# Projeto de Sistema de Reserva de Mesas para Restaurante

## 1. Informações da Equipe

Nome e RA dos membros da equipe:

- Membro 1: Yule Lima Hohenfeld Lyra - 12724134514

- Membro 2: Gabriel Pastore Menezes -12724121514

- Membro 3: Alan Ravi de Sá Lopes carneiro - 12724140887

- Membro 4: gabriel alcantara da purificação- 12724133851

- Membro 5: Kaique Natan de Santana Chagas- 12724134123

- Membro 6: Miguel Reis- 12724115122

- Membro 7: João Victor Queiroz Dias Menezes - 12724137785

## 2. Descrição Geral do Projeto

Este projeto é uma aplicação cliente-servidor para a reserva de mesas em um restaurante. O sistema possui três tipos principais de usuários: Atendente de Reserva, Garçom e Gerente do Restaurante. O Atendente recebe ligações e cadastra reservas, o Garçom registra que uma reserva foi atendida, e o Gerente monitora todas as operações, incluindo reservas, garçons e relatórios. O sistema permite criar, editar, cancelar reservas e alterar o status das mesas (reservada ou livre).

## 3. Tecnologias Utilizadas

Linguagens: Typescript (React Native para o app mobile)

Banco de Dados: MySQL

Backend: API REST para comunicação cliente-servidor

Frontend: Aplicação mobile feita com React Native

Bibliotecas e ferramentas adicionais: Axios para chamadas API, Expo para desenvolvimento React Native

## 4. Funcionalidades Implementadas

• Reservar mesa  
• Cancelar reserva  
• Visualizar mesas reservadas  
• Feedback ao usuário sobre sucesso ou falha nas operações  
• Atualização da lista de mesas reservadas ao entrar na tela  
• Modal para mostrar detalhes da reserva  
• Navegação entre telas  
• confirmar cliente na mesa

## 5. Requisitos de Software para Execução

Para executar e testar a aplicação, os seguintes softwares são necessários:  
• Android Studio (para emular dispositivos Android)  
• Node.js (para rodar o ambiente React Native)  
• Expo CLI (ferramenta para desenvolvimento com React Native)  
• Git (para clonar o repositório)  
• Um emulador Android configurado via Android Studio ou um dispositivo físico conectado

• **Java JDK** (necessário para o Android Studio e build do React Native)

• **Yarn ou npm (gerenciadores de pacotes para o Node.js)**

## 6. Passos para Executar e Testar a Aplicação

1. Instalar Android Studio e configurar um emulador Android.  
2. Instalar Node.js no computador.  
3. Instalar Expo CLI globalmente com o comando: npm install -g expo-cli  
4. Clonar o repositório do projeto com git clone<<https://github.com/YuleHohenfeld/leBolon>>  
5. Navegar até a pasta do projeto no terminal.  
6. Executar npm install para instalar as dependências.  
7. Executar expo start para iniciar o servidor de desenvolvimento.  
8. Abrir o aplicativo Expo Go no dispositivo móvel ou usar o emulador para rodar o app(android Studio).  
9. Testar as funcionalidades de reserva, cancelamento e visualização de mesas.

## 8. Repositorios front end e Back end/ video da aplicaçao

Front end: <https://github.com/YuleHohenfeld/leBolon>

back end: <https://github.com/guelreis/A3>

video: https://www.youtube.com/watch?v=FwYz-RuiNoc

## . justificative para abordagem de comunicaçao

Para a comunicação entre o cliente e o servidor da aplicação, adotamos o padrão **HTTP/REST utilizando o formato JSON**. Essa escolha se deu principalmente pela **simplicidade, ampla adoção no desenvolvimento web e compatibilidade com diversas linguagens e frameworks**, como Express.js no backend e bibliotecas como Axios no cliente. REST permite a criação de **endpoints bem definidos** para cada funcionalidade da aplicação, como por exemplo /mesas, /reservas ou /login, facilitando o desenvolvimento, manutenção e testes da aplicação.

Além disso, o uso do **formato JSON** como meio de troca de dados contribui para uma comunicação leve, legível e eficiente entre as partes, sendo também facilmente manipulável em JavaScript. Essa abordagem atendeu perfeitamente aos requisitos do projeto, que envolvem principalmente **operações CRUD** (criar, ler, atualizar e deletar) sobre entidades como mesas, usuários e reservas.

Por se tratar de uma aplicação que **não exige comunicação em tempo real**, descartamos abordagens como WebSocket, que seriam mais complexas e desnecessárias para esse escopo. Portanto, o uso de REST com JSON mostrou-se a escolha mais equilibrada entre simplicidade, clareza e funcionalidade.

Optamos por utilizar uma API REST para comunicação entre o aplicativo React Native e o backend que gerencia o banco de dados MySQL. Essa abordagem é amplamente utilizada, padronizada e facilita a manutenção e escalabilidade do sistema. Além disso, o uso de HTTP/HTTPS torna a comunicação simples e compatível com diversos clientes, como aplicativos móveis e web. O React Native, junto com o Expo, oferece um ambiente ágil para o desenvolvimento multiplataforma, permitindo que o app rode tanto em Android quanto iOS com o mesmo código-base.