**Frontend BabyTrackMaster – Guía paso a paso en Windows 11 + Esqueleto inicial**

Objetivo: Dejar **Windows 11** listo y un esqueleto funcional en React (Create React App + Material UI + React Router + Axios + JWT) para conectar con tus microservicios locales.

**1) Prerrequisitos en Windows 11**

**1.1 Instalar herramientas base**

Tienes varias opciones. Elige **A) Winget** (nativo Windows 11) o **B) Chocolatey**.

**A) Con Winget (recomendado)**

# Abrir PowerShell como Administrador

winget install --id Git.Git -e

winget install --id Microsoft.VisualStudioCode -e

# Node LTS con winget

winget install --id OpenJS.NodeJS.LTS -e

# Verificar

node -v

npm -v

git --version

B) Con Chocolatey (alternativa)

# Abrir PowerShell como Administrador, habilitar scripts si hace falta

Set-ExecutionPolicy Bypass -Scope Process -Force

[System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

# Instalar Chocolatey

Set-ExecutionPolicy Bypass -Scope Process -Force;

iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

# Instalar paquetes

choco install -y git vscode nodejs-lts

# Verificar

node -v

npm -v

git --version

**Consejo**: Si prefieres gestionar versiones de Node, usa **nvm-windows**:

winget install --id CoreyButler.NVMforWindows -e

nvm install lts

nvm use lts

node -v

npm -v

1.2 Configuración útil de Git en Windows

git config --global core.longpaths true

# Opcional: nombre y correo

git config --global user.name "Tu Nombre"

git config --global user.email "tu@correo.com"

**1.3 Editor**

* **VS Code** ya instalado (winget/choco). Extensiones recomendadas:
  + ESLint, Prettier, GitLens, Material Icon Theme.

**2) Crear el proyecto React (Create React App)**

La documentación del proyecto propone CRA para arrancar.

# Ubicación de trabajo (ajústala a tu preferencia)

mkdir $env:USERPROFILE\Proyectos\babytrackmaster

cd $env:USERPROFILE\Proyectos\babytrackmaster

# Crear la app React

npx create-react-app frontend-baby

cd frontend-baby

# Arrancar en modo desarrollo (abrirá http://localhost:3000)

npm start

Si PowerShell bloquea scripts, ejecuta:

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

3) Instalar librerías del proyecto

# UI (Material UI) + iconos

npm i @mui/material @emotion/react @emotion/styled @mui/icons-material

# Ruteo

npm i react-router-dom

# HTTP + JWT

npm i axios jwt-decode

# Fechas (opcional)

npm i dayjs

4) Estructura de carpetas propuesta

frontend-baby/

└─ src/

├─ api/

│ └─ http.js # instancia Axios con interceptores JWT

├─ components/

│ └─ Navbar.jsx # barra superior de navegación

├─ context/

│ └─ AuthContext.jsx # login/logout, estado del usuario

├─ pages/

│ ├─ Login.jsx

│ ├─ Register.jsx

│ ├─ Dashboard.jsx

│ ├─ Cuidados.jsx

│ ├─ Gastos.jsx

│ ├─ Hitos.jsx

│ ├─ Diario.jsx

│ ├─ Citas.jsx

│ ├─ Rutinas.jsx

│ ├─ Perfil.jsx

│ └─ NotFound.jsx

├─ routes/

│ └─ PrivateRoute.jsx

├─ theme/

│ └─ theme.js # tema MUI (opcional, modo oscuro)

├─ utils/

│ └─ auth.js # helpers de autenticación

├─ App.jsx

├─ index.js

└─ index.css

**5) Variables de entorno (.env) en Windows**

Crea un archivo **.env** en la **raíz** del proyecto frontend-baby (misma carpeta que package.json). **CRA** solo expone variables que empiezan por REACT\_APP\_.

# Ajusta puertos y paths reales de tus APIs

REACT\_APP\_API\_USUARIOS\_URL=http://localhost:8080/api/v1

REACT\_APP\_API\_CUIDADOS\_URL=http://localhost:8081/api/v1

REACT\_APP\_API\_GASTOS\_URL=http://localhost:8082/api/v1

REACT\_APP\_API\_HITOS\_URL=http://localhost:8083/api/v1

REACT\_APP\_API\_CITAS\_URL=http://localhost:8084/api/v1

REACT\_APP\_API\_DIARIO\_URL=http://localhost:8085/api/v1

REACT\_APP\_API\_RUTINAS\_URL=http://localhost:8086/api/v1

REACT\_APP\_API\_BFF\_URL=http://localhost:8087/api/v1

**Recarga**: tras crear o cambiar el .env, reinicia npm start para que se apliquen los cambios.

**CORS**: como llamarás a múltiples microservicios desde http://localhost:3000, habilita CORS en cada backend (o usa el api-gateway cuando pases a entornos más cerrados/producción).

