………...….….…..Pag 31

5. REFERÊNCIAS……………………………………………...……………...….….…..Pag 31

**1. INTRODUÇÃO**

O sistema de gerenciamento de Auto Escola(SysAut) irá controlar o cadastro de clientes e aulas práticas dos mesmos, como também a parte financeira da empresa.

Para a realização do cadastro de clientes, é necessário realizar uma pesquisa para verificar se o mesmo já está cadastrado no sistema, se não está, será feito um novo cadastro com os seguintes dados do cliente: nome, RG(opcional), CPF(obrigatório), endereço, telefone e pagamento.

Este sistema também irá realizar outras operações, tais como: cadastro de instrutor, cadastro de veículos e impressão de aulas. Para realização do cadastro das aulas práticas, deve ser cadastrado um cliente e um instrutor e o dia da aula como também o veículo.

**1.1. DEFINIÇÃO DO PROBLEMA**

A falta de sistemas completos e ágeis para auto escola nos motivou a criar esse projeto no qual iremos desenvolver o mesmo.

**1.2. JUSTIFICATIVA**

A motivação para implantação do sistema SysAut foi a necessidade de um sistema que auxiliasse as auto escolas a controlar suas aulas práticas com mais facilidade, pois o mesmo pode cadastrar o professor, o aluno, e a aula prática, e gerar o relatório diário de quem irá fazer aula neste dia, assim não gerando um choque de horário com os demais alunos.

**1.3. OBJETIVOS**

**1.3.1. OBJETIVO GERAL**

O objetivo geral desse sistema é facilitar e agilizar o dia a dia do instrutor de uma auto escola

**1.3.2. OBJETIVOS ESPECIFICOS**

* Gerar relatórios dos alunos que irão fazer aulas no dia
* Não dar choque de horário com outros alunos
* Gerar o relatório financeiro mensal

**1.4. ORGANIZAÇÃO DO TRABALHO**

O trabalho está organizado da seguinte forma: a seção dois apresenta os dados da instituição, das disciplinas do projeto, e dos alunos inclusos no mesmo. A seção 3 descreve o sistema da auto escola em detalhes mostrando processos de desenvolvimento utilizado e a descrição do sistema; a seção 4 conclui o trabalho.

**2. IDENTIFICAÇÃO DO PROJETO**

**2.1 IDENTIFICAÇÃO DA INSTITUIÇÃO**

**Nome:** Faculdade de Filosofia Ciências e Letras de Cajazeiras(FAFIC)

**Atividades:** Graduandos em Ciência da Computação

**Endereço:** Rua Pe. Ibiapina, s/n, Belo Horizonte, Cajazeiras - PB - Brasil - Cep: 58900.000

**Telefone:** (83) 3531-3500

**Site:**<http://www.fescfafic.edu.br/>

**2.2 IDENTIFICAÇÃO DAS DISCIPLINAS**

**Disciplina:** Engenharia de Requisitos

**Professor:** Sttiwe Washington

**Disciplina:** Banco de Dados

**Professor:** Rivanilson Rodrigues

**Disciplina:** Estrutura de Dados

**Professor:** Daniel Araújo

**2.3 IDENTIFICAÇÃO DOS ALUNOS**

**Nome:** Francisco César da Silva Filho

**Data de nascimento:** 03 Outubro 1997

**Endereço:** Rua Deocleciano Nunes de Rezende

**Cidade:** Sousa

**Bairro:** Doutor Zezé

**UF:** PB

**Estado:** Paraíba

**País:** Brasil**.**

**E-mail:** franciscocesar888@gmail.com

**Telefone**: (083) 9 9406-2713

**Curso:** Bacharelado em Ciência da Computação

**Nome:** Francisco Anderson Furtado Sarmento

**Data de nascimento**: 21 de Abril de 1989

**Endereço:** Rua Dom Pedro I Nº 54

**Cidade:** Sousa

**Bairro:** Alto Capanema

**UF:** PB

**Estado:** Paraíba

**País:** Brasil.

**E-mail:** andesonf100@gmail.com

**Telefone:** (083) 981154162

**Curso:** Bacharelado em Ciência da Computação

**Nome:** João Malvino Júnior

**Data de nascimento:** 09 de Setembro 1988

**Endereço:** Nelson Meira, 31

**Cidade:** Sousa

**Bairro:** Estação

**UF:** PB

**Estado:** Paraíba

**País:** Brasil**.**

**E-mail**: jmalvino77@gmail.com

**Telefone:** (083) 9 9118.6142

**Curso:** Bacharelado em Ciência da Computação

**3. O SISTEMA**

**3.1 O CONTEXTO**

**3.1.1 SISTEMA**

O SysAut (Sistema de Gerenciamento de Auto Escolas) é um sistema para auto escola que efetua o gerenciamento dos processos de auto escolas em geral.

**3.1.1.1 JUSTIFICATIVA**

A motivação para implantação do sistema de auto escolas vem dos benefícios que um sistema informatizado traz em comparação com funções executadas estritamente de forma manual. Neste caso, um ambiente automatizado para auto escola, iria agilizar os processos e facilitar para o instrutor e também para o aluno que em casos extremos não ocorreria mais choques de horários**.**

**3.1.1.2 CONTEXTO DE NEGÓCIO**

O sistema foi desenvolvido para atender à necessidade das auto escolas, incluindo o cadastro dos alunos e impressão de aulas práticas diárias.

**3.1.1.3 DURAÇÃO DO PROJETO**

O projeto pode levar até 300 horas para ser desenvolvido.

**3.2 ARTEFATOS E TECNOLOGIAS**

**3.2.1 RECURSOS HARDWARE**

Os computadores utilizados serão os de uso pessoal.

**3.2.2 RECURSOS DE SOFTWARE**

|  |
| --- |
| Ferramenta |
| · WordPress |
| · MySQL WorkBench |
| · Filezilla |

**3.2.3 RECURSOS HUMANOS**

|  |  |
| --- | --- |
| Membros da Equipe | Papéis |
| Francisco César | Gerente de Projeto, Programador, Analista, Testador |
| João Malvino | Programador, Testador |
| Francisco Anderson | Programador, Testador |

**3.3 REQUISITOS**

**3.3.1 FUNCIONAIS**

**Cadastro:**

Cadastrar Aluno: Realiza o cadastro de um aluno no banco de dados. Cadastrar Instrutor: Realiza o cadastro de um instrutor no banco de dados. Cadastrar Secretária: Realiza o cadastro de uma secretária no banco de dados. Cadastrar Veículo: Realiza o cadastro de um veículo no banco de dados. Cadastrar Aula: Realiza o cadastro de uma aula no banco de dados a partir dos dados de um instrutor e veículo. Cadastrar Aluno Aula: Realiza o cadastro de um aluno em uma aula cadastrada no banco.

**Pesquisas:**

Pesquisar Aluno: Realiza uma pesquisa de um aluno no banco, com a possibilidade de deletar ou editar o mesmo. Pesquisar Veículo: Realiza uma pesquisa de um veículo no banco, com a possibilidade de deletar ou editar o mesmo.

**Funcionários**

Pesquisar Pessoa: Realiza uma pesquisa de uma pessoa no banco, com a possibilidade de deletar ou editar o mesmo. Pesquisar Instrutor: Realiza uma pesquisa de um instrutor no banco, com a possibilidade de deletar ou editar o mesmo. Pesquisar Secretária: Realiza uma pesquisa de uma secretária no banco, com a possibilidade de deletar ou editar o mesmo.

**3.3.2 NÃO-FUNCIONAIS**

**Usabilidade;**

O sistema é intuitivo e promove ao usuário a capacidade de entrar com comando que permitam operar o sistema de modo mais fácil e eficiente.O esforço exigido para alcançar um determinado nível de desempenho é mínimo pois conta com uma interface agradável a realização de qualquer tarefa.

