Para implementar a regra de **desempate pelo primeiro lance do mesmo valor** ao **“bater o martelo”**, podemos seguir a seguinte abordagem:

---

1. Lógica de Competição dos Lances

- **Regra principal:** Quando houver lances com o mesmo valor, **vence o primeiro que ofereceu esse valor**.
- **Solução:** Ao buscar o lance vencedor, ordenamos os registros por **valor decrescente** e **data de criação crescente**.

---

2. Ajuste no Modelo `Bid.php`

No modelo `Bid.php`, adicione um método para buscar o **vencedor do leilão**:

```php
public static function getWinningBid($adId)
{
    return self::where('ad_id', $adId)
        ->orderByDesc('amount')  // Ordena pelo maior valor
        ->orderBy('created_at')  // Em caso de empate, o mais antigo ganha
        ->first();               // Pegamos o primeiro que satisfaz a condição
}
```

---

3. Criar o Controller para Fechar o Leilão

Criamos um **endpoint para “bater o martelo”** e definir o vencedor:

```php
namespace App\Http\Controllers;

use App\Models\Bid;
use Illuminate\Http\Request;
use App\Events\AuctionClosedEvent;

class AuctionController extends Controller
{
    public function closeAuction(Request $request)
    {
        $request->validate([
            'ad_id' => 'required|exists:ads,id',
        ]);
```

```
        $winningBid = Bid::getWinningBid($request->ad_id);

        if (!$winningBid) {
            return response()->json(['message' => 'Nenhum lance
        encontrado para este anúncio.'], 404);
        }

        // Dispara evento via WebSocket para notificar todos os
        usuários
        broadcast(new AuctionClosedEvent($winningBid))-
        >toOthers();

        return response()->json([
            'message' => 'Leilão encerrado!',
            'winner' => $winningBid
        ]);
    }
}
```

---

4. Criar o Evento `AuctionClosedEvent.php`

Este evento será transmitido via **Socket.IO** para avisar todos os usuários quem venceu o leilão.

```
namespace App\Events;

use App\Models\Bid;
use Illuminate\Broadcasting\Channel;
use Illuminate\Queue\SerializesModels;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;

class AuctionClosedEvent implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $winningBid;

    public function __construct(Bid $winningBid)
    {
        $this->winningBid = $winningBid;
    }
```

```
    public function broadcastOn()
    {
        return new Channel('auction');
    }

    public function broadcastWith()
    {
        return [
            'id' => $this->winningBid->id,
            'user' => $this->winningBid->user->name,
            'amount' => $this->winningBid->amount,
            'ad_id' => $this->winningBid->ad_id,
        ];
    }
}
```

---

5. Criar a Rota no `api.php`

Adicione a rota para encerrar o leilão:

```
use App\Http\Controllers\AuctionController;

Route::post('/close-auction', [AuctionController::class,
        'closeAuction']);
```

---

6. Ajustar o `server.js` para Enviar o Vencedor

No **servidor WebSocket**, ajustamos para enviar a atualização para todos os usuários quando o leilão for encerrado.

Edite `server.js` para incluir:

```
const io = require("socket.io")(6001, {
    cors: {
        origin: "*",
    },
});

io.on("connection", (socket) => {
    console.log("Usuário conectado:", socket.id);

    socket.on("newBid", (data) => {
        io.emit("updateBids", data);
```

```
    });

    socket.on("auctionClosed", (data) => {
        io.emit("auctionWinner", data); // Envia o vencedor para
        todos os usuários
    });

    socket.on("disconnect", () => {
        console.log("Usuário desconectado:", socket.id);
    });
});

console.log("Servidor WebSocket rodando na porta 6001...");
```

---

7. Criar a Interface Blade para Exibir o Vencedor

No seu **Blade (bids.blade.php)**, escutamos o evento de fechamento do leilão:

```
<input type="number" id="amount" placeholder="Valor do lance">
<button onclick="placeBid()">Enviar Lance</button>
<button onclick="closeAuction()">Encerrar Leilão</button>

<ul id="bids-list"></ul>
<h2 id="winner"></h2>

<script src="https://cdn.socket.io/4.0.1/socket.io.min.js"></
        script>
<script>
    const socket = io("http://127.0.0.1:6001");

    socket.on("updateBids", function (data) {
        document.getElementById("bids-list").innerHTML +=
            `<li>${data.user} fez um lance de R$ ${data.amount}</
        li>`;
    });

    socket.on("auctionWinner", function (data) {
        document.getElementById("winner").innerText =
            `  O vencedor é ${data.user} com um lance de R$ $
        {data.amount}`;
    });
```

```
    function placeBid() {
        let amount = document.getElementById("amount").value;
        fetch("/api/bids", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
                Authorization: "Bearer SEU_TOKEN_AQUI",
            },
            body: JSON.stringify({ ad_id: 1, amount: amount }),
        });
    }

    function closeAuction() {
        fetch("/api/close-auction", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
                Authorization: "Bearer SEU_TOKEN_AQUI",
            },
            body: JSON.stringify({ ad_id: 1 }),
        });
    }
</script>
```

---

8. Testando o Fluxo

1. Inicie o **Laravel**:

   ```
   php artisan serve
   ```

2. Inicie o **servidor WebSocket**:

   ```
   node server.js
   ```

3. Acesse a página Blade (`bids.blade.php`) e:
   - Usuários fazem lances.
   - O administrador pressiona **“Encerrar Leilão”**.
   - O vencedor aparece na tela conforme a regra do **primeiro a dar o mesmo lance em caso de empate**.

Conclusão

✔ Agora temos um sistema de **lances em tempo real** com **desempate pelo lance mais antigo de mesmo valor**!

✔ O vencedor é transmitido para todos os usuários via **Socket.IO** e Laravel Echo.