Trabalho – Projeto e Arquitetura de Software

Gabriel Brunichaki, Marcelo Bernardy

Version <1.0>

Histórico de revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <24/09/19> | <1.0> | Primeira versão | Marcelo Bernardy, |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1.1 Proposta 3](#__RefHeading___Toc565_393369374)

[1.2 Escopo 3](#__RefHeading___Toc567_393369374)

[1.3 Definições, Acrônimos e Abreviações 3](#__RefHeading___Toc569_393369374)

[1.4 Visão geral 4](#__RefHeading___Toc573_393369374)

[2. Representação da arquietetura 4](#__RefHeading___Toc575_393369374)

[3. Metas de arquitetura 4](#__RefHeading___Toc577_393369374)

[4. Casos de uso 4](#__RefHeading___Toc579_393369374)

[5. Processos 5](#__RefHeading___Toc589_393369374)

[6.2 Cliente pode fazer e propor leilões 6](#__RefHeading___Toc564_1557701131)

[6. Execução 7](#__RefHeading___Toc591_393369374)

[7. Implementation View 7](#__RefHeading___Toc593_393369374)

[7.1 Camadas 7](#__RefHeading___Toc597_393369374)

[8.2 Padrões de projeto utilizados 7](#__RefHeading___Toc1637_393369374)

[8. Tamanho e performance 8](#__RefHeading___Toc599_393369374)

[9. Qualidade 8](#__RefHeading___Toc601_393369374)

O documento a seguir visa apresentar a arquitetura utilizada para o pograma Leilões, para com que possamos seguir boas práticas de arquitetura. Neste documento poderá ser visualizado o problema, a representação da arquitetura, visualização do processo, entre outras boas práticas de programação.

## Proposta

A proposta deste documento é prover uma arquitetura compatível com o escopo definido seguindo os melhores padrões e metas de reusabilidade, escala, etc.

## Escopo

O escopo é construir uma arquitetura de camadas perante os seguintes requisitos:

1. um cliente pode fazer lances e propor leilões;

2. no papel de proponente de um leilão, pode também fechar um leilão;

3. no papel de cliente pode dar quantos lances quiser;

4. no papel de administrador uma pessoa pode consultar os leilões concluídos e pendentes;

5. no papel de proponente, uma pessoa pode consultar todos os seus leilões ativos e fechados.

E no que diz respeito aos padrões de projeto:

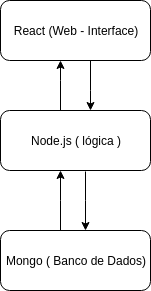
1. deve ser utilizado, pelo menos, 1 padrão de projeto em cada uma das camadas da aplicação;

## Definições, Acrônimos e Abreviações

## Visão geral

# Representação da arquietetura

A arquitetura utilizada vai seguir o padrão de Camadas, com uma parte de visualização realizada em React, camada de persistência e de negócio realizada em Node.js. Nosssa base de dados vai ser feita em Mongo.

