Instituto Federal Catarinense
Ciências da computação
Campus Videira

ALUGUEL DE CARROS

Carros oficiais para servidores do campus

Crislley Borga dos Santos; Roan Poggere da Silva

INTRODUÇÃO

Este relatório tem como objetivo a explicação do projeto feito para o aluguel de carros oficiais do Instituto Federal Catarinense, Campus Videira, o projeto foi desenvolvido na linguagem de programação web, PHP, para melhor funcionamento do programa foi adicionado banco de dados(MySql) e o padrão de projeto utilizado foi o MVC(MODEL, VIEW e CONTROLLER). O servidor que usamos é localhost
No início do programa é necessário fazer o cadastro ou se já for cadastrado fazer o login, inserindo seu login e senha.

RESULTADOS E DISCUSSÃO

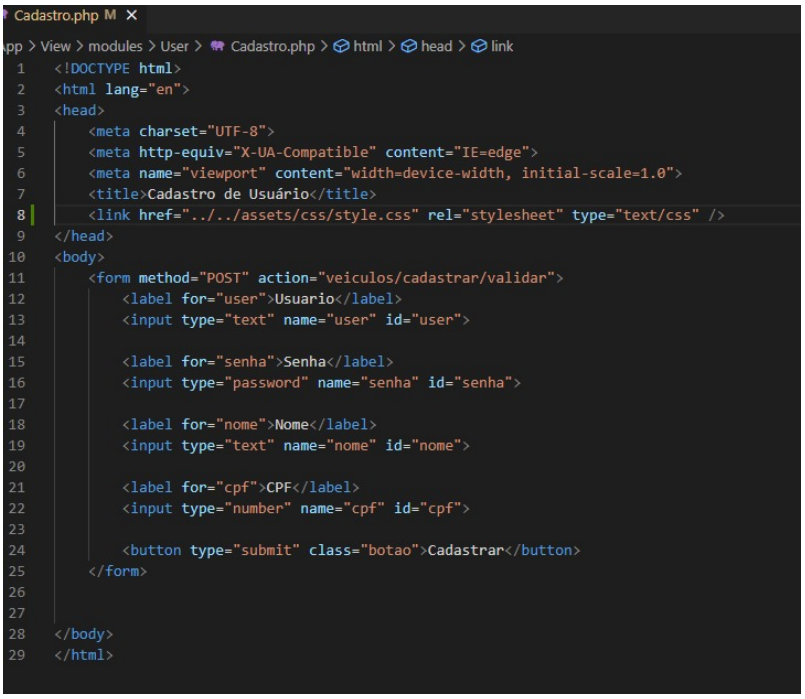
Este projeto é de suma importância, pois se utilizado da maneira correta é dispensado o uso de outras maneiras de fazer o “aluguel” dos carros oficiais, livrando os usuários e servidores de uma menor burocracia e gasto desnecessário de outros materiais(folhas, impressões, etc), visto que com a elaboração do projeto ficará salvo em banco de dados e o mesmo terá a função de listar todos os "aluguéis" realizados e o servidor que alugou o mesmo.

MVC

O padrão de projeto MVC, é muito utilizado pois sua arquitetura possui a divisão do projeto em camadas, as camadas são muito bem definidas na qual cada uma tem seu papel específico, esse padrão facilita na hora da programação pois cada coisa tem a sua função

Classes View

Nas classes view é onde estão armazenados frames de códigos html e CSS, as mesmas são necessárias e importantíssimas para que o usuário possa enxergar o que está acontecendo e não ficar somente uma tela em preto, ela mostra aos usuários dados. Os arquivos da classe view conseguem acessar os dados da model, mas não consegue manipulá-los.



Neste modelo acima é a classe do Cadastro onde se tem um formulário de cadastro

Index.php

No index.php de acordo com a rota que o usuário está chamando vai definir o controller

index.php

App > index.php > ...

```php
<?php
session_start();
include 'Controller/UserController.php';
include 'Controller/VeiculosController.php';
include 'Controller/ServidoresController.php';
include 'Controller/ReservasController.php';

$url = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

switch($url){
    case '/':
        UserController::inicial();
    break;
    case '/login':
        UserController::logar();
    break;
    case '/login/validar':
        UserController::validarLogin();
    break;
    case '/cadastrar':
        UserController::cadastrar();
    break;
    case '/cadastrar/validar':
        UserController::validarCadastro();
    break;
    case '/inicio':
        UserController::inicial();
    break;
    case '/sair':
        UserController::sair();
    break;

    case '/veiculos':
        VeiculosController::listar();
    break;
    case '/veiculos/cadastrar':
        VeiculosController::cadastrar();
    break;
    case '/veiculos/cadastrar/validar':
        VeiculosController::validarCadastro();
    break;

    case '/servidores':
        ServidoresController::listar();
    break;
    case '/servidores/cadastrar':
        ServidoresController::cadastrar();
    break;
```

Classes Model

Nas classes model é realizado uma ação em conjunto com as classes da DAO que é realizado o CRUD e a manipulação com o banco de dados do projeto. Neste exemplo abaixo temos a classe ServidoresModel que uma das funções é chamar a função insertServidor da classe ServidoresDao.

