Logo da Casa da Criança Batuíra

Logo da Faculdade Processus

Projeto de Consultoria para Desenvolvimento de Bot de Atendimento via WhatsApp: Casa da Criança Batuira

Brasília 2026

Resumo

O presente trabalho visa apresentar um relatório de consultoria para o desenvolvimento de um bot de atendimento via WhatsApp para a Casa da Criança Batuira e Casa da Mãe Batuira, Organizações da Sociedade Civil (OSC) que promovem acolhimento a crianças, adolescentes e famílias em situação de vulnerabilidade. A consultoria em TI teve como objetivos desenvolver um sistema completo de atendimento automatizado, composto por um backend em Python (FastAPI), um frontend web em Next.js/TypeScript e integração com a Evolution API para comunicação via WhatsApp. O sistema permite que um único número de WhatsApp receba todas as mensagens, apresente um menu interativo de setores e direcione os contatos para atendentes humanos através de um painel web centralizado com login individual, fila de espera, chat em tempo real, configurações dinâmicas e editor visual de fluxos de atendimento. O projeto buscou entregar uma solução rob usta, escalável, econômica e de fácil manutenção, atendendo às necessidades da instituição e agregando valor ao seu atendimento.

Alunos:
Cleude Rodrigues Machado / (61) 99992-9820
Luís Felipe da Silva Barreto / (61) 98491-2580
Raphael Diesel Brasileiro / (61) 98179-0661
Fábio William Pereira da Rocha / (61) 98484-5927

Coordenadores:
Henderson Matsuura Sanches

Responsável da Casa da Criança Batuira:
Giovanna Queiroz, recebido no dia 26 de Março de 2026

Apresentação da Disciplina

Este relatório é o trabalho final para aprovação na disciplina Atividade de Extensão do curso Sistemas de Informação/Análise e Desenvolvimento de Sistemas da Faculdade Processus, sob orientação dos professores coordenadores, e que possibilita o contato e interferência direta do corpo acadêmico para além dos muros da faculdade, prestando consultoria tecnológica a uma organização da sociedade civil.

Cronograma de Atividades

Atividade	Data/Período
Reunião inicial e levantamento de requisitos	17/02/2026
Análise de requisitos e divisão de tarefas	18/02/2026 a 28/02/2026
Desenvolvimento do backend (FastAPI + SQLite)	01/03/2026 a 31/03/2026
Desenvolvimento do frontend (Next.js + React)	15/03/2026 a 20/04/2026
Integração com Evolution API (WhatsApp)	10/04/2026 a 30/04/2026
Implementação da Engine de Fluxo Visual	01/05/2026 a 18/05/2026
Testes e ajustes	19/05/2026 a 05/06/2026
Deploy (Fly.io + Vercel + Docker)	06/06/2026 a 12/06/2026
Redação do projeto	13/06/2026 a 22/06/2026
Revisão e entrega do projeto final	25/06/2026

Sumário

1. Introdução

As Organizações da Sociedade Civil (OSCs) têm histórica contribuição para as sociedades modernas, atuando em áreas como saúde, serviço social, educação e assistência. No Brasil, são mais de 780 mil OSCs, sendo quase 40 mil voltadas ao serviço social (IPEA, 2023). É nesse contexto que a Casa da Criança Batuira se insere, promovendo acolhimento, desenvolvimento e apoio a crianças, adolescentes e famílias em situação de vulnerabilidade na região do Distrito Federal.

Em um universo globalizado, onde o acesso à Internet já abrange cerca de 90% dos lares brasileiros (MINISTÉRIO DA CASA CIVIL, 2022), a comunicação digital tornou-se essencial para qualquer organização. O WhatsApp, presente em 99% dos smartphones brasileiros, é o principal canal de comunicação instantânea do país. Nesse cenário, a Casa da Criança Batuira identificou a necessidade de profissionalizar e automatizar seu atendimento via WhatsApp, que antes era feito de forma manual e descentralizada.

O presente projeto propõe o desenvolvimento de um bot de atendimento inteligente, denominado Batuira Bot, capaz de receber todas as mensagens em um único número de WhatsApp, apresentar um menu interativo de setores, direcionar os contatos para os atendentes responsáveis e gerenciar todo o ciclo de atendimento através de um painel web centralizado. O sistema contempla também a Casa da Mãe Batuira como segunda instituição atendida.

Como metodologia de abordagem, foi feito levantamento de requisitos com entrevistas e troca direta com o cliente, seguindo padrões de Engenharia de Software e metodologia ágil (Soares, 2004). O desenvolvimento seguiu o modelo iterativo, com entregas parciais validadas pela instituição.

2. Objetivo

O objetivo versa sobre qual horizonte se pretende alcançar com o projeto, definindo tanto o escopo geral quanto as etapas específicas para obter o resultado final.

2.1 Objetivo Geral

Este trabalho busca desenvolver e entregar um sistema completo de atendimento via WhatsApp para a Casa da Criança Batuira, construindo uma solução robusta, simples, escalável, manutenível e funcional, utilizando tecnologias modernas e consagradas, tanto por sua segurança quanto por facilidade de uso e economia.

