Consumindo uma API Curio com React + TypeScript - Primeiros Passos

Guia completo de como começar o seu projeto React para atender uma API Curio, como configurar requests, casos de usos, rotas e estilização de components com MaterialUI

- Guia do Framework Curio + React

Guia do Framework Curio + React

Configurando o Curio em um Ambiente React

Pré-requisitos

- Node.js (v16 ou superior)
- npm ou yarn (Versao atual da documentacao utilizando yarn)
- React (v16.8 ou superior, com React 18+ recomendado)

Nesse projeto estamos usando as versões mais atuais das ferramentas [Node 23v, React 19, MaterialUI 7.1.1]

Configuracoes Iniciais

Primeiro configure o yarn para acessar a biblioteca da Evologica no nexus

```
yarn config set registry https://nexus.dev.evologica.com.br/repository/npm/
```

1. **Instale os pacotes necessários:**

```
```

```
# Dependências principais
yarn add react react-dom
# Tipagens
yarn add -D @types/react @types/react-dom
# TypeScript
yarn add -D typescript ts-loader
# Babel
yarn add -D @babel/core @babel/preset-env @babel/preset-react babel-loader
# Webpackyarn add -D webpack webpack-cli webpack-dev-server html-webpack-plugin clean-webpack-plugin
yarn add -D ejs html-webpack-plugin
yarn add process --dev
# Outros loaders
yarn add -D css-loader style-loader file-loader
# Express (para server.js)
yarn add express
# Ferramentas extra (igual Cognera)
yarn add -D rimraf ts-node
```

E entao instale as bibliotecas Curio

```
yarn add @curio/client
```

2. **Criando arquivos de configuração:**

No diretorio do seu projeto vamos criar **3 arquivos**, `tsconfig.json`, `.babelrc`, `webpack.config.js` e uma pasta `public` na raiz do seu diretorio

Configuracao do arquivo `tsconfig.json`

```
{
    "compilerOptions": {
        "target": "ES6",
        "jsx": "react",
        "module": "ESNext",
        "moduleResolution": "Node",
```

```
        "esModuleInterop": true,
        "strict": true,
        "skipLibCheck": true,
        "forceConsistentCasingInFileNames": true
    },
    "include": [
        "src"
    ]
}
```

Configuracao do arquivo `.babelrc`

```
{
    "presets": [
        "@babel/preset-env",
        "@babel/preset-react"
    ]
}
```

Configuracao do arquivo `webpack.config.js`

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');const { CleanWebpackPlugin } =
require('clean-webpack-plugin');
const webpack = require('webpack');
module.exports = {
    entry: './src/index.tsx',
    devtool: 'eval-source-map',
    output: {
        path: path.resolve(__dirname, 'dist'),        filename:
'bundle.[contenthash].js',
        publicPath: '/',
    },
    mode: 'development',
    resolve: {
        extensions: ['.ts', '.tsx', '.js'],
```

```
        fallback: {
            "process": require.resolve("process/browser")
        }
    },
    devServer: {
        historyApiFallback: true,
        port: 3000,
        static: path.resolve(__dirname, 'public'),
        hot: true,
    },
    module: {
        rules: [
            {
                test: /\.tsx?$/,
                use: 'ts-loader',
                exclude: /node_modules/,
            },
            {
                test: /\.css$/,
                use: ['style-loader', 'css-loader'],
            },
            {
                test: /\.(png|jpg|jpeg|gif|svg)$/i,                type:
'asset/resource',
            },
        ],
    },
    plugins: [
        new CleanWebpackPlugin(),
        new HtmlWebpackPlugin({
            template: './public/index.ejs',
            filename: 'index.html',            title: 'Treinamento React Curio', // Esse valor
será usado no <title>            inject: false // Opcional: se quiser controlar os scripts
manualmente (como no seu ejs)
        }),
        new webpack.ProvidePlugin({
            process: 'process/browser',
```

```
        })
    ]
};
```

Na Pasta `public` adicione o arquivo `config.json` com o modelo de configuracao a baixo:

