**Projeto: Controle Remoto para Cadeira de Rodas com Arduino, API PHP e App Mobile**

**Índice**

1. Introdução  
   1.1. Contexto e Motivação  
   1.2. Objetivos do Projeto
2. Arquitetura do Sistema  
   2.1. Visão Geral  
   2.2. Componentes do Sistema
3. Configuração do Servidor e Banco de Dados  
   3.1. Ambiente de Desenvolvimento (PHP + MySQL)  
   3.2. Criação do Banco de Dados  
   3.3. Desenvolvimento da API em PHP
4. Desenvolvimento do App Mobile com React Native (Expo)  
   4.1. Criação do Projeto  
   4.2. Conversão para JavaScript Puro  
   4.3. Implementação da Tela de Controle  
   4.4. Comunicação com a API via Axios
5. Integração com o Arduino  
   5.1. Conexão Física e Comunicação  
   5.2. Exemplo de Código para Polling de Comandos
6. Fluxo de Dados e Comunicação
7. Considerações de Segurança e Boas Práticas
8. Possíveis Expansões e Melhorias Futuras
9. Conclusão
10. Referências

**1. Introdução**

**1.1. Contexto e Motivação**

A mobilidade de pessoas com deficiência (PCD) é um desafio constante, principalmente quando se trata de acesso a tecnologias assistivas de baixo custo. Este projeto visa desenvolver um sistema integrado que permita o controle remoto de uma cadeira de rodas utilizando um microcontrolador (Arduino) conectado a uma API desenvolvida em PHP, que, por sua vez, se conecta a um banco de dados MySQL. Um aplicativo mobile, desenvolvido com React Native (Expo), possibilitará que o usuário e o cuidador controlem e monitorem a cadeira remotamente.

**1.2. Objetivos do Projeto**

* **Objetivo Geral:**  
  Desenvolver um sistema de controle remoto para cadeira de rodas automatizada e de baixo custo, integrando Arduino, API PHP e um aplicativo mobile.
* **Objetivos Específicos:**
  + Criar uma API em PHP para gerenciamento dos comandos e usuários, utilizando um banco de dados MySQL.
  + Desenvolver um aplicativo mobile (com Expo/React Native) para envio e recepção de comandos.
  + Implementar a comunicação entre o app, a API e o Arduino, permitindo o controle dos motores da cadeira.
  + Permitir dois tipos de usuários: cadeirante (principal) e cuidador (para monitoramento e alertas).

**2. Arquitetura do Sistema**

**2.1. Visão Geral**

O sistema é composto por três camadas principais:

* **Camada de Dispositivos:**  
  A cadeira de rodas equipada com Arduino e módulos de comunicação (Wi-Fi ou Bluetooth) para controle dos motores e leitura de sensores.
* **Camada de Servidor/API:**  
  Uma API desenvolvida em PHP que se conecta a um banco de dados MySQL, onde são armazenados os dados dos usuários e os comandos enviados pelo app.
* **Camada de Aplicativo Mobile:**  
  Um app desenvolvido em React Native com Expo, que permite aos usuários (cadeirantes e cuidadores) enviar comandos para a cadeira e monitorar seu status.

**2.2. Componentes do Sistema**

* **Arduino:** Controla os motores da cadeira, recebe comandos e pode enviar dados (por exemplo, status da bateria ou alertas) ao servidor.
* **Módulo de Comunicação:** Módulo Wi-Fi (ex. ESP8266/ESP32) ou Bluetooth para comunicação entre o Arduino e a API.
* **Servidor Web:** Hospedagem da API em PHP e banco de dados MySQL.
* **Aplicativo Mobile:** Interface de controle e monitoramento, desenvolvido em React Native (Expo).
* **Banco de Dados:** Armazena informações de usuários, comandos e logs de ações.

**3. Configuração do Servidor e Banco de Dados**

**3.1. Ambiente de Desenvolvimento (PHP + MySQL)**

* **Ferramenta:** XAMPP, WAMP ou outro ambiente de desenvolvimento local.
* **Serviços Necessários:** Apache (para PHP) e MySQL.