**Segurança;**

Autenticar usuário usando login e senha;

Autorizar usuário, criando os seguintes grupos:

* Administrador - Acesso geral a todas as funcionalidades do sistema, incluído o cadastro de usuário;
* Atendimento - Acesso às funcionalidades de atendimento a clientes;
* Clientes - Cadastros e agendamento de aulas.

**Portabilidade;**

Uso da linguagem Java e de bibliotecas e mecanismos de persistência capazes de rodar nos sistemas operacionais Windows e Linux.

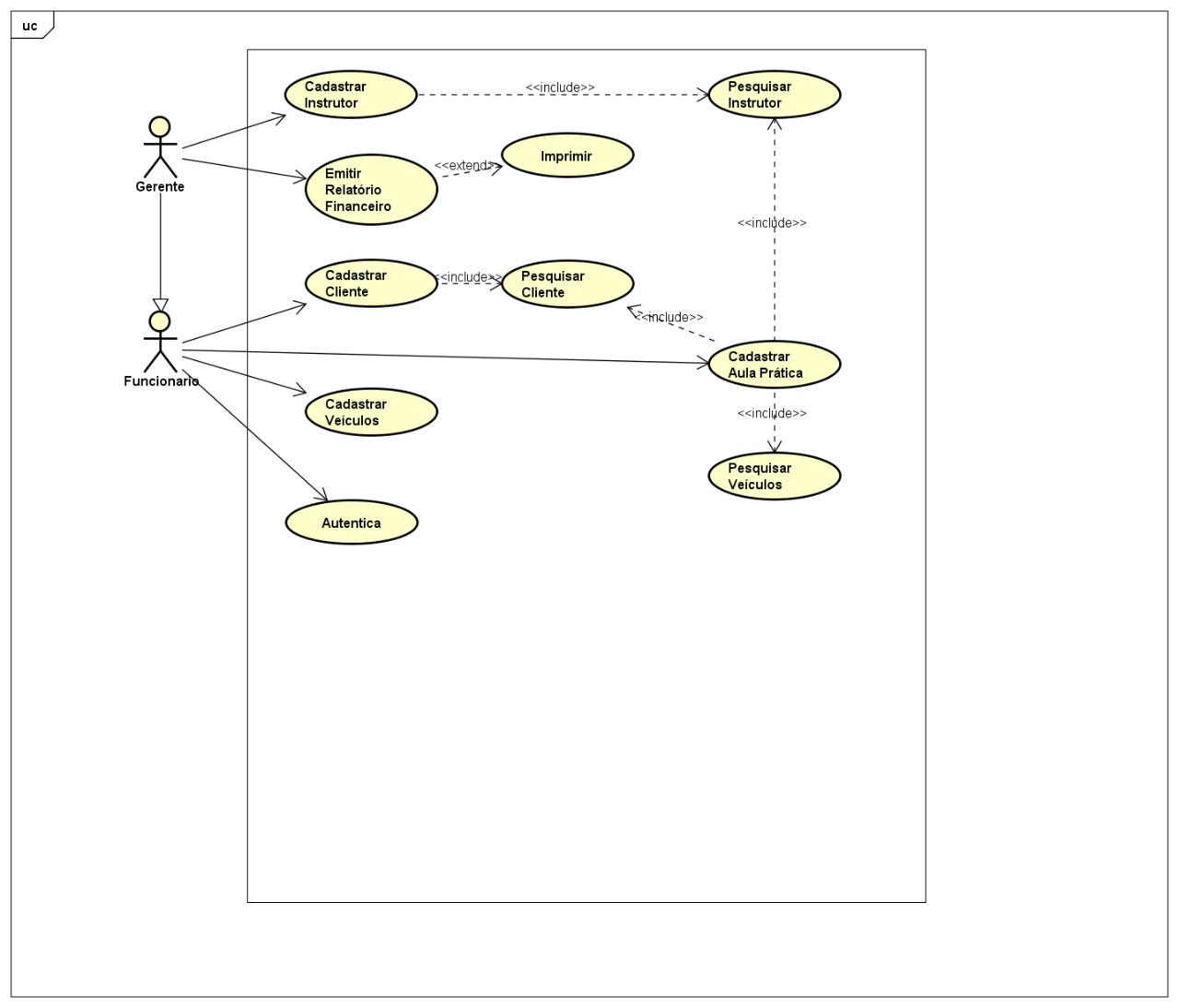
**Manutenibilidade;**

A organização do sistema é por meio de módulos que trabalhem em conjunto sem depender excessivamente de outros módulos;

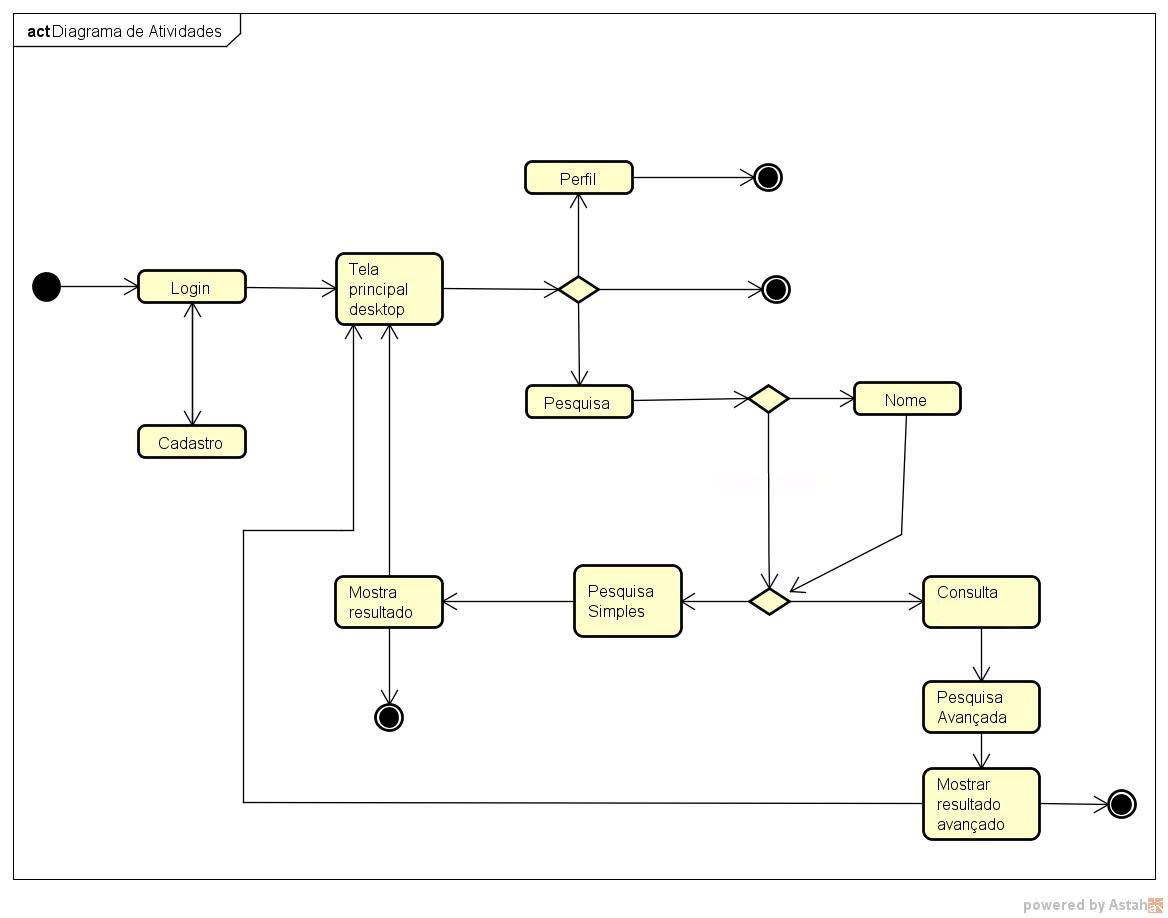
As interfaces têm o papel de ocultar informações específicas de cada módulo imprementado;