**6) Código base – Archivos clave (copiar/pegar)**

Crea estos archivos con el contenido exacto. Funcionan en Windows igual que en Linux/macOS.

**6.1 src/index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import { AuthProvider } from './context/AuthContext';

import App from './App';

import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<BrowserRouter>

<AuthProvider>

<App />

</AuthProvider>

</BrowserRouter>

</React.StrictMode>

);

6.2 src/App.jsx

import React from 'react';

import { Routes, Route } from 'react-router-dom';

import { CssBaseline, Container } from '@mui/material';

import Navbar from './components/Navbar';

import PrivateRoute from './routes/PrivateRoute';

import Login from './pages/Login';

import Register from './pages/Register';

import Dashboard from './pages/Dashboard';

import Cuidados from './pages/Cuidados';

import Gastos from './pages/Gastos';

import Hitos from './pages/Hitos';

import Diario from './pages/Diario';

import Citas from './pages/Citas';

import Rutinas from './pages/Rutinas';

import Perfil from './pages/Perfil';

import NotFound from './pages/NotFound';

export default function App() {

return (

<>

<CssBaseline />

<Navbar />

<Container sx={{ mt: 3, mb: 6 }}>

<Routes>

<Route path="/login" element={<Login />} />

<Route path="/registro" element={<Register />} />

<Route element={<PrivateRoute />}>

<Route path="/" element={<Dashboard />} />

<Route path="/cuidados" element={<Cuidados />} />

<Route path="/gastos" element={<Gastos />} />

<Route path="/hitos" element={<Hitos />} />

<Route path="/diario" element={<Diario />} />

<Route path="/citas" element={<Citas />} />

<Route path="/rutinas" element={<Rutinas />} />

<Route path="/perfil" element={<Perfil />} />

</Route>

<Route path="\*" element={<NotFound />} />

</Routes>

</Container>

</>

);

}

6.3 src/context/AuthContext.jsx

import React, { createContext, useState, useEffect } from 'react';

import jwtDecode from 'jwt-decode';

import axios from 'axios';

export const AuthContext = createContext(null);

export function AuthProvider({ children }) {

const [token, setToken] = useState(localStorage.getItem('token'));

const [user, setUser] = useState(null); // aquí puedes guardar userId/roles si los incluyes en el JWT

useEffect(() => {

if (token) {

try {

const decoded = jwtDecode(token);

// decoded.exp está en segundos

if (decoded && decoded.exp && decoded.exp \* 1000 > Date.now()) {

setUser(decoded);

} else {

logout();

}

} catch (e) {

logout();

}

}

}, [token]);

const login = async (email, password) => {

const base = process.env.REACT\_APP\_API\_USUARIOS\_URL;

const { data } = await axios.post(`${base}/auth/login`, { email, password });

// asumiendo que devuelve { token: "..." }

localStorage.setItem('token', data.token);

setToken(data.token);

};

const register = async (nombre, email, password) => {

const base = process.env.REACT\_APP\_API\_USUARIOS\_URL;

await axios.post(`${base}/auth/register`, { nombre, email, password });

};

const logout = () => {

localStorage.removeItem('token');

setToken(null);

setUser(null);

};

return (

<AuthContext.Provider value={{ token, user, login, register, logout }}>

{children}

</AuthContext.Provider>

);

}

6.4 src/utils/auth.js

import jwtDecode from 'jwt-decode';

export function isTokenValid(token) {

if (!token) return false;

try {

const decoded = jwtDecode(token);

if (!decoded || !decoded.exp) return false;

return decoded.exp \* 1000 > Date.now();

} catch (e) {

return false;

}

}

6.5 src/routes/PrivateRoute.jsx

import React, { useContext } from 'react';

import { Navigate, Outlet, useLocation } from 'react-router-dom';

import { AuthContext } from '../context/AuthContext';

import { isTokenValid } from '../utils/auth';

export default function PrivateRoute() {

const { token } = useContext(AuthContext);

const location = useLocation();

if (!token || !isTokenValid(token)) {

return <Navigate to="/login" replace state={{ from: location }} />;

}

return <Outlet />;

}

6.6 src/api/http.js

import axios from 'axios';

const http = axios.create({

// baseURL se fijará por servicio al usar httpX (ver ejemplo abajo),

// aquí dejamos una instancia base genérica por si quieres un gateway más adelante.