ServidoresModel.php

App > Model > ServidoresModel.php > ServidoresModel

```php
<?php
class ServidoresModel{

    public $nome, $cpf;
    public $rows;

    public function salvarServidor(){
        include 'DAO/ServidoresDAO.php';

        $dao = new ServidoresDAO();

        if(empty($this->id))
        {
            $dao->insertServidor($this);
        }
    }

    public function getAllRows(){
        include 'DAO/ServidoresDAO.php';

        $dao = new ServidoresDAO();

        $this->rows = $dao->selectServidores();
    }
}
?>
```

Classes DAO

Nestas classes como dito no parágrafo anterior ela trabalha em ação conjunta com o model, pois o CRUD fica nessas classes DAO e o model chama os métodos e funções. Nesta classe temos a conexão com o banco de dados e as informações inseridas no mesmo.

Classes Controller

Nas classes controller como o próprio nome já diz é realizado o controle do projeto como por exemplo, ao iniciar o projeto vai abrir a aba de login, e após isso se você não tiver cadastro irá para o cadastro ou verificação de login, com isso ele é responsável por “guiar” o usuário entre as pastas, nesta foto abaixo vemos um exemplo de como fica uma classe controller, no modelo é a servidoresController

ServidoresController.php

App > Controller > ServidoresController.php > ServidoresController > validarCadastro

```php
<?php

class ServidoresController{
    public static function cadastrar(){

        include 'View/Modules/Servidores/CadastroServidores.php';
    }

    public static function validarCadastro(){
        include 'Model/ServidoresModel.php';

        $model = new ServidoresModel();

        $model->nome = $_POST['nome'];
        $model->cpf = $_POST['cpf'];

        $model->salvarServidor();

        include 'View/Modules/User/Inicio.php';
    }

    public static function listar(){

        include "Model/ServidoresModel.php";

        $model = new ServidoresModel();
        $model->getAllRows();

        include 'View/Modules/Servidores/ListarServidores.php';
    }

}

?>
```

BANCO DE DADOS

Para fazer a conexão com o banco de dados é necessário usar o banco de dados MySql e rodar esse script :

Esta tabela é responsável pelos usuários.

```
CREATE TABLE `user` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `senha` VARCHAR(45) NOT NULL,
  `login` VARCHAR(45) NOT NULL,
 `cpf` char(11) DEFAULT NULL,
  PRIMARY KEY (`id`));
)
```

Esta tabela ‘servidor’ seria os funcionários que farão o “aluguel” do carro.

```
CREATE TABLE `servidores` (
  `id` int NOT NULL AUTO_INCREMENT,
  `nome` varchar(45) DEFAULT NULL,
  `cpf` char(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

Esta tabela é responsável por definir os veículos e suas características.

```
CREATE TABLE `veiculos` (
  `id` int NOT NULL AUTO_INCREMENT,
  `placa` varchar(7) NOT NULL,
  `modelo` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

Esta tabela é responsável por fazer a reserva do carro, tanto como obter as informações do servidor que vai utilizar e as informações do veículo (idVeiculo, id Servidor).

```
CREATE TABLE `reservas` (
  `id` int NOT NULL AUTO_INCREMENT,
  `data` date NOT NULL,
  `descricao` varchar(45) NOT NULL,
  `idServidor` int NOT NULL,
  `idVeiculo` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `idVeiculo_idx` (`idVeiculo`),
  KEY `idServidor_idx` (`idServidor`),
  CONSTRAINT `idServidor` FOREIGN KEY (`idServidor`) REFERENCES `servidor` (`id`),
  CONSTRAINT `idVeiculo` FOREIGN KEY (`idVeiculo`) REFERENCES `veiculos` (`id`)
)
```

Obs → Não conseguimos fazer com que a classe reservar Veículos funcione corretamente, pois a mesma tem chave estrangeira e não conseguimos realizar a programação da mesma.