**3.4 ANÁLISE DE DESIGN**

**3.4.1 DIAGRAMA DE CASO DE USO**

****

**3.4.2 DIAGRAMA DE ATIVIDADES**

****

**3.4.3 DIAGRAMA DE CLASSES**

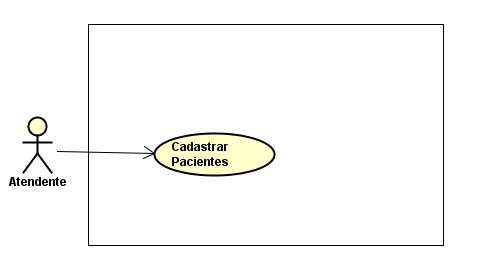
****

Figura 1. Diagrama de Casos de Uso Cadastrar Paciente

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | **Cadastrar paciente** |
| Caso de uso Geral |  |
| Ator Principal | Atendente |
| Atores Secundários |  |
| Resumo | Descreve as etapas percorridas pelo atendente para realizar o cadastro de um paciente |
| Pré-Condições | Realizar pesquisa com parte do nome do paciente verificando se já possui cadastro |
| Pós-Condições |  |
| **Fluxo Principal** | |
| Ações do Ator | Ações do Sistema |
| 1. Solicitar pesquisa do paciente |  |
| 2. Inserir parte do nome do paciente |  |
|  | 3. Localizar paciente com parte do nome inserido |
| 4. Solicitar Cadastro de Paciente |  |
| 5. Informar dados do Paciente |  |
|  | 6. Exibir mensagem de cadastrado realizado com sucesso |
| Restrições/validações | 1. O paciente deve apresentar um documento oficial com foto |
| **Fluxo Alternativo I – Paciente possui cadastro** | |
|  | 1. Exibir os dados do cliente já cadastrado e presentar as opções de Edição, Listar Exames e Realizar Exame. |

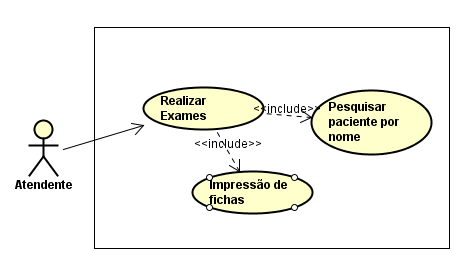


Figura 2. Diagrama de Casos de Uso Realizar Exames

|  |  |
| --- | --- |
| **Nome do Caso de Uso** | **Realizar Exames** |
| Caso de uso Geral |  |
| Ator Principal | Atendente |
| Atores Secundários |  |
| Resumo | Descreve as etapas percorridas pelo atendente para realizar o exame de um paciente |
| Pré-Condições | Realizar pesquisa do paciente pelo nome. |
| Pós-Condições | Realizar impressão de ficha de exame. |
| **Fluxo Principal** | |
| Ações do Ator | Ações do Sistema |
| 1. Solicitar pesquisa do paciente |  |
| 2. Inserir parte do nome do paciente |  |
|  | 3. Localizar paciente com parte do nome inserido |
| 4. Solicitar realização do exame |  |
|  | 5. Gerar ficha do exame |
| 6. Realizar impressão da ficha do exame |  |
| Restrições/validações | 1. O paciente deve apresentar um documento oficial com foto |
| **Fluxo Alternativo I – Paciente não possui cadastro** | |
|  | 1. Executar caso de uso Cadastrar paciente. |

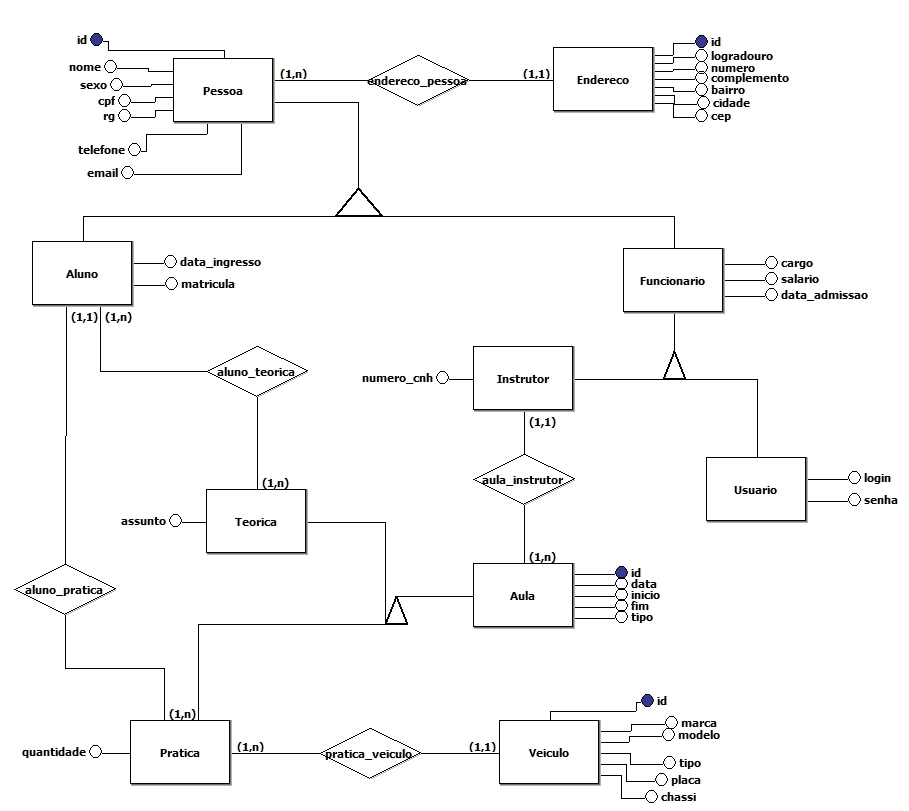
**3.4.4 DESCRIÇÃO CONCEITUAL**

É a descrição do mundo real, o domínio de negócios, que se pretende implementar. O mini mundo em questão é uma Autoescola, onde será cadastrado alunos, instrutores, secretárias, veículos, gerente(s), também deve cadastrar aulas teóricas com o instrutor(professor) que vai ministrá-la, assim como deve agendar aulas práticas para os alunos com o veículo que será usado e o instrutor que vai auxiliar na aula. Assim o banco deve armazenar os dados do domínio que abrange esse sistema e manter a integridade dos mesmos.

