



CENTRO UNIVERSITÁRIO UNIVEL

**GUSTAVO PERUZZO**

**MATHEUS BUTKOSKI**

**AS - ATIVIDADE SUPERVISIONADA**

CASCADEL - PR

2022

## **INTRODUÇÃO**

Neste trabalho apresentaremos um modelo de documentação de um sistema de leilões, conseqüentemente seus atributos de qualidade, modelos de arquitetura, restrições, e implementações em diferentes padrões de projeto que podem ser utilizados na criação de um sistema real.

## **OBJETIVOS DO NEGÓCIO/PROJETO**

O objetivo do projeto é expandir exponencialmente a quantidade de leilões e atingir o máximo de usuários possíveis, para isso será realizado leilões em uma plataforma online para que seus usuários possam utilizar de qualquer lugar.

## **ATRIBUTOS DE QUALIDADE**

Segurança(Security): O projeto deverá ter o menor nível de risco possível, pois estará lidando com sistemas de pagamentos e dados dos usuários.

Disponibilidade(Availability): O software deverá estar disponível para utilização sempre que o usuário precisar.

Escalabilidade(Scalability) e elasticidade(Elasticity): O software deve realizar o maior número de leilões simultâneos possível, para isso ele deverá atender a um número grande de requisições sem uma perda significativa de desempenho.

Desempenho(Performance): Os leilões devem ser o mais próximo do tempo real, isso requer que as tarefas sejam executadas rapidamente.

## **FUNÇÃO PRINCIPAL**

O software deverá prover uma plataforma onde os leiloeiros e os usuários poderão utilizar com facilidade, com uma interface amigável, o leiloeiro precisará visualizar e controlar, em tempo real, os lances que os usuários farão.