});

// Interceptor para añadir JWT

http.interceptors.request.use((config) => {

const token = localStorage.getItem('token');

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

});

export default http;

// Instancias por servicio (opcional)

export const httpUsuarios = axios.create({ baseURL: process.env.REACT\_APP\_API\_USUARIOS\_URL });

export const httpCuidados = axios.create({ baseURL: process.env.REACT\_APP\_API\_CUIDADOS\_URL });

export const httpGastos = axios.create({ baseURL: process.env.REACT\_APP\_API\_GASTOS\_URL });

export const httpHitos = axios.create({ baseURL: process.env.REACT\_APP\_API\_HITOS\_URL });

export const httpCitas = axios.create({ baseURL: process.env.REACT\_APP\_API\_CITAS\_URL });

export const httpDiario = axios.create({ baseURL: process.env.REACT\_APP\_API\_DIARIO\_URL });

export const httpRutinas = axios.create({ baseURL: process.env.REACT\_APP\_API\_RUTINAS\_URL });

export const httpBff = axios.create({ baseURL: process.env.REACT\_APP\_API\_BFF\_URL });

// Interceptor JWT para todas

[httpUsuarios, httpCuidados, httpGastos, httpHitos, httpCitas, httpDiario, httpRutinas, httpBff]

.forEach((cli) => {

cli.interceptors.request.use((config) => {

const token = localStorage.getItem('token');

if (token) config.headers.Authorization = `Bearer ${token}`;

return config;

});

});

6.7 src/components/Navbar.jsx

import React, { useContext } from 'react';

import { AppBar, Toolbar, Typography, Button, Stack } from '@mui/material';

import { Link as RouterLink } from 'react-router-dom';

import { AuthContext } from '../context/AuthContext';

export default function Navbar() {

const { token, logout } = useContext(AuthContext);

return (

<AppBar position="static">

<Toolbar>

<Typography variant="h6" sx={{ flexGrow: 1 }}>

BabyTrackMaster

</Typography>

{token ? (

<Stack direction="row" spacing={1}>

<Button color="inherit" component={RouterLink} to="/">Dashboard</Button>

<Button color="inherit" component={RouterLink} to="/cuidados">Cuidados</Button>

<Button color="inherit" component={RouterLink} to="/gastos">Gastos</Button>

<Button color="inherit" component={RouterLink} to="/hitos">Hitos</Button>

<Button color="inherit" component={RouterLink} to="/diario">Diario</Button>

<Button color="inherit" component={RouterLink} to="/citas">Citas</Button>

<Button color="inherit" component={RouterLink} to="/rutinas">Rutinas</Button>

<Button color="inherit" component={RouterLink} to="/perfil">Perfil</Button>

<Button color="inherit" onClick={logout}>Salir</Button>

</Stack>

) : (

<Stack direction="row" spacing={1}>

<Button color="inherit" component={RouterLink} to="/login">Login</Button>

<Button color="inherit" component={RouterLink} to="/registro">Registro</Button>

</Stack>

)}

</Toolbar>

</AppBar>

);

}

6.8 src/pages/Login.jsx

import React, { useContext, useState } from 'react';

import { useNavigate, useLocation, Link as RouterLink } from 'react-router-dom';

import { Box, Button, TextField, Typography, Stack, Alert } from '@mui/material';

import { AuthContext } from '../context/AuthContext';

export default function Login() {

const { login } = useContext(AuthContext);

const navigate = useNavigate();

const location = useLocation();

const from = location.state?.from?.pathname || '/';

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [error, setError] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

setError('');

try {

await login(email, password);

navigate(from, { replace: true });

} catch (err) {

setError('Credenciales inválidas o servicio no disponible');

}

};

return (

<Box maxWidth={420} mx="auto" mt={6}>

<Typography variant="h5" mb={2}>Iniciar Sesión</Typography>

{error && <Alert severity="error" sx={{ mb: 2 }}>{error}</Alert>}

<form onSubmit={handleSubmit}>

<Stack spacing={2}>

<TextField label="Email" type="email" value={email} onChange={(e) => setEmail(e.target.value)} required />

<TextField label="Contraseña" type="password" value={password} onChange={(e) => setPassword(e.target.value)} required />

<Button type="submit" variant="contained">Entrar</Button>

</Stack>

</form>

<Typography mt={2}>

¿No tienes cuenta? <Button component={RouterLink} to="/registro">Regístrate</Button>

</Typography>

</Box>

);

}

6.9 src/pages/Register.jsx

import React, { useContext, useState } from 'react';

import { Box, Button, TextField, Typography, Stack, Alert } from '@mui/material';

import { useNavigate } from 'react-router-dom';

import { AuthContext } from '../context/AuthContext';

export default function Register() {

const { register } = useContext(AuthContext);

const navigate = useNavigate();

const [nombre, setNombre] = useState('');