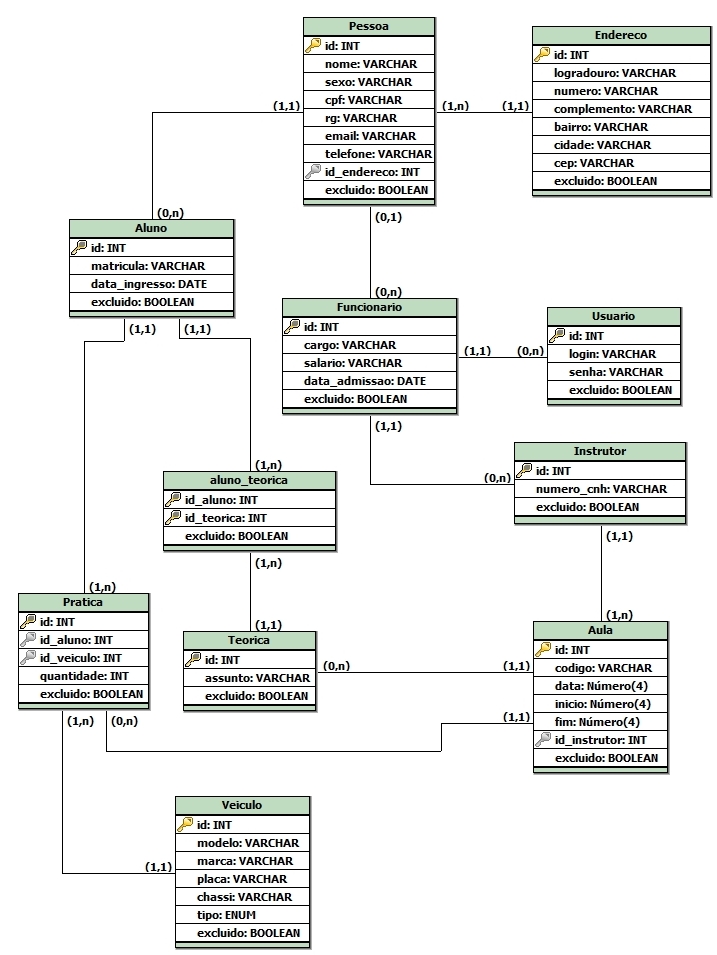
**3.4.5. LINGUAGENS E FERRAMENTAS**

O website foi desenvolvido utilizando a linguagem de programação, html5, PHP, CSS, JavaScript. Para a criação e manipulação do banco de dados foi utilizado o Phpadmin, e a linguagem SQL (Structured Query Language - Linguagem de Consulta Estruturada). A ferramenta utilizada foi o Wordpress.

**3.4.6. DIAGRAMA ENTIDADE-RELACIONAMENTO (DER)**



**3.4.7. MODELO RELACIONAL**



**3.4.8. CRIAÇÃO FÍSICA DO MODELO DE DADOS - SQL**

CREATE DATABASE projetobd1 DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

USE projetobd1;

CREATE TABLE admin (

id INT DEFAULT 1,

login VARCHAR(30) DEFAULT 'admin',

senha VARCHAR(255) DEFAULT 'admin',

CONSTRAINT pk\_admin PRIMARY KEY(id),

CONSTRAINT uk\_admin UNIQUE KEY(login)

)ENGINE=InnoDB;

INSERT INTO admin() VALUE();

CREATE TABLE endereco (

id INT NOT NULL AUTO\_INCREMENT,

logradouro VARCHAR(100) NOT NULL,

numero VARCHAR(10) DEFAULT "S/Nº",

bairro VARCHAR(100) NOT NULL,

complemento VARCHAR(100) DEFAULT NULL,

cidade VARCHAR(100) NOT NULL,

cep VARCHAR(9) NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_endereco PRIMARY KEY(id)

)ENGINE=InnoDB;

CREATE TABLE pessoa (

id INT NOT NULL AUTO\_INCREMENT,

nome VARCHAR(80) NOT NULL,

sexo ENUM('masculino', 'feminino') NOT NULL,

cpf varchar(11) NOT NULL,

rg varchar(11) DEFAULT NULL,

email varchar(80) DEFAULT NULL,

telefone VARCHAR(15) DEFAULT NULL,

id\_endereco INT DEFAULT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_pessoa PRIMARY KEY(id),

CONSTRAINT fk\_endereco\_pessoa FOREIGN KEY(id\_endereco) REFERENCES endereco(id),

CONSTRAINT uk\_pessoa UNIQUE KEY(cpf)

)ENGINE=InnoDB;

CREATE TABLE funcionario (

id INT NOT NULL,

cargo ENUM('gerente', 'instrutor', 'secretaria') NOT NULL,

salario DECIMAL NOT NULL,

data\_admissao DATE NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_funcionario PRIMARY KEY(id),

CONSTRAINT fk\_pessoa\_funcionario\_id FOREIGN KEY(id) REFERENCES pessoa(id)

)ENGINE=InnoDB;

CREATE TABLE usuario (

id INT NOT NULL,

login VARCHAR(30),

senha VARCHAR(255),

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_usuario PRIMARY KEY(id),

CONSTRAINT fk\_funcionario\_usuario\_id FOREIGN KEY(id) REFERENCES funcionario(id),

CONSTRAINT uk\_usuario UNIQUE(login)

)ENGINE=InnoDB;

CREATE TABLE veiculo (

id INT NOT NULL AUTO\_INCREMENT,

modelo VARCHAR(30) NOT NULL,

marca VARCHAR(30) NOT NULL,

placa VARCHAR(30) NOT NULL,

chassi VARCHAR(50) NOT NULl,

tipo Enum('carro', 'moto', 'onibus', 'caminhão') NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_veiculo\_id PRIMARY KEY(id),

CONSTRAINT uk\_veiculo\_placa UNIQUE KEY(placa),

CONSTRAINT uk\_veiculo\_chassi UNIQUE KEY(chassi)

)ENGINE=InnoDB;

CREATE TABLE aluno (

id INT NOT NULL,

matricula VARCHAR(10) NOT NULL,

data\_ingresso DATE,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_aluno PRIMARY KEY(id),

CONSTRAINT fk\_pessoa\_aluno\_id FOREIGN KEY(id) REFERENCES pessoa(id),

CONSTRAINT uk\_aluno UNIQUE KEY(matricula)

)ENGINE=InnoDB;

CREATE TABLE instrutor (

id INT NOT NULL,

numero\_cnh VARCHAR(30),

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_instrutor\_id PRIMARY KEY(id),

CONSTRAINT fk\_funcionario\_instrutor\_id FOREIGN KEY(id) REFERENCES funcionario(id),

CONSTRAINT uk\_instrutor UNIQUE KEY(numero\_cnh)

)ENGINE=InnoDB;

CREATE TABLE aula (

id INT NOT NULL AUTO\_INCREMENT,

tipo ENUM('pratica', 'teorica') NOT NULL,

`data` DATE NOT NULL,

inicio TIME NOT NULL,

fim TIME NOT NULL,

id\_instrutor INT NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_aula PRIMARY KEY(id),

CONSTRAINT fk\_instrutor\_aula FOREIGN KEY(id\_instrutor) REFERENCES instrutor(id)

)ENGINE=InnoDB;

CREATE TABLE teorica (

id INT NOT NULL,

assunto VARCHAR(40) NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_teorica PRIMARY KEY(id),

CONSTRAINT fk\_aula\_teorica FOREIGN KEY(id) REFERENCES aula(id)

)ENGINE=InnoDB;

CREATE TABLE pratica (

id INT NOT NULL,

id\_aluno INT NOT NULL,

id\_veiculo INT NOT NULL,

quantidade ENUM('1','2','3') NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_pratica PRIMARY KEY(id),

CONSTRAINT fk\_aula\_pratica FOREIGN KEY(id) REFERENCES aula(id),

CONSTRAINT fk\_veiculo\_pratica FOREIGN KEY(id\_veiculo) REFERENCES veiculo(id),

CONSTRAINT fk\_aluno\_pratica FOREIGN KEY(id\_aluno) REFERENCES aluno(id)

)ENGINE=InnoDB;

CREATE TABLE aluno\_teorica (

id\_aluno INT NOT NULL,

id\_teorica INT NOT NULL,

excluido BOOLEAN NOT NULL DEFAULT 0,

CONSTRAINT pk\_aluno\_aula\_teorica PRIMARY KEY(id\_aluno, id\_teorica),

CONSTRAINT fk\_aluno FOREIGN KEY(id\_aluno) REFERENCES aluno(id),

CONSTRAINT fk\_teorica FOREIGN KEY(id\_teorica) REFERENCES teorica(id)

