Mudanças para Conexão com Servidor Local

Projeto: Pokédex App - React Native
Data: 12 de Janeiro de 2026
Autor: Isaac Barros

📋 Resumo Executivo

Este documento detalha todas as mudanças necessárias para configurar o aplicativo React Native Pokédex para buscar dados de um servidor local Node.js, em vez de usar a PokéAPI pública.

🎯 Objetivo

Permitir que o aplicativo mobile busque dados de Pokémons de um servidor backend local rodando na mesma rede, facilitando desenvolvimento e testes offline.

🔧 Mudanças Realizadas

1. Configuração do Serviço de API

Arquivo: `services/pokeApi.js`

Antes:

```
import axios from 'axios';

const BASE_URL = 'https://pokeapi.co/api/v2';

export const fetchPokemons = async (limit = 20, offset = 0) => {
  const response = await axios.get(`${BASE_URL}/pokemon`, {
    params: { limit, offset }
  });

  const pokemonPromises = response.data.results.map(async (pokemon) => {
    const details = await fetchPokemonDetails(pokemon.url);
    return details;
  });

  return await Promise.all(pokemonPromises);
};
```

Depois:

```
import axios from 'axios';

// Configuração para usar servidor local
const USE_LOCAL_SERVER = true;
const LOCAL_SERVER_URL = 'http://192.168.0.4:3000';
const POKEAPI_URL = 'https://pokeapi.co/api/v2';

const BASE_URL = USE_LOCAL_SERVER ? LOCAL_SERVER_URL : POKEAPI_URL;

console.log('🔧 Configuração da API:');
console.log('   Modo:', USE_LOCAL_SERVER ? 'SERVIDOR LOCAL' : 'POKEAPI PÚBLICA');
console.log('   URL:', BASE_URL);

export const fetchPokemons = async (limit = 20, offset = 0) => {
  try {
    if (USE_LOCAL_SERVER) {
      // Buscar do servidor local
      console.log('Buscando do servidor local:', `${BASE_URL}/api/pokemon`);
      const response = await axios.get(`${BASE_URL}/api/pokemon`);
      console.log('Resposta do servidor local:', response.data);
      return response.data;
    } else {
      // Buscar da PokéAPI pública
      const response = await axios.get(`${BASE_URL}/pokemon`, {
        params: { limit, offset }
      });

      const pokemonPromises = response.data.results.map(async (pokemon) => {
        const details = await fetchPokemonDetails(pokemon.url);
        return details;
      });

      return await Promise.all(pokemonPromises);
    }
  } catch (error) {
    console.error('Erro ao buscar pokémons:', error);
    console.error('URL tentada:', USE_LOCAL_SERVER ? `${BASE_URL}/api/pokemon` :
`${BASE_URL}/pokemon`);
    throw error;
  }
};
```

Mudanças:

- ☑ Adicionada flag `USE_LOCAL_SERVER` para alternar entre servidor local e PokéAPI
- ☑ Configuração do IP local (`192.168.0.4:3000`)
- ☑ Lógica condicional para buscar dados do servidor correto
- ☑ Logs detalhados para debug

---

2. Configuração do Servidor Backend

Arquivo: `aula/server/app.js`

Antes:

```
const express = require("express")
const path = require("path")

const pokemonRouter = require("./routers/PokemonRouter")

const app = express()
const PORT = 3000

app.use(express.static(path.join(__dirname,"..","public")))
app.use("/api/pokemon", pokemonRouter)

app.listen(
    PORT,
    () => {
        console.log(`API rodando em http://localhost:${PORT}`)
    }
)
```

Depois:

```
const express = require("express")
const path = require("path")

const pokemonRouter = require("./routers/PokemonRouter")

const app = express()
const PORT = 3000

app.use(express.static(path.join(__dirname,"..","public")))
app.use("/api/pokemon", pokemonRouter)

app.listen(
    PORT,
    '0.0.0.0', // Aceita conexões de qualquer IP da rede
    () => {
        console.log(`API rodando em http://localhost:${PORT}`)
        console.log(`Acessível na rede em http://192.168.0.4:${PORT}`)
    }
)
```

Mudanças:

- ☑ Adicionado parâmetro `'0.0.0.0'` para aceitar conexões da rede local
- ☑ Mensagem de log indicando URL acessível na rede

Por quê? Sem o parâmetro `'0.0.0.0'`, o servidor Node.js só aceita conexões de `localhost`, impedindo que dispositivos móveis na mesma rede acessem o servidor.

