

# SIMULADOR DE SITE DE ALUGUEL POR TEMPORADA

---

## Introdução

---

A partir dos conhecimentos em programação orientada a objeto, utilizando os princípios de classe, objeto, herança, polimorfismo, juntamente com os conhecimentos da linguagem de marcação HTML/CSS para interface do site, foi implementado um site de reserva de imóveis, por temporada, que simula sistemas reais como trivago, Airbnb, Tripadvisor, etc.

A linguagem de programação utilizada: PHP (*Hypertext Preprocessor*).

## Da motivação

Este projeto foi proposto como composição da segunda nota da disciplina de Linguagem de Programação II, pela professora Maria Inés restovic, referente ao 3ª semestre do curso de [Sistemas de Informação](#) na [Universidade do Estado da Bahia \(UNEB\)](#).

## Do limite

---

O projeto não possui uma relação unitária e exclusiva do imóvel (objeto) para o local da reserva, onde o imóvel só poderá conter 1(um) local de reserva e o local de reserva poderá conter N imóveis.

Isso deve-se ao fato de ainda não haver um armazenamento dos imóveis intrínseco ao local da reserva. Entretanto, em projetos futuros, pretende-se implementar essa forma de armazenamento a fim de aproximar o projeto de um sistema de aluguel de imóvel real.

## Instruções de uso

---

A elaboração do projeto parte com base em informações solicitadas ao usuários no index (*página 1*) como, quantidade de hóspedes, local da reserva, data de check-in e data de check-out. A partir desses dados informados pelo usuário, cabe ao sistema mostrar no index principal (*página 2*) todos os imóveis disponíveis, com seus respectivos valores, nomes e um botão de reservar. Escolhido o imóvel,

no botão de reservar, o sistema parte para o detalhe do produto (*página 3*), responsável por mostrar todos os atributos do imóvel, a opção alterar check-in, caso o usuário encontre um erro ou deseje alterar as informações fornecidas no index(*página 1*), e o valor total no qual ele deverá pagar ,pelo imóvel, sendo este valor o resultado da operação check-out subtraído pelo check-in, multiplicado pelo valor da diária do imóvel.

Caso o sistema não encontre erro no número de hóspedes, local da reserva, e nas datas de check-in/check-out, fornecidos pelo usuário no index(*página 1*) ou no momento da alteração do check-In, o sistema permite a execução do botão “CONFIRMAR”, no qual encarrega-se de verificar se o imóvel já encontra-se reservado nas data de check-In e check-Out solicitados. Caso o imóvel esteja disponível o sistema trata de validar a reserva do imóvel e armazenar a data para impossibilitar futuras reservas deste imóvel nessas mesmas datas, caso contrário, o sistema trata de informar que o imóvel está indisponível na data escolhida e não valida a reserva.

## Análise do Sistema

---

### Classes

Por trás da interface do sistema, foram criadas classes para modularizar, facilitar a manutenção do código, instanciar e proteger o código e os dados recebidos a partir dos pilares da programação orientada a objeto: herança, polimorfismo, encapsulamento.(ver UML)

As classes criadas foram, form, Imóvel, Casa, apartamento, Casa de campo e Hotel, onde ocorre um nível de herança entre as classes. Todos os atributos das classes são protegidos, onde só podem ser utilizado através de métodos .

A classe Form é responsável por receber os dados do formulário e tratá-los. Quando recebe uma data de um formulário, o mesmo vem no formato (Ano-mês-dia), como se trata de um sistema Brasileiro, faz necessário alterar o formato para (Dia-mês-ano), para isso, os atributos #checkin:char, #checkout:char, foram instanciados com a classe **Datetime** para poder utilizar o método **format('d-m-Y')**. Ainda na classe form, o método **Getdias( )** foi utilizado para validação do formulário, onde a partir da data de #checkin:char e #checkout:char faz a operação de diferença entre as datas devolvendo o total de dias que usuário ficaria. Caso o #checkout:char fosse maior que o #checkin:char, o retorno de dias é

0(zero) inviabilizando a validação da reserva. Assim como a data do `#checkin:char` fosse menor que a data do dia atual recebida pela classe **date**.

A classe abstrata Imóvel herda a classe form. A classe Imóvel é responsável pela base dos objetos seguinte, pois ele representa “ o molde para um molde” A classe possui os métodos abstratos **+exibirImovel( )**, **+reservar( )**, na qual devem ser implementados nas classes que herdam a classe imóvel. Entretanto, os métodos **+verificarReserva( )** e o método **+Imagem( )** já estão implementados na classe o que facilita quando a classe for herdada por outra.

A classe Casa herda a classe abstrata imóvel. A classe casa é responsável por instanciar os atributos e os métodos da classe Imóvel, acrescentando alguns outros e implementando os métodos abstratos **+exibirImovel( )**, **+reservar( )**.

A classe Apartamento também herda a classe abstrata imóvel. Também é responsável por instanciar os atributos e os métodos e acrescenta alguns outros. Também implementa os métodos abstratos **+exibirImovel( )**, **+reservar( )**.

Por fim, as classes casa de campo e hotel, herdam respectivamente a classe Casa e classe Hotel, também implementa os métodos abstratos da classe imóvel e os métodos abstratos **+exibirImovel( )**, **+reservar( )**.