```
{
    "service": {        "url":
"https://srvd1.dev.evologica.com.br/cxClient/cxIsapiClient.dll/gatewayJSONBalanced?version=v3",
        "server": "192.168.0.59",        "system": "38", //Faz a conexao para o nosso sistema
Treinamento
        "port": "5370"
    },
    "accessToken": "3ab910293193f6d5b2d09c368959be9d9cb7ed68_50",
    "resources": {        "get":
"https://srvd1.dev.evologica.com.br/cxClient/cxIsapiClient.dll/getpr",        "put":
"https://srvd1.dev.evologica.com.br/cxClient/cxIsapiClient.dll/putpr"
    },
    "logs": true
}
```

Configurando os Arquivos Core do Curio

Agora no seu diretorio `src` crie um módulo `core` dedicado para integração com o Curio. Este servirá como ponto de entrada para toda a funcionalidade relacionada ao Curio em sua aplicação React.

1. **Crie um arquivo `Session.ts` com a classe Session:**

```
// src/curio/Session.ts
import { MainUseCase } from '@curio/client'
export class Session extends MainUseCase {
  module: number | undefined
  storageToken: string | undefined
}
```

Esse arquivo vai ser a base para mapear as informacoes de Sessao de usuarios do Curio

2. **Crie um arquivo `SessionManager.ts` com a classe SessionManager:**

```
// src/curio/SessionManager.tsimport { MainUseCase, RequestDriver, SecurityManager,
UseCaseMessageType } from '@curio/client'import { ConnectionError, DestinataryNotFoundError }
from '@curio/client/errors'
import { ABORT } from '@curio/client/utils/constants'
import { Session } from './Session'
export interface ConfigService {
    url: string
    server: string
    system: number
    port: number
    module: number
    version: number
}
export interface Service {
    service?: string
    port: string
    server: string
    system: string
    isapi?: string
}
export const STORAGEKEY = 'br.com.cognera'
export class SessionManager extends SecurityManager {    public session: MainUseCase |
undefined
    private _service: ConfigService
    private _driver: RequestDriver
```

```
    constructor(service: any, driver: RequestDriver) {
        super(service, driver)
        this._service = service
        this._driver = driver
    }
    public async openMainUseCase<U extends typeof MainUseCase>(
        username: string,
        password: string,
        constructor?: U
    ) {        this.session = await super.openMainUseCase(username, password,
constructor)
        this.session!.addListener(ABORT, () => this._unauthenticate(false))
this.session!.addListener(UseCaseMessageType.RESPONSE, (message: any) => {            if
(
                message.error instanceof DestinataryNotFoundError ||
message.error instanceof ConnectionError
            ) {
                this._unauthenticate(true)
            }
        })
        return this.session! as InstanceType<U>
    }
    public connectMainUseCase(mainUseCase: MainUseCase) {        this.session =
mainUseCase
        this.session.addListener(ABORT, () => this._unauthenticate(false))
this.session.addListener(UseCaseMessageType.RESPONSE, (message: any) => {            if
(
                message.error instanceof DestinataryNotFoundError ||
message.error instanceof ConnectionError
            ) {
                this._unauthenticate(true)
            }
        })
        return super.connectMainUseCase(mainUseCase)
    }
    public anonymousSession() {
        return new Session(0, this._service, this._driver)
```

```
    }
    private _unauthenticate(expired: boolean) {
        this.session = undefined
        if (expired) {
            localStorage.removeItem(STORAGEKEY)
        }
    }
}
```

3. **Crie um arquivo `index.ts` para configuração do uso de SessionManager:**

```
// src/curio/index.ts
import { FetchLink } from '@curio/client/links'import { V3RequestParser } from
'@curio/client/parsers'
import { createV3Validator } from '@curio/client/validators'import { Session } from './Session'
import { Service as SV, SessionManager } from './SessionManager'
type Service = SV
export interface Config {
    service: {
        url: string
        server: string
        system: number
        port: number
    }
    accessToken: string
    resources: {
        get: string
        put: string
        viewer: string
    }
    logs: boolean
}
const createSessionManager = (configuration: Config) => {    const link = new
FetchLink(configuration.service)    const requestParser = new
V3RequestParser(configuration.service)
    return new SessionManager(configuration.service, request => {        return
```