---

3. Atualização do Componente Card

Arquivo: `pokedex/card.jsx`

Antes:

```
const Card = ({id, nome}) => {
    const imageUrl =
`https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/${id}.png`;

    return(
        <View style={styles.card}>
            <Image source={{uri: imageUrl}} style={styles.image} />
            <Label nome={nome} />
            <PokedexButton title="CAPTURAR" onPress={() => alert('Você capturou '
+ nome + '!')} />
        </View>
    )
}
```

Depois:

```
const Card = ({id, nome, imagem}) => {
    // Se vier imagem do servidor, usa ela; senão constrói a URL
    const imageUrl = imagem ||
`https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/official-artwork/${id}.png`;

    return(
        <View style={styles.card}>
            <Text style={styles.pokemonId}>#{id}</Text>
            <Image source={{uri: imageUrl}} style={styles.image} />
            <Label nome={nome} />
            <PokedexButton title="CAPTURAR" onPress={() => alert('Você capturou '
+ nome + '!')} />
        </View>
    )
}
```

Mudanças:

- ☑ Adicionado prop `imagem`
- ☑ Lógica para usar URL do servidor ou construir URL padrão

- ☑ Exibição do ID do Pokémon

---

4. Atualização do Componente Main

Arquivo: `pokedex/main.jsx`

Antes:

```
renderItem={
    ({item}) => {
        return <Card
        id={item.id}
        nome={item.nome}
        />
    }
}
```

Depois:

```
renderItem={
    ({item}) => {
        return <Card
        id={item.id}
        nome={item.nome}
        imagem={item.imagem}
        />
    }
}
```

Mudanças:

- ☑ Passando campo `imagem` para o componente Card

🔑 Mudança Crítica

A mudança mais importante foi no arquivo `aula/server/app.js`:

```
app.listen(PORT, '0.0.0.0', ...)
```

Por que `'0.0.0.0'` é essencial?

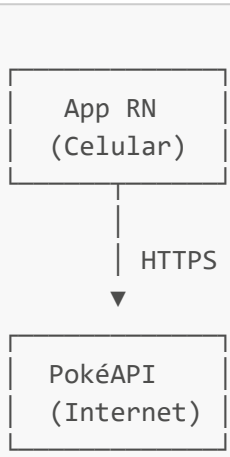
Configuração	Comportamento
`app.listen(PORT)`	Aceita apenas conexões de `localhost` (127.0.0.1)

Configuração	Comportamento
`app.listen(PORT, '0.0.0.0')`	Aceita conexões de qualquer IP da rede local

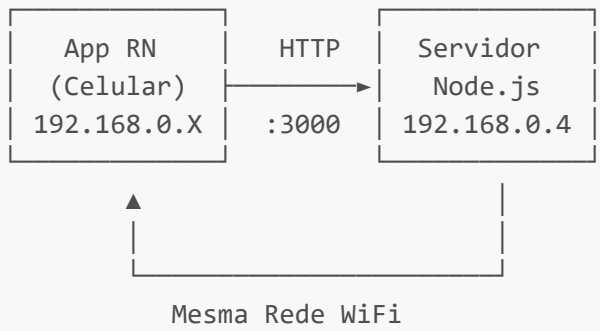
Sem o `'0.0.0.0'`, o celular na rede WiFi não consegue acessar o servidor, resultando em erro de "Network Error".