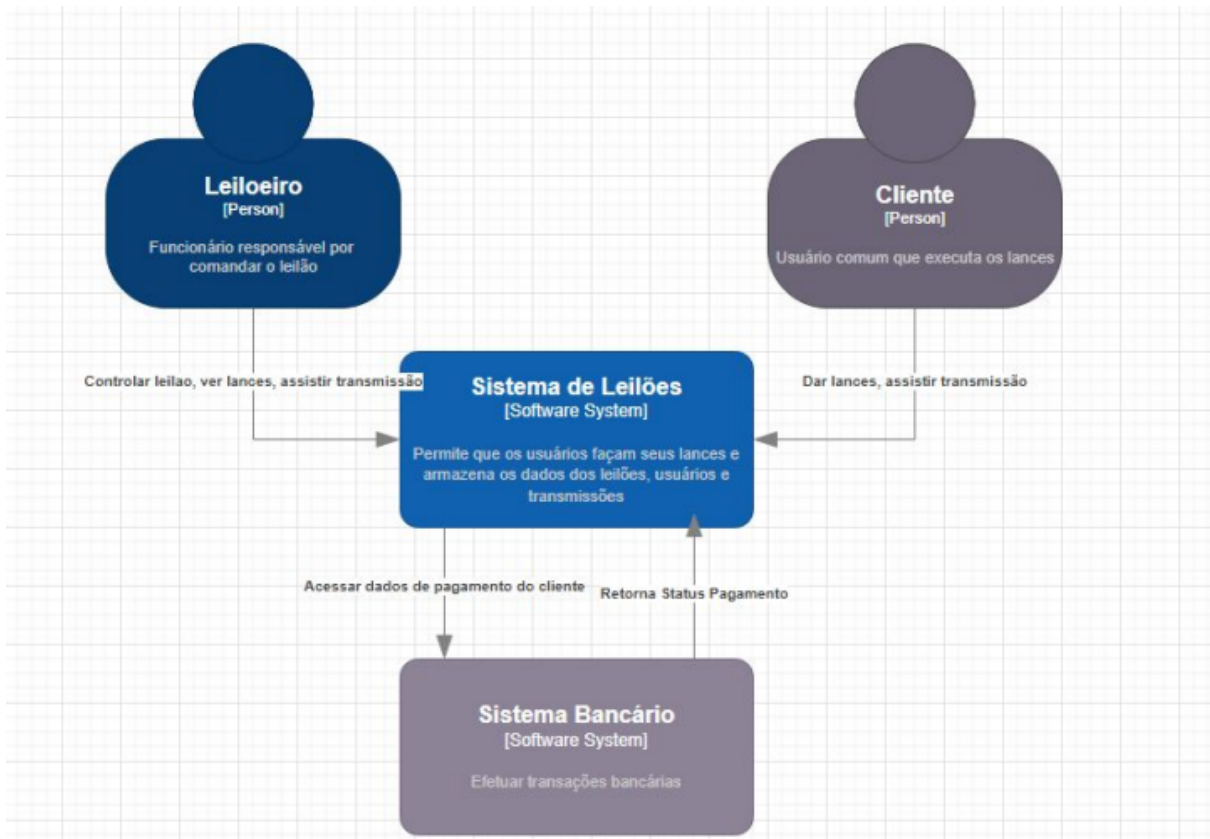
## **PREOCUPAÇÕES ARQUITETURAIS**

- O software deverá conter uma boa integração com as empresas fundidas, para isso o software deve disponibilizar um protocolo de comunicação.
- Os lances deverão acontecer em tempo real e na ordem correta que foram efetuados.
- A plataforma deverá efetuar uma transmissão ao vivo dos leilões para que os usuários possam acompanhar e dar lances em tempo real.