**3.2. Criação do Banco de Dados**

Utilize o phpMyAdmin (acessível via http://localhost/phpmyadmin) para criar o banco de dados:

CREATE DATABASE cadeira\_db;

USE cadeira\_db;

CREATE TABLE Usuarios (

id\_usuario INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

senha\_hash VARCHAR(255) NOT NULL,

tipo ENUM('cadeirante', 'cuidador') NOT NULL

);

CREATE TABLE Comandos (

id\_comando INT AUTO\_INCREMENT PRIMARY KEY,

id\_usuario INT NOT NULL,

comando VARCHAR(50) NOT NULL,

data\_hora TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('pendente','executado') DEFAULT 'pendente',

FOREIGN KEY (id\_usuario) REFERENCES Usuarios(id\_usuario)

);

**3.3. Desenvolvimento da API em PHP**

Crie uma pasta no diretório do servidor (por exemplo, C:\xampp\htdocs\cadeira\_api) e adicione os seguintes arquivos:

**config.php**

<?php

$host = "localhost";

$db = "cadeira\_db";

$user = "root";

$pass = "";

try {

$pdo = new PDO("mysql:host=$host;dbname=$db;charset=utf8", $user, $pass);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

die(json\_encode(["error" => "Erro na conexão: " . $e->getMessage()]));

}

?>

**comandos.php**

Este arquivo serve tanto para receber comandos (via POST) quanto para retornar os comandos pendentes (via GET).

<?php

header("Access-Control-Allow-Origin: \*");

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

require "config.php";

$method = $\_SERVER['REQUEST\_METHOD'];

if ($method == 'POST') {

// Recebe dados do comando

$data = json\_decode(file\_get\_contents("php://input"));

if (!isset($data->id\_usuario) || !isset($data->comando)) {

echo json\_encode(["error" => "Dados insuficientes"]);

exit;

}

$id\_usuario = $data->id\_usuario;

$comando = $data->comando;

$stmt = $pdo->prepare("INSERT INTO Comandos (id\_usuario, comando) VALUES (?, ?)");

if($stmt->execute([$id\_usuario, $comando])){

echo json\_encode(["message" => "Comando inserido com sucesso"]);

} else {

echo json\_encode(["error" => "Falha ao inserir comando"]);

}

} elseif ($method == 'GET') {

// Retorna comandos pendentes

$stmt = $pdo->query("SELECT \* FROM Comandos WHERE status='pendente' ORDER BY data\_hora ASC");

$comandos = $stmt->fetchAll(PDO::FETCH\_ASSOC);

echo json\_encode($comandos);

} else {

echo json\_encode(["error" => "Método não suportado"]);

}

?>

**Teste a API:**  
Acesse http://localhost/cadeira\_api/comandos.php via navegador ou use ferramentas como Postman para testar o envio e a consulta de comandos.

**4. Desenvolvimento do App Mobile com React Native (Expo)**

**4.1. Criar o Projeto com Expo**

No terminal, execute:

npx create-expo-app@latest appCadeiraDeRodas --template blank

Isso criará um projeto Expo básico.

**4.2. Instalação e Configuração**

1. Acesse a pasta do projeto:
2. cd appCadeiraDeRodas
3. Instale as dependências (por exemplo, Axios para requisições):
4. npm install axios
5. Inicie o app:
6. npx expo start

**4.3. Implementação da Tela de Controle**

Abra o arquivo App.js (em JavaScript puro) e insira o seguinte código:

import React from 'react';

import { View, Text, Button, StyleSheet, Alert } from 'react-native';

import axios from 'axios';

export default function App() {

// Exemplo: Usuário cadastrado com ID 1 (cadeirante)

const id\_usuario = 1;

// Substitua pela URL real do seu servidor

const API\_URL = "http://192.168.1.10/cadeira\_api/comandos.php";

// Função para enviar comando à API

const sendCommand = (command) => {

axios.post(API\_URL, {

id\_usuario: id\_usuario,

comando: command

})

.then(response => {

Alert.alert("Sucesso", response.data.message || "Comando enviado");

})

.catch(error => {

console.error("Erro:", error);

Alert.alert("Erro", "Falha ao enviar comando");

});

};

return (

<View style={styles.container}>

<Text style={styles.title}>Controle da Cadeira</Text>

<Button title="Frente" onPress={() => sendCommand("frente")} />

<View style={styles.row}>

<Button title="Esquerda" onPress={() => sendCommand("esquerda")} />

<Button title="Direita" onPress={() => sendCommand("direita")} />

</View>

<Button title="Trás" onPress={() => sendCommand("tras")} />

</View>

);

}

const styles = StyleSheet.create({

container: { flex: 1, padding: 20, justifyContent: 'center', backgroundColor: '#f8f9fa' },

title: { fontSize: 24, fontWeight: 'bold', textAlign: 'center', marginBottom: 20 },

row: { flexDirection: 'row', justifyContent: 'space-around', marginVertical: 10 }

});

**Observação:**

* Altere a URL do API\_URL para o endereço real do seu servidor (por exemplo, o IP local obtido via ipconfig ou um domínio, se estiver online).