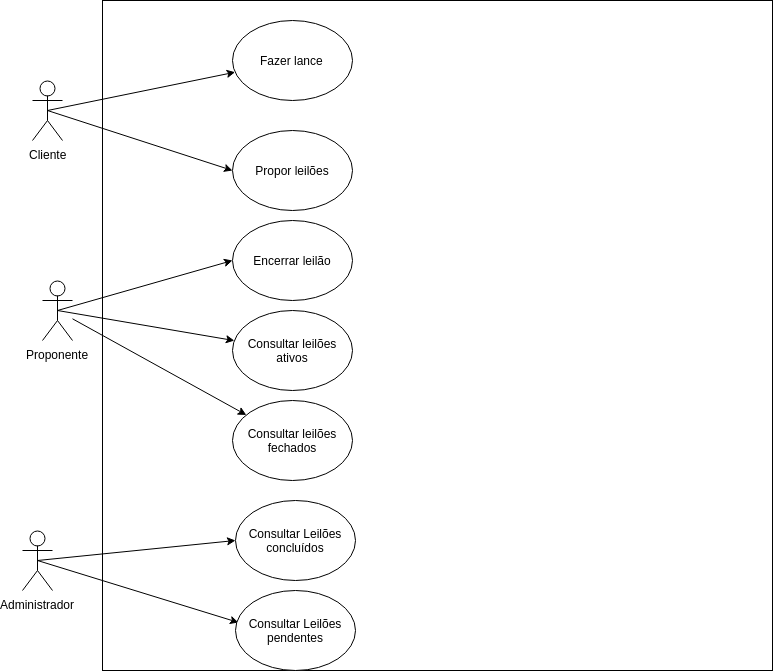


# Metas de arquitetura

A aplicação tem como objetivo obter performance, ter padrões de projeto bem definidos, além de

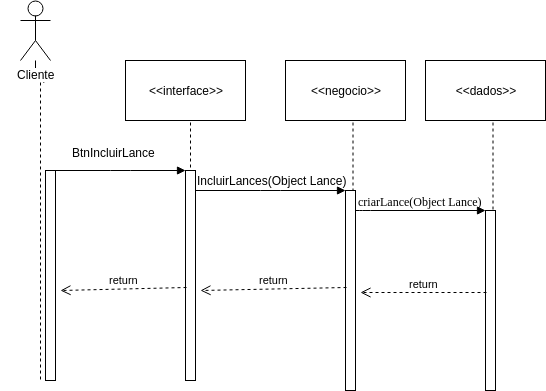
# Casos de uso

Os casos de uso dos requisitos estipulados são os seguintes:

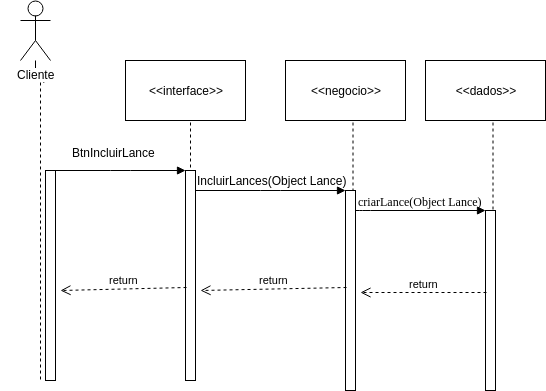


# Processos

6.1 Cliente pode fazer e propor leilões

 Cliente clica em adicionar leilão …

## 6.2 Cliente pode fazer e propor leilões

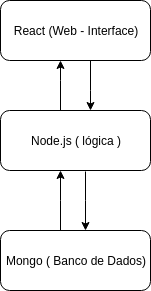


# Execução

Para executar a aplicação é necessário acesso a internet. Para desenvolvimento é necessário um servidor web.

# Implementation View

Para a implementação será utilizado React (camada de apresentação), Node.js (API) e Mongo (Base de dados).



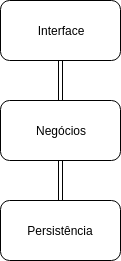
## Camadas

Para a arquitetura do projeto utilizamos as seguintes camadas:

Interface : Camada de exibição realizado em React.

Negócio : Camada onde toda a lógica de negócios.

Persistência : Camada de acesso ao banco de dados.



## 8.2 Padrões de projeto utilizados

Como padrões de projeto utilizamos RenderProps, Promise e Require:

**RenderProps:** Técnica de compartilhar código entre componentes react utilizando uma prop, cujo valor é uma função. Este padrão foi amplamente utilizado com na interface com o React.

**Promises:** Padrão de programação assíncrona no qual aguardamos a promise, isso significa, que enviamos uma requisição e uma promessa da volta desta assim não é necessário esperar a requisição ser concluída para continuar o fluxo de dados, transformando numa chamada assíncrona, utilizado na interface com o React.

**Require:** P

# Tamanho e performance

A aplicação tem um grau de complexidade relativamente simples, o que causa por consequência um menor tamanho para a aplicação. No quesito de performance por a aplicação conter schema também é simples, e por utilizar o protocolo padrão REST API, a aplicação deve conter uma boa performance.

# Qualidade

A arquitetura definida deve manter uma boa performance, simplicidade e testabilidade do software, o índice de acoplamento deve ser baixo e de alta reusabilidade.