- O software deverá conter um bom mecanismo anti-fraude, pois conterà dados importantes dos usuários cadastrados.

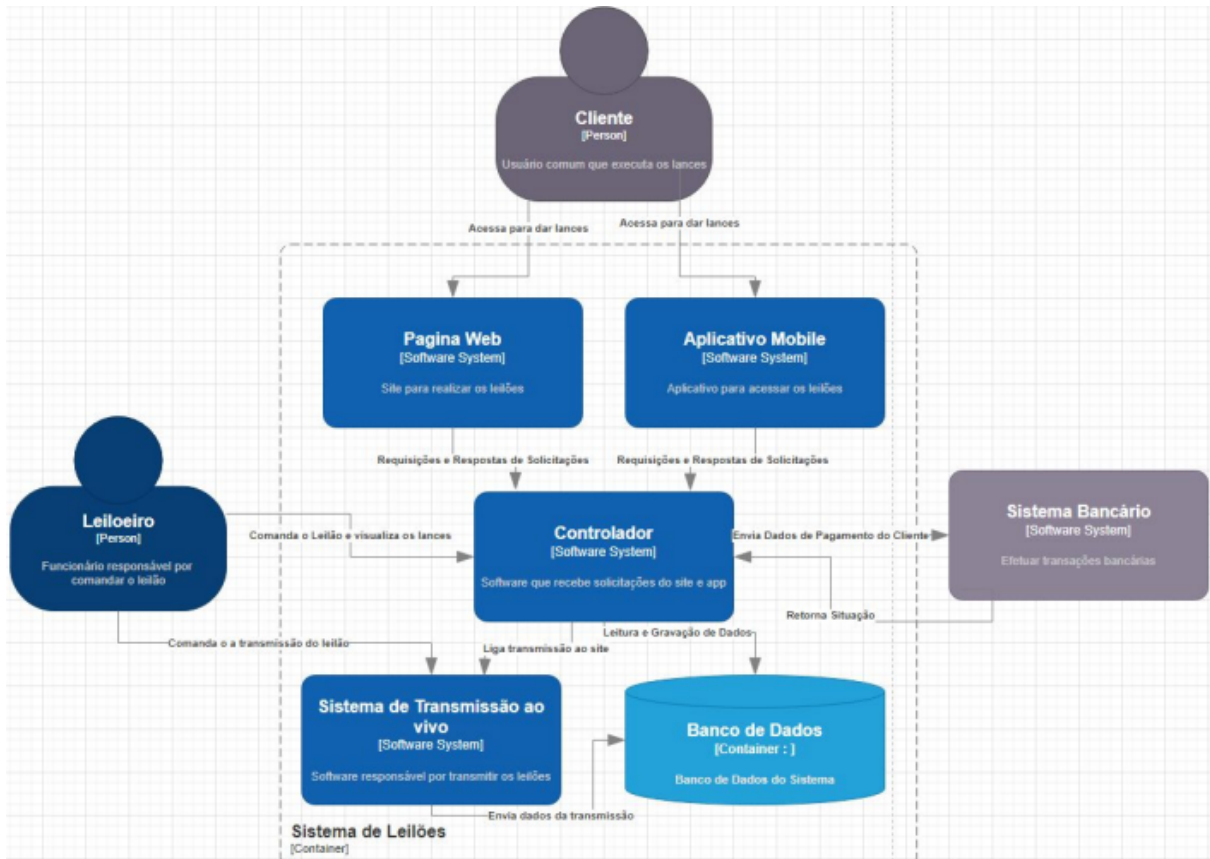
## **RESTRIÇÕES**

- O software deverá suportar o maior número de plataformas operacionais possíveis, incluindo as versões antigas.
- Utilizar tecnologias de baixo custo ou livres.