2.2 Objetivos Específicos

- Compreender as necessidades de atendimento da instituição

- Desenvolver um bot de WhatsApp com menu interativo de setores e instituições
- Criar um painel web para gerenciamento de atendentes, filas e conversas
- Implementar sistema de autenticação seguro com Supabase Auth (JWT ES256)
- Desenvolver engine de fluxo visual para configuração do bot sem código
- Implementar sistema de API Keys para integrações externas
- Preparar infraestrutura de deploy para produção (Fly.io, Vercel, Docker)
- Garantir configurações dinâmicas sem necessidade de reiniciar o servidor

3. Produto da Consultoria

O produto do trabalho é um sistema completo de atendimento via WhatsApp, composto por três camadas principais: 1) um backend em Python com FastAPI que gerencia toda a lógica de negócio, webhook e banco de dados; 2) um frontend web em Next.js/TypeScript que oferece o painel de atendimento, configurações e editor de fluxos; 3) integração com a Evolution API para envio e recebimento de mensagens via WhatsApp.

3.1 Conceito

O conceito central do Batuira Bot é simples: um único número de WhatsApp recebe todas as mensagens destinadas à instituição. Ao receber uma mensagem, o bot apresenta automaticamente um menu de seleção de instituição (Casa da Criança Batuira ou Casa da Mãe Batuira), seguido de um menu de setores (Financeiro, Pedagógico, Administrativo, Assistência Social, entre outros configuráveis). O contato escolhe digitando o número correspondente, e a conversa é direcionada para a fila do setor escolhido.

Um atendente humano então visualiza a fila no painel web, assume a conversa e responde diretamente pelo painel. O bot notifica automaticamente o contato com o nome do atendente. Todo o histórico de mensagens é armazenado, permitindo rastreabilidade e, ao final, uma pesquisa de satisfação (CSAT) é enviada automaticamente.

A escolha por menu numerado (1, 2, 3, 4) ao invés de botões interativos do WhatsApp foi intencional: botões da Evolution API têm limite de 3 opções e nem sempre funcionam em todas as versões do WhatsApp. O texto numerado funciona universalmente e é facilmente expansível.

3.2 Arquitetura e Tecnologias

A arquitetura do sistema segue o padrão cliente-servidor, com separação clara entre backend e frontend. A tabela a seguir apresenta as tecnologias utilizadas em cada camada:

Camada	Tecnologia	Proporção no Código
Backend	Python 3.13, FastAPI, SQLite	34,2%
Frontend	TypeScript, Next.js 14, React 18, Tailwind CSS, Zustand	63,5%
Autenticação	Supabase Auth (JWT ES256)	—
WhatsApp	Evolution API	—
Scripts/Infra	Shell, Batch, VBScript, Docker	2,3%

3.3 Fluxo de Atendimento

O fluxo completo de uma mensagem recebida segue as seguintes etapas:

1) Contato envia mensagem para o número WhatsApp institucional
2) Evolution API dispara POST no webhook /api/whatsapp/webhook
3) Backend verifica o status da conversa no banco SQLite
4) Se não existe conversa: cria registro e envia menu de instituição
5) Contato escolhe instituição (1 ou 2) e recebe menu de setores
6) Contato escolhe setor — conversa muda para status 'waiting' (aguardando)
7) Atendente visualiza a fila no painel web (/atendimento)
8) Atendente clica 'Assumir' — POST /api/conversations/{id}/assign
9) Bot envia mensagem automática ao contato com nome e setor do atendente
10) Atendente troca mensagens pelo painel até encerrar o atendimento
11) Ao encerrar: sistema envia pesquisa de satisfação CSAT (1 a 4 estrelas)

A tabela a seguir resume os estados possíveis de uma conversa no sistema:

Status	Significado
pending_menu	Contato enviou mensagem mas ainda não escolheu setor
waiting	Escolheu setor, aguardando atendente disponível
active	Atendente assumiu e está em atendimento
closed	Conversa encerrada

O contato pode digitar '0', 'menu' ou 'voltar' a qualquer momento para reiniciar a conversa e voltar ao menu inicial.

4. Desenvolvimento

O desenvolvimento do projeto foi dividido em quatro grandes frentes: backend, frontend, engine de fluxo e banco de dados. A seguir, cada uma é detalhada.