)ENGINE=InnoDB;

DELIMITER $$

CREATE TRIGGER trg\_pessoa\_aluno\_verifica\_excluido AFTER UPDATE ON pessoa FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE aluno INNER JOIN pessoa SET aluno.excluido = pessoa.excluido WHERE aluno.id = pessoa.id AND aluno.excluido != pessoa.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_pessoa\_endereco\_verifica\_excluido AFTER UPDATE ON pessoa FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE endereco INNER JOIN pessoa SET endereco.excluido = pessoa.excluido WHERE endereco.id = pessoa.id\_endereco AND endereco.excluido != pessoa.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_pessoa\_funcionario\_verifica\_excluido AFTER UPDATE ON pessoa FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE funcionario INNER JOIN pessoa SET funcionario.excluido = pessoa.excluido WHERE funcionario.id = pessoa.id AND funcionario.excluido != pessoa.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_pessoa\_instrutor\_verifica\_excluido AFTER UPDATE ON pessoa FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE instrutor INNER JOIN pessoa SET instrutor.excluido = pessoa.excluido WHERE instrutor.id = pessoa.id AND instrutor.excluido != pessoa.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_pessoa\_usuario\_verifica\_excluido AFTER UPDATE ON pessoa FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE usuario INNER JOIN pessoa SET usuario.excluido = pessoa.excluido WHERE usuario.id = pessoa.id AND usuario.excluido != pessoa.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_pratica\_verifica\_excluido AFTER UPDATE ON aula FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE pratica INNER JOIN aula SET pratica.excluido = aula.excluido WHERE aula.id = pratica.id AND aula.excluido != pratica.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_teorica\_verifica\_excluido AFTER UPDATE ON aula FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE teoria INNER JOIN aula SET teoria.excluido = aula.excluido WHERE teorica.id = aula.id AND aula.excluido != teoria.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_instrutor\_aula\_verifica\_excluido AFTER UPDATE ON instrutor FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE aula INNER JOIN instrutor SET aula.excluido = instrutor.excluido WHERE aula.id\_instrutor = instrutor.id AND aula.excluido != instrutor.excluido;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE TRIGGER trg\_veiculo\_verfica\_excluido AFTER UPDATE ON veiculo FOR EACH ROW

BEGIN

SET SQL\_SAFE\_UPDATES = 0;

UPDATE pratica INNER JOIN veiculo SET pratica.excluido = veiculo.excluido WHERE pratica.id\_veiculo = veiculo.id AND pratica.excluido != veiculo;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE FUNCTION gera\_matricula(ano YEAR, param\_id INT) RETURNS VARCHAR(10)

BEGIN

IF param\_id < 10 THEN

RETURN concat(ano, '000',param\_id);

ELSEIF param\_id >= 10 AND param\_id <= 99 THEN

RETURN concat(ano, '00' , param\_id);

ELSEIF param\_id > 99 AND param\_id <= 999 THEN

RETURN concat(ano, '0' , param\_id);

ELSE RETURN concat(ano, param\_id);

END IF;

END $$

CREATE FUNCTION retorna\_id\_cpf(param\_cpf VARCHAR(11)) RETURNS INT

BEGIN

RETURN (SELECT pessoa.id FROM pessoa WHERE pessoa.cpf = param\_cpf);

END $$

CREATE FUNCTION retorna\_id\_matricula(param\_matricula VARCHAR(10)) RETURNS INT

BEGIN

RETURN (SELECT aluno.id FROM aluno WHERE aluno.matricula = param\_matricula);

END $$

CREATE PROCEDURE trg\_atualiza\_admin(param\_login VARCHAR(30), param\_senha VARCHAR(255))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE admin SET login = param\_login, senha = param\_senha;

END $$

CREATE PROCEDURE stp\_inserir\_pessoa(param\_nome VARCHAR(80), param\_sexo ENUM('masculino', 'feminino'), param\_cpf VARCHAR(11), param\_rg VARCHAR(11),

param\_email VARCHAR(80), param\_telefone VARCHAR(15))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO pessoa(nome, sexo, cpf, rg, email, telefone)

VALUES(param\_nome, param\_sexo, param\_cpf, param\_rg, param\_email, param\_telefone);

SET SQL\_SAFE\_UPDATES = 0;

UPDATE pessoa SET pessoa.id\_endereco = retorna\_id\_cpf(param\_cpf) WHERE pessoa.cpf = param\_cpf;

SET SQL\_SAFE\_UPDATES = 1;

END $$

CREATE PROCEDURE stp\_atualiza\_pessoa(param\_id INT, param\_nome VARCHAR(80), param\_sexo ENUM('masculino', 'feminino'), param\_email VARCHAR(80),

param\_telefone VARCHAR(18))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE pessoa SET nome = param\_nome, sexo = param\_sexo, email = param\_email, telefone = param\_telefone WHERE pessoa.id = param\_id;

END $$

CREATE PROCEDURE stp\_deleta\_pessoa(param\_id INT)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE pessoa SET pessoa.excluido = 1 WHERE pessoa.id = param\_id;

END $$

CREATE PROCEDURE stp\_inserir\_endereco(param\_logradouro VARCHAR(100), param\_numero VARCHAR(10), param\_bairro VARCHAR(30),

param\_complemento VARCHAR(100), param\_cidade VARCHAR(100), param\_cep VARCHAR(9))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO endereco(logradouro, numero, bairro, complemento, cidade, cep)

VALUES(param\_logradouro, param\_numero, param\_bairro, param\_complemento, param\_cidade, param\_cep);

END $$

CREATE PROCEDURE stp\_atuliza\_endereco(param\_id INT,param\_logradouro VARCHAR(100), param\_numero VARCHAR(10), param\_bairro VARCHAR(30),

param\_complemento VARCHAR(100), param\_cidade VARCHAR(100), param\_cep VARCHAR(9))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE endereco SET logradouro = param\_logradouro, numero = param\_numero, bairro = param\_bairro, complemento = param\_complemento,

cidade = param\_cidade, cep = param\_cep WHERE endereco.id = param\_id;

END $$

CREATE PROCEDURE stp\_inserir\_aluno(param\_cpf VARCHAR(11))

BEGIN

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

INSERT INTO aluno(id, matricula, data\_ingresso) VALUES(retorna\_id\_cpf(param\_cpf), gera\_matricula(year(now()), retorna\_id\_cpf(param\_cpf)), utc\_date());

COMMIT;

END $$

CREATE PROCEDURE stp\_inserir\_funcionario(param\_cpf VARCHAR(11), param\_cargo ENUM('gerente', 'instrutor', 'secretaria'), param\_salario DECIMAL)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO funcionario(id, cargo, salario, data\_admissao) VALUES(retorna\_id\_cpf(param\_cpf), param\_cargo, param\_salario, utc\_date());

END $$

CREATE PROCEDURE stp\_inserir\_usuario(param\_cpf VARCHAR(11), param\_login VARCHAR(30), param\_senha VARCHAR(255))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO usuario(id, login, senha) VALUES(retorna\_id\_cpf(param\_cpf), param\_login, param\_senha);

END $$

CREATE PROCEDURE stp\_inserir\_instrutor(param\_cpf VARCHAR(11), param\_numero\_cnh VARCHAR(30))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO instrutor(id, numero\_cnh) VALUES (retorna\_id\_cpf(param\_cpf), param\_numero\_cnh);

END $$

CREATE PROCEDURE stp\_atualiza\_salario(param\_id INT, param\_salario DECIMAL)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE funcionario SET funcionario.salario = param\_salario WHERE funcionario.id = param\_id;