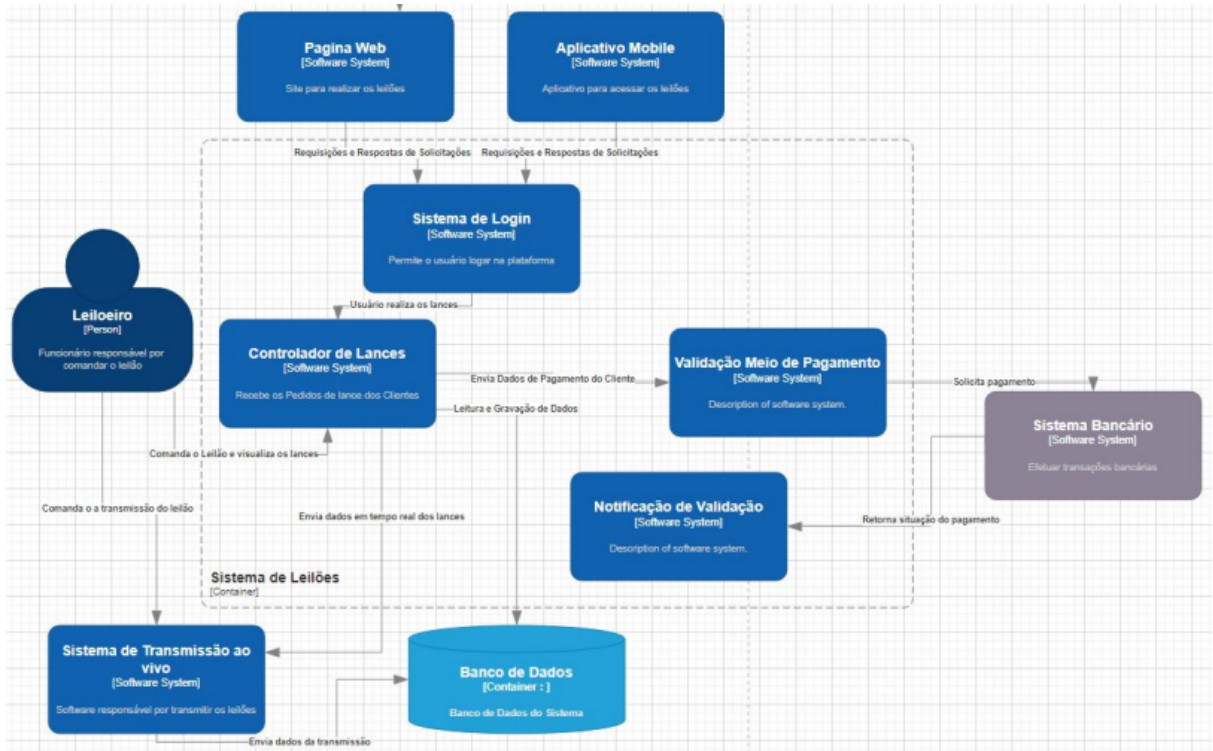
## VISÃO GERAL DO CENÁRIO



## VISÃO DOS COMPONENTES



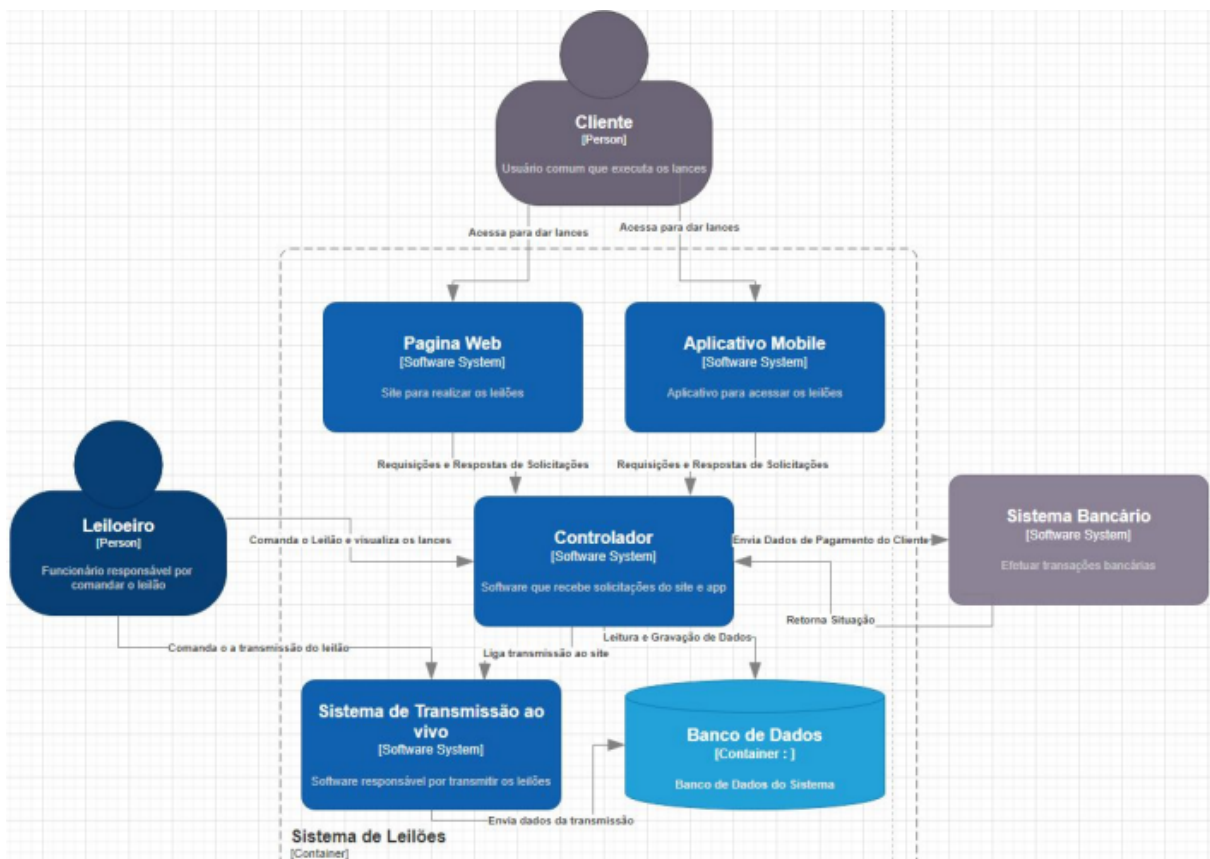
## VISÃO DOS MÓDULOS



## PADRÃO ARQUITETURAL

O padrão arquitetural MVC(MODEL-VIEW-CONTROLLER) divide o projeto em 3 partes independentes: a manipulação de dados, a interface do usuário e o controlador, optamos por utilizar este padrão por conta da facilidade de manutenção de código, além de permitir a reutilização de código.

## ILUSTRAÇÃO DO MODELO ARQUITETURAL





## TECNOLOGIAS UTILIZADAS

Utilizamos o JAVA pois é uma linguagem que já está consolidada no mercado, assim nos proporcionando uma quantidade enorme de bibliotecas para auxiliar no desenvolvimento do software. Para a manipulação de banco de dados, optamos por utilizar o MySQL, pois além de fácil utilização e curva de aprendizagem, ela possui uma ótima integração com java.