📊 Arquitetura

Antes (PokéAPI Pública):



Depois (Servidor Local):



🚀 Como Usar

Iniciar o Servidor Local:

```
cd c:\Users\Ulisses\Desktop\isaac\barbeiroapp\aula\server
node app.js
```

Saída esperada:

```
API rodando em http://localhost:3000
Acessível na rede em http://192.168.0.4:3000
```

Iniciar o App React Native:

```
cd c:\Users\Ulisses\Desktop\isaac\barbeiroapp
npx expo start
```

Testar no Navegador:

```
http://localhost:3000/api/pokemon
```

Alternar Entre Servidor Local e PokéAPI

Em `services/pokeApi.js`, altere:

```
// Usar servidor local
const USE_LOCAL_SERVER = true;

// Usar PokéAPI pública
const USE_LOCAL_SERVER = false;
```

Estrutura de Dados

Servidor Local (`/api/pokemon`):

```
[
  {
    "id": 1,
    "nome": "Bulbasaur",
    "imagem": "https://raw.githubusercontent.com/.../1.png",
    "habilidades": ["planta", "veneno"]
  },
  {
    "id": 2,
    "nome": "Ivysaur",
    "imagem": "https://raw.githubusercontent.com/.../2.png",
    "habilidades": ["planta", "veneno"]
  }
]
```

PokéAPI Pública:

```
{
  "id": 1,
  "nome": "bulbasaur",
  "types": [{"type": {"name": "grass"}}, {"type": {"name": "poison"}}],
  "sprites": {...},
  "height": 7,
  "weight": 69
}
```

⚠ Requisitos

Rede:

- ☑ Celular e PC na mesma rede WiFi
- ☑ IP do PC: `192.168.0.4`
- ☑ Porta do servidor: `3000`

Software:

- ☑ Node.js instalado
- ☑ Expo Go no celular
- ☑ Dependências instaladas (`npm install`)

Firewall:

- ⚠ Porta 3000 pode precisar ser liberada no firewall do Windows

---

Troubleshooting

Erro: "Network Error"

Causas possíveis:

1. Servidor não está rodando
2. Celular e PC em redes diferentes
3. Firewall bloqueando a porta 3000
4. IP incorreto no `LOCAL_SERVER_URL`

Soluções:

```
# Verificar IP do PC
ipconfig | Select-String "IPv4"

# Testar servidor localmente
curl http://localhost:3000/api/pokemon
```

```
# Liberar porta no firewall (executar como Admin)
netsh advfirewall firewall add rule name="Node.js Server" dir=in action=allow
protocol=TCP localport=3000
```

Erro: "Cannot find module"

Solução:

```
cd aula/server
npm install
```

Benefícios

Servidor Local:

- ☑ Desenvolvimento offline
- ☑ Controle total dos dados
- ☑ Sem rate limits
- ☑ Resposta mais rápida
- ☑ Dados customizados

PokéAPI Pública:

- ☑ Sem necessidade de servidor
- ☑ Dados completos e atualizados
- ☑ Sempre disponível
- ☑ Sem manutenção

Referências

- **Express.js:** https://expressjs.com/
- **Axios:** https://axios-http.com/
- **PokéAPI:** https://pokeapi.co/
- **React Native:** https://reactnative.dev/
- **Expo:** https://expo.dev/

Checklist de Implementação

- ☑ Criar flag `USE_LOCAL_SERVER` em `services/pokeApi.js`
- ☑ Configurar IP local (`192.168.0.4:3000`)
- ☑ Adicionar lógica condicional para buscar dados
- ☑ Modificar servidor para aceitar conexões da rede (`0.0.0.0`)
- ☑ Atualizar componente Card para suportar campo `imagem`

- ☑ Passar campo `imagem` no componente Main
- ☑ Instalar dependências do servidor
- ☑ Testar servidor local no navegador
- ☑ Testar app no Expo Go
- ☑ Adicionar logs de debug

📞 Suporte

Para alternar entre modos ou resolver problemas, consulte os arquivos:

- `SERVIDOR_LOCAL.md` - Guia completo de uso
- `services/pokeApi.js` - Configurações da API

Documento gerado em: 12/01/2026
Versão: 1.0
Status: ☑ Implementado e Testado