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [repeat, setRepeat] = useState('');

const [error, setError] = useState('');

const [ok, setOk] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

setError('');

setOk('');

if (password !== repeat) {

setError('Las contraseñas no coinciden');

return;

}

try {

await register(nombre, email, password);

setOk('Registro correcto. Ya puedes iniciar sesión.');

setTimeout(() => navigate('/login'), 1000);

} catch (err) {

setError('No se pudo registrar el usuario');

}

};

return (

<Box maxWidth={480} mx="auto" mt={6}>

<Typography variant="h5" mb={2}>Crear Cuenta</Typography>

{error && <Alert severity="error" sx={{ mb: 2 }}>{error}</Alert>}

{ok && <Alert severity="success" sx={{ mb: 2 }}>{ok}</Alert>}

<form onSubmit={handleSubmit}>

<Stack spacing={2}>

<TextField label="Nombre" value={nombre} onChange={(e) => setNombre(e.target.value)} required />

<TextField label="Email" type="email" value={email} onChange={(e) => setEmail(e.target.value)} required />

<TextField label="Contraseña" type="password" value={password} onChange={(e) => setPassword(e.target.value)} required />

<TextField label="Repetir Contraseña" type="password" value={repeat} onChange={(e) => setRepeat(e.target.value)} required />

<Button type="submit" variant="contained">Registrarse</Button>

</Stack>

</form>

</Box>

);

}

6.10 Páginas de ejemplo (placeholders)

Crea páginas mínimas para el resto, así podrás verificar las rutas:

src/pages/Dashboard.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Dashboard(){

return <Typography variant="h4">Dashboard (conectará al api-bff)</Typography>;

}

src/pages/Cuidados.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Cuidados(){

return <Typography variant="h4">Cuidados</Typography>;

}

src/pages/Gastos.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Gastos(){

return <Typography variant="h4">Gastos</Typography>;

}

src/pages/Hitos.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Hitos(){

return <Typography variant="h4">Hitos</Typography>;

}

src/pages/Diario.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Diario(){

return <Typography variant="h4">Diario</Typography>;

}

src/pages/Citas.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Citas(){

return <Typography variant="h4">Citas</Typography>;

}

src/pages/Rutinas.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Rutinas(){

return <Typography variant="h4">Rutinas</Typography>;

}

src/pages/Perfil.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function Perfil(){

return <Typography variant="h4">Perfil</Typography>;

}

src/pages/NotFound.jsx

import React from 'react';

import { Typography } from '@mui/material';

export default function NotFound(){

return <Typography variant="h4">404 - No encontrado</Typography>;

}

6.11 src/index.css (opcional simple)

html, body, #root { height: 100%; }

\* { box-sizing: border-box; }

**7) Probar el login real en Windows**

1. Arranca **api-usuarios** en http://localhost:8080 y verifica endpoints:
   * POST /api/v1/auth/register
   * POST /api/v1/auth/login que devuelva { token: "..." } (ajusta el código si tu backend usa otra propiedad)
2. En el frontend, npm start, entra a http://localhost:3000/login.
3. Inicia sesión. Si el token es válido, te envía a / (Dashboard). Verás el **Navbar** completo y las **rutas privadas** activas.

**Firewall de Windows**: Si algo no carga en localhost:808X, permite el acceso cuando Windows lo solicite la primera vez que el microservicio abre el puerto.

**8) Conectar páginas con sus APIs**

* **Cuidados**: usa httpCuidados para listar/crear registros.
* **Gastos**: httpGastos con formulario + tabla de resumen mensual.
* **Dashboard**: httpBff para /dashboard cuando tengas api-bff listo.

El **JWT** se añade automáticamente por el interceptor en Authorization: Bearer <token>.

**9) Consejos (Windows 11)**

* **CORS**: añade @CrossOrigin o configura CORS global en Spring Security en cada microservicio.
* **.env**: tras modificarlo, **reinicia** npm start.
* **Rutas de archivos**: en Windows usa \\ en rutas si llegas a manipular ficheros locales desde scripts.
* **WSL (opcional)**: Si prefieres entorno Linux, instala "Subsistema de Windows para Linux" y trabaja con Ubuntu. El frontend en WSL también sirve en http://localhost:3000.
* **Puertos**: tus servicios corren en 8080-8087 y React en 3000. No uses puertos ocupados por otras apps.

**10) Próximos pasos guiados**

1. Añadir **snackbars** de éxito/error (Material UI) como pide la doc.
2. Formularios de cada módulo según wireframes.
3. **DataGrid** (MUI) para listados y filtros.
4. Tema MUI con **modo oscuro**.
5. Conectar **Dashboard** al api-bff.

Fin de la guía para Windows 11.