END $$

CREATE PROCEDURE stp\_atualiza\_senha(param\_login VARCHAR(30), param\_senha VARCHAR(255))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE usuario SET senha = param\_senha WHERE usuario.login = param\_login;

END $$

CREATE PROCEDURE stp\_inserir\_teorica(param\_data DATE, param\_inicio TIME, param\_fim TIME, param\_id\_instrutor INT, param\_assunto VARCHAR(30))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO aula(tipo, `data`, inicio, fim, id\_instrutor) VALUES('teorica', param\_data, param\_inicio, param\_fim, param\_id\_instrutor);

INSERT INTO teorica(id,assunto) VALUES(last\_insert\_id(), param\_Assunto);

END $$

CREATE PROCEDURE stp\_atualiza\_teorica(param\_id INT, param\_data DATE, param\_inicio TIME, param\_fim TIME, param\_id\_instrutor INT, param\_assunto VARCHAR(30))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE aula SET `data` = param\_data, inicio = param\_inicio, fim = param\_fim, id\_instrutor = param\_id\_instrutor WHERE aula.id = param\_id;

UPDATE teoria SET assunto = param\_assunto WHERE teorica.id = param\_id;

END $$

CREATE PROCEDURE stp\_inserir\_pratica(param\_data DATE, param\_inicio TIME, param\_fim TIME, param\_quantidade ENUM('1', '2', '3'),param\_id\_instrutor INT, param\_id\_aluno INT, param\_id\_veiculo INT)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO aula(tipo, `data`, inicio, fim, id\_instrutor) VALUES('pratica', param\_data, param\_inicio, param\_fim, param\_id\_instrutor);

INSERT INTO pratica(id, id\_aluno, id\_veiculo, quantidade) VALUES(last\_insert\_id(), param\_id\_aluno, param\_id\_veiculo, param\_quantidade);

END $$

CREATE PROCEDURE stp\_atualiza\_pratica(param\_id INT, param\_data DATE, param\_inicio TIME, param\_fim TIME, param\_id\_instrutor INT, param\_id\_veiculo INT)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE aula SET `data` = param\_data, inicio = param\_inicio, fim = param\_fim, id\_instrutor = param\_id\_instrutor WHERE aula.id = param\_id;

UPDATE pratica SET id\_veiculo = param\_id\_veiculo WHERE partica.id = param\_id;

END $$

CREATE PROCEDURE stp\_deleta\_aula(param\_id INT)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE aula SET aula.exluido = 1 WHERE aula.id = param\_id;

END $$

CREATE PROCEDURE stp\_inserir\_aluno\_teorica(param\_id\_aluno INT, param\_id\_aula\_teorica INT)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO aluno\_aula(id\_aluno, id\_aula) VALUES(param\_id\_aluno, param\_id\_aula\_teorica);

END $$

CREATE PROCEDURE stp\_atualiza\_aluno\_aula\_teorica(param\_id\_aula\_teorica INT)

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE aluno\_teorica SET id\_aula\_teorica = param\_id\_aula\_teorica;

END $$

CREATE PROCEDURE stp\_inserir\_veiculo(param\_modelo VARCHAR(30), param\_marca VARCHAR(30), param\_placa VARCHAR(30), param\_chassi VARCHAR(50),

param\_tipo\_veiculo Enum('carro', 'moto', 'onibus', 'caminhão'))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

INSERT INTO veiculo(modelo, marca, placa, chassi, tipo) VALUES(param\_modelo, param\_marca, param\_placa, param\_chassi, param\_tipo\_veiculo);

END $$

CREATE PROCEDURE stp\_autaliza\_veiculo(param\_id INT, param\_modelo VARCHAR(30), param\_marca VARCHAR(30), param\_placa VARCHAR(30), param\_chassi VARCHAR(30),

param\_tipo\_veiculo Enum('carro', 'moto', 'onibus', 'caminhão'))

BEGIN

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE veiculo SET modelo = param\_modelo, marca = param\_marca, placa = param\_placa,

chassi = param\_chassi, tipo = param\_tipo WHERE veiculo.id = param\_id;

END $$

CREATE PROCEDURE stp\_deleta\_veiculo(param\_id INT)

BEGIN

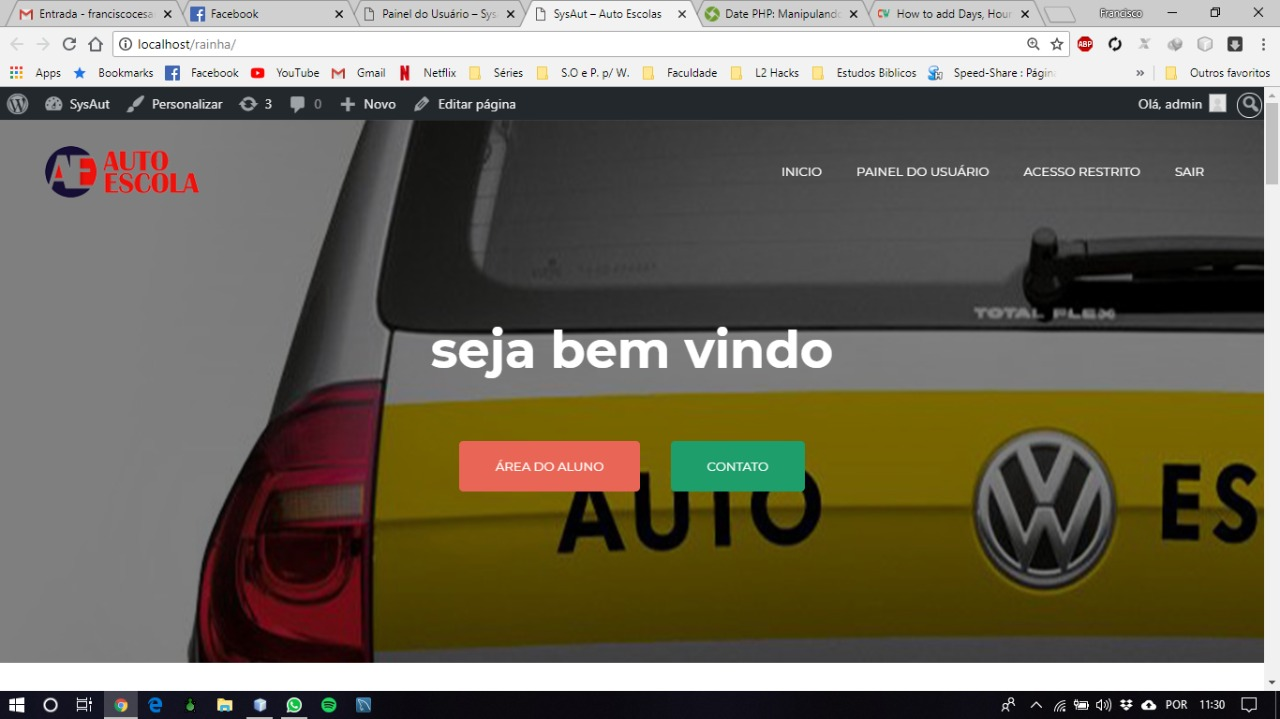
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