**4.4. Teste do App**

* Inicie o app com npx expo start e use o **Expo Go** no seu celular para visualizar e testar os botões que enviam comandos à API.

**5. Integração com o Arduino**

**5.1. Conexão Física e Comunicação**

* **Arduino:** Utilize um Arduino (ou ESP32/ESP8266) para controlar os motores da cadeira de rodas.
* **Módulo de Comunicação:** Para conectar o Arduino à API, use um módulo Wi‑Fi (por exemplo, ESP8266) ou Bluetooth.
* **Circuito:** Monte o circuito para controlar os motores (por meio de um driver, como L298N) e para receber comandos do servidor via comunicação sem fio.

**5.2. Exemplo de Código Arduino (Poll de Comandos)**

Caso opte por fazer o Arduino realizar polling na API, um exemplo básico usando ESP32 pode ser:

#include <WiFi.h>

#include <HTTPClient.h>

const char\* ssid = "SEU\_SSID";

const char\* password = "SUA\_SENHA";

String apiUrl = "http://192.168.1.10/cadeira\_api/comandos.php"; // URL da API

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nWiFi conectado!");

}

void loop() {

if(WiFi.status() == WL\_CONNECTED){

HTTPClient http;

http.begin(apiUrl);

int httpCode = http.GET();

if(httpCode > 0) {

String payload = http.getString();

Serial.println("Comandos pendentes:");

Serial.println(payload);

// Processar o JSON e executar comandos conforme necessário

} else {

Serial.println("Erro na requisição HTTP");

}

http.end();

}

delay(5000); // Requisição a cada 5 segundos

}

**6. Fluxo de Dados e Comunicação**

1. **App Mobile (React Native):**
   * Envia comandos (ex.: "frente", "tras", etc.) via requisição HTTP POST para a API PHP.
2. **API PHP:**
   * Recebe os comandos e os insere na tabela de **Comandos** no banco de dados MySQL.
3. **Arduino (ESP32/ESP8266):**
   * Realiza polling na API (GET) para obter comandos pendentes.
   * Executa os comandos recebidos, acionando os motores da cadeira de rodas.
4. **Feedback:**
   * O Arduino pode enviar status de volta para a API (através de requisições POST) para atualizar o status do comando e possibilitar monitoramento remoto.

**7. Considerações de Segurança e Boas Práticas**

* **Segurança na Comunicação:**  
  Utilize HTTPS se o servidor estiver online para proteger os dados.
* **Armazenamento Seguro de Senhas:**  
  Armazene as senhas dos usuários utilizando algoritmos de hash (ex.: bcrypt).
* **Validação dos Dados:**  
  Implemente validações tanto no app quanto na API para evitar injeção de dados e outros ataques.
* **Gerenciamento de Erros:**  
  Inclua tratamento de erros e logs tanto no lado do servidor quanto no Arduino para facilitar a manutenção.

**8. Possíveis Expansões e Melhorias Futuras**

* **Controle via Celular com IA:**  
  Desenvolver uma camada de IA para ajustar a velocidade e potência automaticamente com base em dados de sensores.
* **Integração de Sensores Adicionais:**  
  Incluir sensores ultrassônicos, de inclinação, e de bateria para monitoramento em tempo real.
* **Notificações e Alertas:**  
  Implementar um sistema de alertas no app para notificar cuidadores em situações de emergência.
* **Interface Gráfica Avançada:**  
  Expandir o app para incluir gráficos, históricos e controles mais sofisticados.

**9. Conclusão**

Este projeto integra um aplicativo mobile em React Native, uma API RESTful em PHP conectada a um banco de dados MySQL e um sistema de controle baseado em Arduino para desenvolver uma solução remota para cadeira de rodas. O fluxo de dados é estruturado para permitir o controle e monitoramento remoto, com possibilidades de expansão para incorporar IA e sensores adicionais, aumentando a autonomia e segurança dos usuários.

**10. Referências**

* Documentação do [React Native](https://reactnative.dev/docs/getting-started)
* Documentação do [Expo](https://docs.expo.dev/)
* [Axios Documentation](https://axios-http.com/docs/intro)
* Documentação do [PHP PDO](https://www.php.net/manual/en/book.pdo.php)
* [Tutorial de criação de API RESTful com PHP e MySQL](https://www.tutorialspoint.com/php/php_restful_web_services.htm)
* [Exemplo de comunicação com Arduino via Wi-Fi (ESP32)](https://www.arduino.cc/en/Reference/WiFi)