## PADRÃO ADAPTER

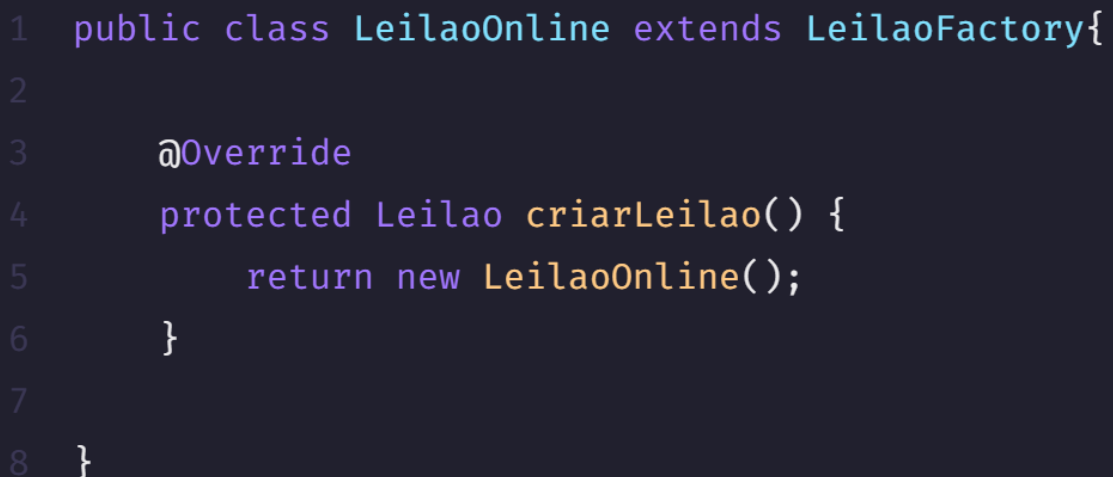
Optamos por usar o padrão Adapter por conta de que as empresas fundidas podem ter implementações diferentes, para isso o padrão ajuda com uma classe mediadora para adaptar as classes sem modificar os códigos de ambas as partes.

```
1 public class AdapterLanceOnline implements Lance {
2
3     private LanceOnline lance;
4
5     public AdapterLanceOnline(LanceOnline lance, Leilao leilao) {
6         this.lance = lance;
7         lance.setLeilao(leilao);
8     }
9
10    @Override
11    public void fazer() {
12        lance.processar(2.000, lance.getLeilao());
13    }
14
15 }
```

Como por exemplo neste caso, onde a classe Lance é uma interface e através do adapter é criado diferentes tipos de classes relacionadas, dessa forma permitindo instanciar novos objetos diferentes sem a necessidade de alteração de código.

## PADRÃO FACTORY

O padrão factory foi escolhido pois ele fornece uma interface para classes relacionadas, além de permitir mais de uma implementação por interface, já que no padrão Factory podemos escolher qual método retornar para as subclasses.

A screenshot of a code editor with a dark background and light-colored text. At the top left, there are three colored circles: red, yellow, and green. The code is written in Java and is as follows:

```
1 public class LeilaoOnline extends LeilaoFactory{  
2  
3     @Override  
4     protected Leilao criarLeilao() {  
5         return new LeilaoOnline();  
6     }  
7  
8 }
```

Como por exemplo, na classe abstrata pode conter implementações que não serão úteis para a subclasse, neste caso o padrão Factory nos auxilia permitindo qual método poderemos retornar na subclasse.

## PADRÃO OBSERVER

O padrão Observer auxilia no controle dos lances, ele ficará responsável por adicionar, remover ou notificar sobre os lances, ele retornará uma lista com todos os lances registrados para o leiloeiro e retornará uma mensagem para o usuário sobre o estado atual do seu lance.

```
1 public class Subject {
2
3     private List<Observer> observers = new ArrayList<Observer>();
4
5     public void addObserver(Observer observer) {
6
7         observers.add(observer);
8
9     }
10
11     public void removeObserver(Observer observer) {
12
13         observers.remove(observer);
14
15     }
16
17     public void notifyObservers() {
18
19         Iterator<Observer> it = observers.iterator();
20         while (it.hasNext()) {
21             Observer obs= it.next();
22             obs.update(this);
23         }
24     }
25
26 }
```

## CONCLUSÃO

Com a realização deste artigo, foi possível interpretar corretamente como a organização de uma arquitetura de projetos realmente funciona, aprendendo a lidar com os padrões de projetos e suas implementações. Além disso, tivemos contato com diferentes modelos de arquitetura, compreendendo sua importância dentro do projeto. Finalmente, tivemos contato com os formatos e escritas de trabalhos científicos

## REFERÊNCIAS

Disponível em:

<https://www.devmedia.com.br/padrao-de-projeto-adapter-em-java/26467>. Acesso em: 30/11/2022

Disponível em:

<https://www.devmedia.com.br/padrao-de-projeto-factory-method-em-java/26348>. Acesso em: 30/11/2022

Disponível em:

<https://www.devmedia.com.br/padrao-de-projeto-observer-em-java/26163>. Acesso em: 30/11/2022