```
Promise.resolve(request)
            .then(requestParser.parse)
            .then(JSON.stringify)
            .then(link.parse)
            .then(link.post)
            .then(response => response.json())
            .then(json => json)
            .then(createV3Validator(request).validate)
    })
}
export const getSessionManager = async () => {
    const CONFIG_PATH = './config.json'
    const config = await (await fetch(CONFIG_PATH)).json()    const sessionManager =
createSessionManager(config)
    return sessionManager
}
export { Service, Session }
```

Integrando com React

Agora, com a nossa conexao ao Curio configurada precisamos implementar um metodo para criar a Sessao referente ao nosso usuario.

1. **Crie uma pasta `api` no seu diretorio `src` e nela um arquivo `session.ts`**

```
import { ConnectionError, DestinataryNotFoundError } from '@curio/client/errors'import {
Session, getSessionManager } from '../core'
import { STORAGEKEY } from '../core/SessionManager'
export interface SessionParams {
    channel: string
    server: string
    port: string
    system: string
}
export interface LoginParams {
```

```
    username: string
    password?: string
}
export async function login(params: LoginParams) {
    const { username, password } = params
    try {
        const sessionManager = await getSessionManager()        const session = await
sessionManager.openMainUseCase(
            username,
            password ?? '',
            Session
        )
        session.module = 0
        session.storageToken = STORAGEKEY        sessionStorage.setItem(STORAGEKEY,
session.token.toString())
        return session
    } catch (error) {
        return error as Error
    }
}
export async function connect(token: string) {
    try {
        const sessionManager = await getSessionManager()
sessionManager.connectMainUseCase(            new Session(token, sessionManager.service,
sessionManager.driver)
        )
        await sessionManager.session!.timeoutCheck()        const session =
sessionManager.session! as Session
        session.module = 0
        session.storageToken = STORAGEKEY        sessionStorage.setItem(STORAGEKEY,
session.token.toString())
        return session
    } catch (error) {
        if (error instanceof DestinataryNotFoundError) {            return error as
DestinataryNotFoundError
        }
        if (error instanceof ConnectionError) {            return error as
```

```
ConnectionError
        }
    }
}
export async function logoutSession(session?: Session) {
    if (session) {
        session.abort()
    }
    localStorage.clear()
}
```

2. **Crie uma pasta `context` em `src` para definir um Contexto de autenticacao Curio:**

```
// src/contexts/CurioContext.tsximport React, { createContext, useContext, useEffect, useState }
from 'react'
import { Session } from '../core'
import {
    LoginParams,
    connect as connectRequest,
    login as loginRequest,
} from '../api/session'
import { STORAGEKEY } from '../core/SessionManager'import { DestinataryNotFoundError } from
'@curio/client/errors'
interface AuthContextType {
    session?: Session
    isLoading: boolean
    isAuth: boolean
    login: (params: LoginParams) => Promise<void>
    logout: () => void
}
const CurioContext = createContext<AuthContextType | undefined>(undefined)
export const useAuth = () => {
    const context = useContext(CurioContext)
    if (!context) {        throw new Error('useAuth deve ser utilizado dentro de um
AuthProvider')
    }
```

```
    return context
}
export const CurioProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
const [session, setSession] = useState<Session>()    const [isAuth, setIsAuth] =
useState<boolean>(false)
    const [isLoading, setIsLoading] = useState<boolean>(false)
    useEffect(() => {
        connect()
    }, [])
    const connect = async () => {
        setIsLoading(true)
        const token = sessionStorage.getItem(STORAGEKEY)
        if (token) {
            setIsAuth(true)
            const sessionAPI = await connectRequest(token)            if (sessionAPI instanceof
Session) {
                setSession(sessionAPI)
                setIsAuth(true)            } else if (sessionAPI instanceof
DestinataryNotFoundError) {
                logout()
            }
        } else {
            logout()
        }
        setIsLoading(false)
    }
    const login = async (params: LoginParams) => {
        setIsLoading(true)
        setIsAuth(false)
        const sessionAPI = await loginRequest(params)        if (sessionAPI instanceof Error)
{
            logout()
            alert('Nao foi possivel realizar o login!')
        } else {
            setSession(sessionAPI)
            setIsAuth(true)
            alert('Login realizado com sucesso!')
```