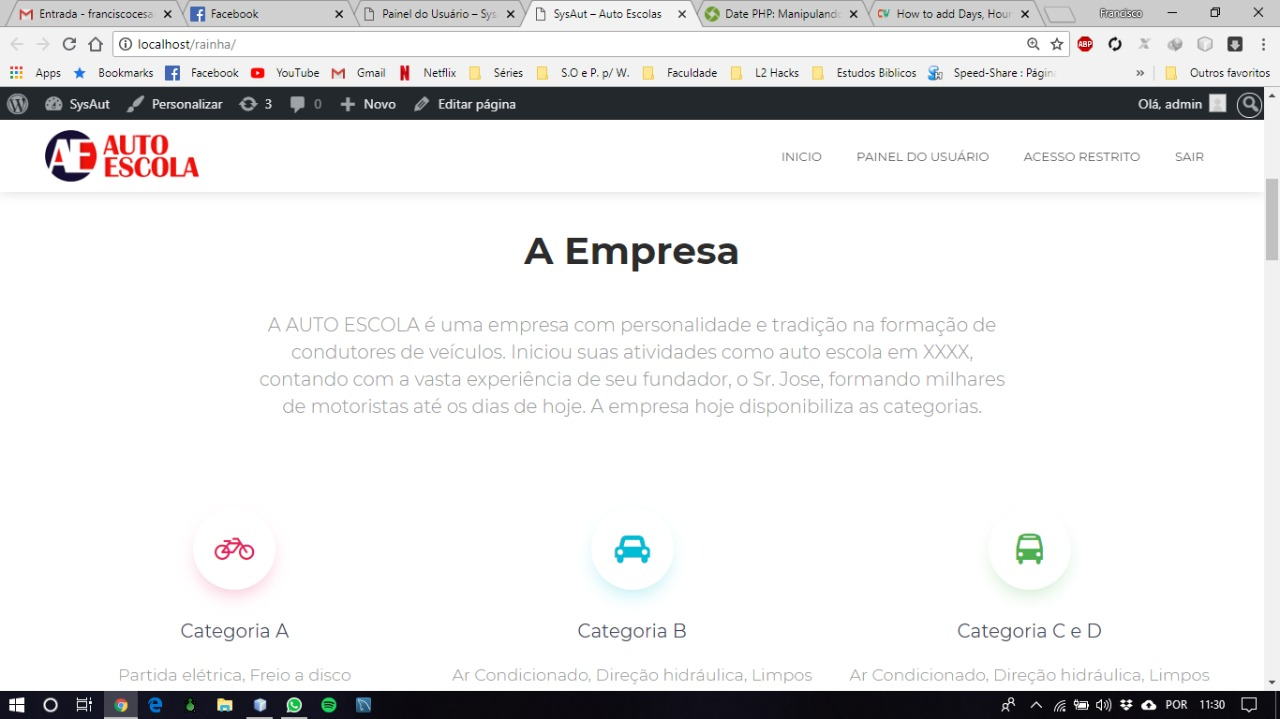
UPDATE veiculo SET veiculo.excluido = 1 WHERE veiculo.id = param\_id;

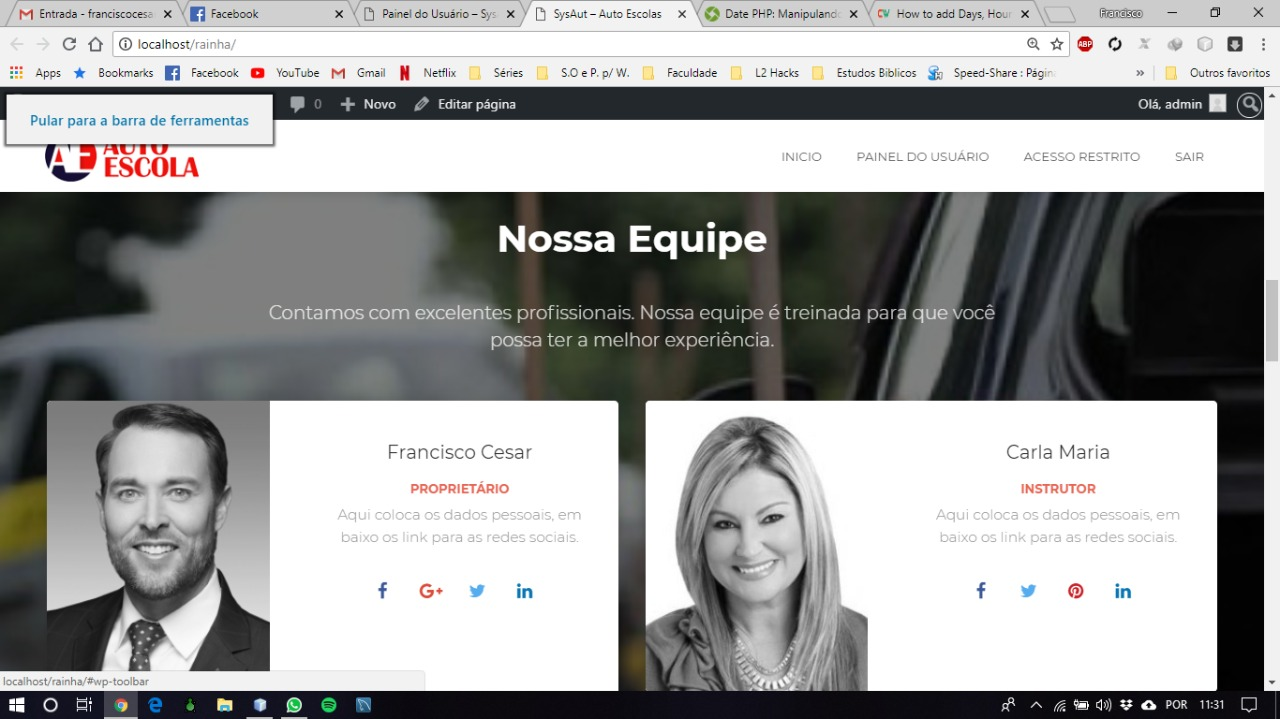
END $$

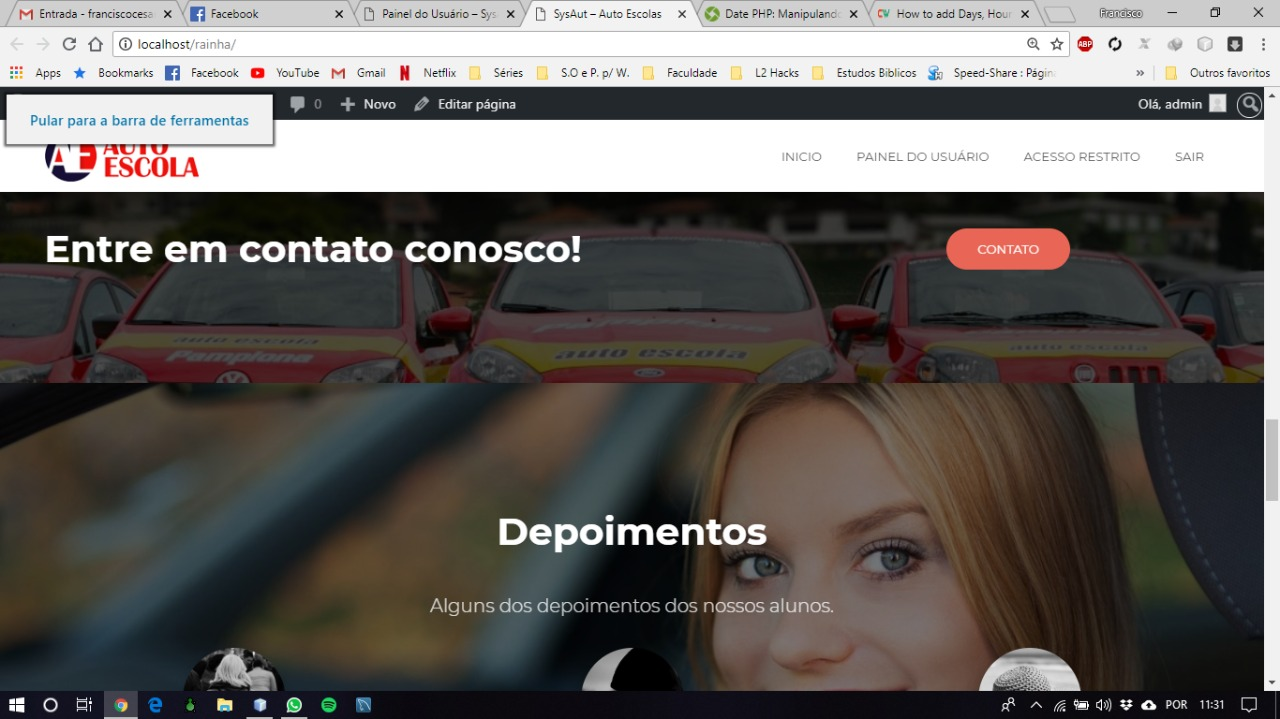
DELIMITER ;