Um dos motivos para os métodos **+exibirImovel( )**, **+reservar( )** já não serem implementados na classe imóvel, igualmente ao método **+VerificarReservar( )** e **+Imagem( )**, é que mesmo com o conceito de polimorfismo, os métodos possuem tags `<form></form>` que abrem e fecham no local onde faz -se necessário a alteração e incremento do método, o que a princípio dificultou e fez com que o método fosse colocado como abstrato na classe Imóvel, forçando a todas classe que a herdasse e subjacentes, implementar o método. Uma outra forma de que poderia ser utilizada para forçar a implementação dos métodos seria utilizando interface, entretanto optou-se por não utilizar

## Descrição dos métodos

---

**+verificarReserva( )**: responsável pela leitura e escrita do arquivo de reservas que informa se o imóvel pode ser reservado na data solicitada, se sim, valida a reserva e salva no arquivo a data da reserva, caso contrário não valida a reserva e informa as datas que o imóvel já encontra-se reservado.

**+Imagem( ):** responsável por mostrar a imagem do objeto e seus atributos com seus respectivos valores no detalhe do produto (página 3). A depender do resultado do método +verificarReserva( ), mostra as datas que o imóvel já está reservado.

**+exibirImovel( ):** responsável por exibir os imóveis no index principal (página 2) mostrando o valor do objeto, sua imagem e a opção de reservar.

**+reservar( ):** Responsável pela possibilidade de alteração do check-In (página 3) e validação dos dados fornecidos. Se a quantidade de dias for igual a 0 (zero) e a quantidade de hóspedes for igual ou menor a 0 (zero) e a data do check-In for menor que a data do Check-out ou menor do que a data atual, o sistema não valida a confirmação da reserva e informa o erro da validação.

## ARQUIVO

Uma solução para otimizar a instanciamento dos objetos foi a utilização de arquivo. A utilização do arquivo de maneira dinâmica torna-se viável na medida que após criado e bem estruturado, torna-se mais seguro e eficaz instanciar objetos, e criar novos objetos, isso quando comparado a instanciamento estática. Para isso fez-se necessário de um arquivo texto básico, no formato txt, onde a depender da necessidade se ler e/ou escreve no arquivo os atributos do objeto. Para isso fez-se necessário também uma estrutura de repetição para o processo sequencial de leitura e/ou escrita do atributo.

Para cada tipo de classe instanciável, Casa, Apartamento, Casa campo, Hotel, foi utilizado o atributo tipo para criar um arquivo separado, onde o valor 1 está condicionado a criação do objeto tipo casa, valor 2 ao objeto tipo apartamento, valor 3 ao objeto tipo casa campo, valor 4 ao objeto tipo Hotel. Caso haja a necessidade de criação de uma nova classe basta condicionar o valor a instância do objeto e na leitura do arquivo os objetos do arquivo serão passados como valor.

Uma outra utilização recorrente do arquivo foi a leitura das datas dos imóveis reservados, onde o método utilizado consiste basicamente em ler o arquivo de reservas, verificar se a data pertence ao imóvel, se pertencer, informa a indisponibilidade do imóvel e as datas que já se encontra reservado, caso a data não pertença, reserve o imóvel e escreva a nova data no arquivo para futuras reservas deste mesmo imóvel.

Doravante, vale a ressalva que a utilização de banco de dados é muito mais eficiente e simples do que arquivo, visto que o banco de dados já possui métodos já prontos onde facilita o uso e a manipulação dos dados sem a perda dos mesmo ao

fim a execução do sistema, ou o envio dos dados que serão utilizados na página seguinte.

## HTML

A utilização do HTML consistiu em um template baixado na internet, de maneira gratuita, na qual possui bootstrap e foi modificado de acordo com as necessidades do projeto. O uso do HTML faz necessário visto que cria-se uma interface amigável com o usuário já que não é o objetivo final do PHP criar interface.

Além de utilizar a linguagem de marcação para a interface do sistema, na qual foi o seu principal objetivo, o HTML foi essencial para a transferência de dados entre página. Mesmo com a utilização de arquivo, no qual seu intuito é salvar os dados para que não haja perda no fim da execução, ainda assim houve uma problemática bastante comum no trabalho: Enviar os dados de uma página para outra de maneira rápida. A princípio uma maneira de se fazer isso é com a utilização de arquivo, entretanto, torna-se custoso e a depender do tamanho do arquivo, o tempo de busca cresceria.

A forma utilizada para contornar essa situação foi a utilização da tag **<form action = “Destino de envio.php” method= “POST”>** e com o **<input type= “hidden” name= “campo” id= “campo” value= “valor enviado”>**. Isso fica evidenciado no método `+exibirImoveis()` onde além de exibir o imóvel, caso seja clicado em reservar, os atributos que ali estão naquele objeto, presente naquela página, será enviado para a página destino, e na página destino, faz necessário um método `&_POST[“campo”]` para receber o valor enviado do input da página anterior.

O tipo Hidden é responsável por enviar valores de maneira na qual o usuário não veja, o que facilita o envio de um objeto de uma página para outra. entretanto o valor só pode ser recebido pela variável global POST.

## EQUIPE

---

Danilo Nascimento  
José Cleiton