```
        }
        setIsLoading(false)
    }
    const logout = () => {
        setIsAuth(false)
        setSession(undefined)
        sessionStorage.clear()
        localStorage.clear()
        setIsLoading(false)
    }
    return (        <CurioContext.Provider value={{ isAuth, session, isLoading, login, logout
}}>
            {children}
        </CurioContext.Provider>
    )
}
```

3. **Envolva sua aplicação React com o CurioProvider:**

```
import React from 'react';
import App from './App';
import { createRoot } from 'react-dom/client';
const root = createRoot(document.getElementById('root')!);
root.render(
    <App />
);
```

```
// src/App.tsx
import React from 'react'
import App from './App'
import { CurioProvider } from './contexts/CurioContext'
export default function App() {
  return (
    <CurioProvider>
      <App />
```

```
    </CurioProvider>
)
}
```

4. **Usando o Curio em um componente do seu Projeto:**

Para esse exemplo estamos utilizando uma pagina que deve fazer o login

```
// src/pages/Login/Login.tsx
import React, { useState } from 'react'
import { useAuth } from '../../context/CurioContext'
const Login: React.FC = () => {
    const [username, setUsername] = useState('')    const [password, setPassword] =
useState('')
    const { login, isLoading } = useAuth()    const [error, setError] = useState<string |
null>(null)
    const handleLogin = async (e: React.FormEvent) => {
        e.preventDefault()
        setError(null)
        try {
            await login({ username, password })
        } catch (err) {            setError('Falha no login: ' + (err instanceof Error ?
err.message : String(err)))
        }
    }
    if (isLoading) {
        return <div>Carregando Curio...</div>
    }
    return (
        <div className="login-container">            <form
onSubmit={handleLogin}>
                <h2>Login</h2>
                {error && <div className="error">{error}</div>}
                <div className="form-group">                    <label
htmlFor="username">Usuário:</label>
                    <input
```

```
                        id="username"
                        type="text"
value={username}                        onChange={(e) =>
setUsername(e.target.value)}
                        required
                    />
                </div>
                <div className="form-group">                    <label
htmlFor="password">Senha:</label>
                    <input
                        id="password"
type="password"
                        value={password}                        onChange={(e) =>
setPassword(e.target.value)}
                        required
                    />
                </div>
                <button type="submit" disabled={isLoading}>                    {isLoading ?
'Entrando...' : 'Entrar'}
                </button>
            </form>
        </div>
    )
}
export default Login
```

Pronto! Com esses passos, agora temos uma interface em React utilizando o Curio que executa o MainUseCase para poder gerar uma sessão de usuário!

Implementando biblioteca de componentes para estilização MaterialUI

Agora o nosso projeto Curio + React vamos dar inicio a utilizacao da biblioteca de components MaterialUI

1. **Instalando dependências necessárias**

```
yarn add @mui/material @emotion/react @emotion/styled
yarn add @fontsource/roboto
```

2. **Configure o tema do Material UI:**

No seu diretório `src` cria uma pasta `theme` e nela um arquivo de tema para personalizar a aparência do Material UI:

```
import { createTheme } from '@mui/material/styles';
// Crie um tema personalizado
const theme = createTheme({
    palette: {
        primary: {            main: '#00b07b', // Cor primária (pode ser ajustada conforme sua marca)
            light: '#4FD1DF',
            dark: '#005e42',
        },
        secondary: {
            main: '#316b52', // Cor secundária
        },
```