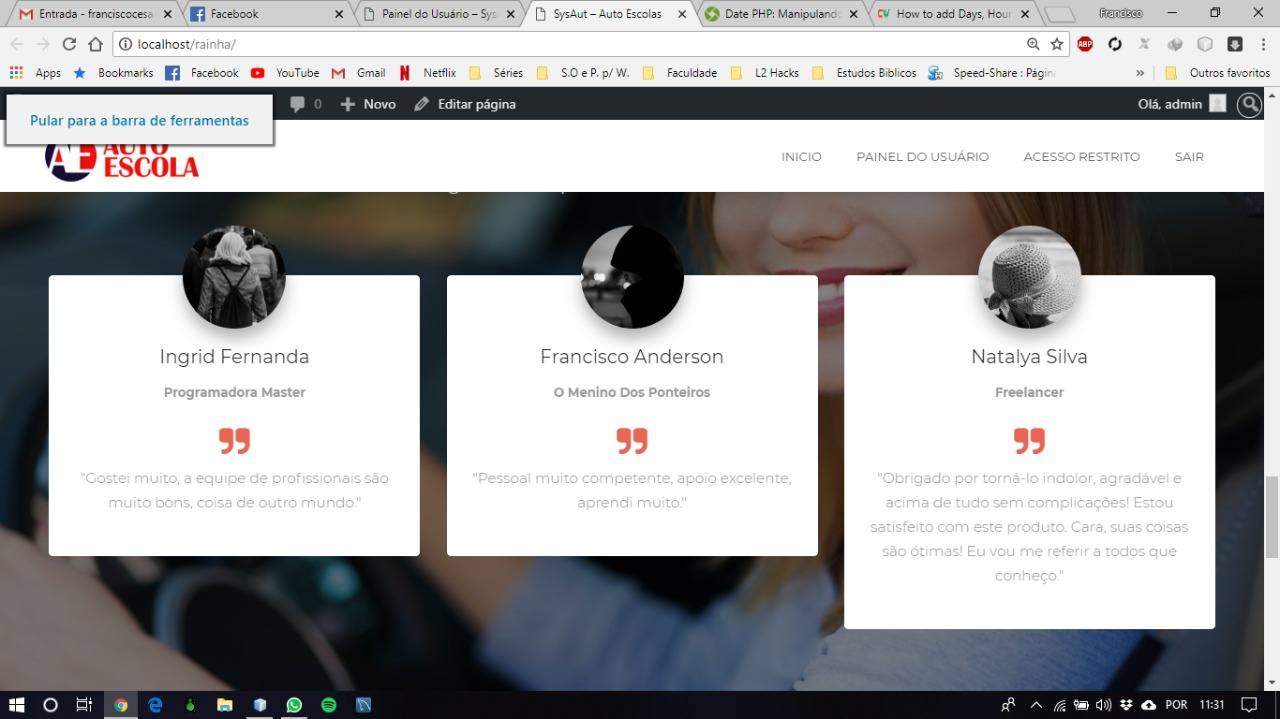
**3.4.9. TELAS**

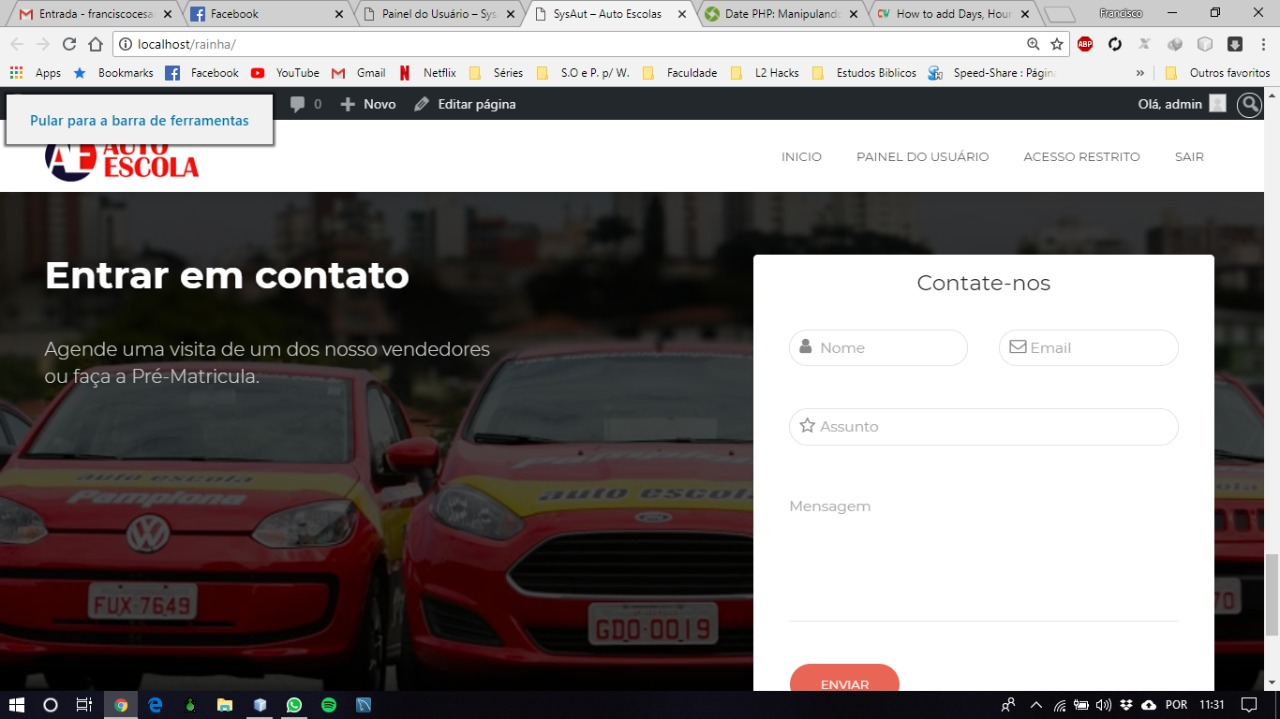
****

****

****

****

****

****

**4. CONCLUSÃO**

Após necessitar do apoio especializado de empresa de Auto Escola na cidade de Sousa-PB, percebeu-se que a maioria das empresas desse segmento não conta com um Website que controle e agenda de aulas. O Website é capaz de controlar todas as aulas dos alunos e listar os horárias para os respectivos instrutores. Bem como, aumentar o controle e a organização da empresa.

**5. REFERÊNCIAS**

ELMASRI, R.; NAVATHE, S. Sistemas de Bancos de Dados. 6a ed. Addison-

Wesley. 2013.

SILBERSCHATZ, A.; KORTH, H. F..; SUDARSHAN, S. Sistema de Banco de Dados.

6a ed. Elsevier-Campus. 2012.

MACHADO, F. N. R.; ABREU, M. P. Projeto de Banco de Dados - Uma Visão

Prática. 17a ed. Érica. 2012