```
        error: {
            main: '#f44336',
        },
        background: {
            default: '#f5f5f5',
        },
    },
    typography: {
        fontFamily: '"Roboto", "Helvetica", "Arial", sans-serif',
        h1: {
            fontSize: '2.5rem',
            fontWeight: 500,
        },
        h2: {
            fontSize: '2rem',
            fontWeight: 500,
        },
        button: {            textTransform: 'none', // Evita que os botões fiquem em
maiúsculas
        },
    },
    components: {
        MuiButton: {
            styleOverrides: {
                root: {
                    borderRadius: 8,
                },
            },
        },
    },
});
export default theme;
```

3. **Aplique o tema ao seu aplicativo:**

Modifique o arquivo `App.tsx` para incluir o provedor de tema:

```
```

```
// src/App.tsx
import React, { Suspense } from 'react';import { ThemeProvider } from '@mui/material/styles';
import CssBaseline from '@mui/material/CssBaseline';
import theme from './theme';
import Login from './pages/Login/Login';
import { CurioProvider } from './context/CurioContext';
export default function App() {
    return (
        <CurioProvider>
            <ThemeProvider theme={theme}>                {/* CssBaseline normaliza os estilos
entre navegadores */}
                <CssBaseline />                <Suspense
fallback={<div>Carregando...</div>}>
                    <Login />
                </Suspense>
            </ThemeProvider>
        </CurioProvider>
    )
}
```

Refatorando a Página de Login com Material UI

Vamos refatorar nossa página de login para usar componentes do Material UI, melhorando a aparência e a experiência do usuário.

```
// src/pages/Login/Login.tsx
import React, { useState } from 'react';import { useAuth } from '../../context/CurioContext';
import {
    Box,
    Button,
    CircularProgress,
    Container,
    TextField,
    Typography,
```

```
    Alert,
    Paper,
    Avatar
} from '@mui/material';
const Login: React.FC = () => {
    const [username, setUsername] = useState('');    const [password, setPassword] =
useState('');
    const { login, isLoading } = useAuth();    const [error, setError] = useState<string |
null>(null);
    const handleLogin = async (e: React.FormEvent) => {
        e.preventDefault();
        setError(null);
        try {
            await login({ username, password });
        } catch (err) {            setError('Falha no login: ' + (err instanceof Error ?
err.message : String(err)));
        }
    };
    return (
        <Container component="main" maxWidth="xs">
            <Box
                sx={{
                    marginTop: 8,
                    display: 'flex',                    flexDirection:
'column',
                    alignItems: 'center',
                }}
            >
                <Paper
                    elevation={3}
                    sx={{
                        padding: 3,                        display:
'flex',
                        flexDirection: 'column',                        alignItems:
'center',
                        width: '100%',
                    }}
```

```
                >                    <Avatar sx={{ m: 1, width: 75, height: 75, bgcolor:
'primary.dark' }} src='/assets/curio.png' />                    <Typography component="h1"
variant="h5">
                        Login
                    </Typography>
                    {error && (                        <Alert severity="error" sx={{ width:
'100%', mt: 2 }}>
                            {error}
                        </Alert>
                    )}
                    <Box component="form" onSubmit={handleLogin} sx={{ mt: 1, width: '100%'
}}>
                        <TextField
margin="normal"
                            required
fullWidth
                            id="username"
label="Usuário"
                            name="username"
autoComplete="username"
                            autoFocus
value={username}                            onChange={(e) =>
setUsername(e.target.value)}
disabled={isLoading}
                        />
                        <TextField
margin="normal"
                            fullWidth
name="password"
                            label="Senha"
type="password"
                            id="password"                            autoComplete="current-
password"
                            value={password}                            onChange={(e) =>
setPassword(e.target.value)}
disabled={isLoading}
                        />
```

```
                        <Button
type="submit"
                            fullWidth
variant="contained"
                            sx={{ mt: 3, mb: 2 }}
disabled={isLoading}
                        >                            {isLoading ?
(                                <CircularProgress size={24} color="inherit"
/>
                            ) : (
'Entrar'
                            )}
                        </Button>
                    </Box>
                </Paper>
            </Box>
        </Container >
    );
};
export default Login;
```