4.1 Backend (Python / FastAPI)

O backend foi construído com FastAPI e Python 3.13, utilizando SQLite como banco de dados local. A estrutura de arquivos segue uma organização modular:

- **main.py** — Ponto de entrada da aplicação, inicializa banco e servidor Uvicorn
- **api/routes.py** — Todos os endpoints REST: webhook, setores, atendentes, conversas, dashboard, API keys, fluxos
- **agents/whatsapp_agent.py** — Agente de envio de mensagens via Evolution API com templates parametrizados
- **engine/flow_engine.py** — Engine de interpretação de fluxos visuais
- **database.py** — DDL das tabelas e seed inicial (4 setores padrão)
- **utils/** — Autenticação JWT (Supabase ES256), sanitização de telefone, gestão de settings em tempo real

O backend expõe 15 endpoints REST, incluindo webhook para recepção de mensagens, CRUD de setores e atendentes, gerenciamento de conversas, dashboard com estatísticas e geração de API keys externas. A tabela completa de endpoints:

Método	Rota	Auth	Descrição
GET	/api/health	Não	Status das integrações
GET/PATCH	/api/settings	PATCH sim	Configurações gerais
POST	/api/whatsapp/webhook	Não	Recebe mensagens da Evolution API
CRUD	/api/sectors	Escrita sim	Gerenciamento de setores
CRUD	/api/attendants	Escrita sim	Gerenciamento de atendentes
GET	/api/conversations	Sim	Lista conversas com filtros
POST	/api/conversations/{id}/assign	Sim	Atendente assume conversa
POST	/api/conversations/{id}/reply	Sim	Atendente responde
POST	/api/conversations/{id}/close	Sim	Encerra conversa
POST	/api/conversations/{id}/transfer	Sim	Transferência entre setores
GET	/api/dashboard/stats	Sim	Contadores por setor
CRUD	/api/api-keys	Sim	Gerencia API keys externas
CRUD	/api/flows	Escrita sim	Gerencia fluxos visuais

Método	Rota	Auth	Descrição
GET	/api/whatsapp/qrcode	Sim	QR Code de conexão

4.2 Frontend (Next.js / TypeScript)

O frontend foi construído com Next.js 14, React 18, TypeScript e Tailwind CSS, utilizando Zustand para gerenciamento de estado. As telas principais do painel são:

- **Dashboard (/)** — Visão geral com estatísticas e fila por setor
- **Login (/login)** — Autenticação via Supabase Auth
- **Atendimento (/atendimento)** — Painel principal do atendente com fila de espera e chat em tempo real
- **Editor de Fluxos (/flow)** — Editor visual drag-and-drop para configurar fluxos de atendimento
- **Configurações (/configuracoes)** — Setores, atendentes, API keys e configurações gerais

Todas as rotas são protegidas pelo componente AuthGuard, que redireciona para /login se não houver sessão ativa. O frontend comunica-se com o backend via módulo centralizado em lib/api.ts, que injeta automaticamente o token de autenticação em todas as requisições.

Imagem 2: Tela do Painel de Atendimento.

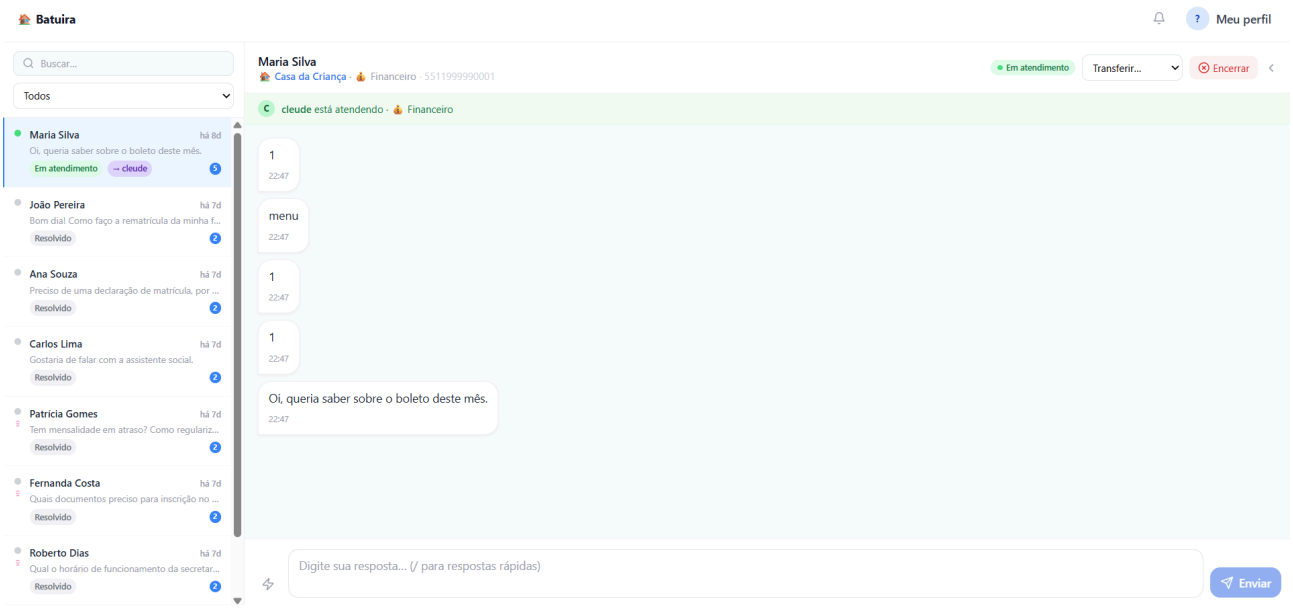
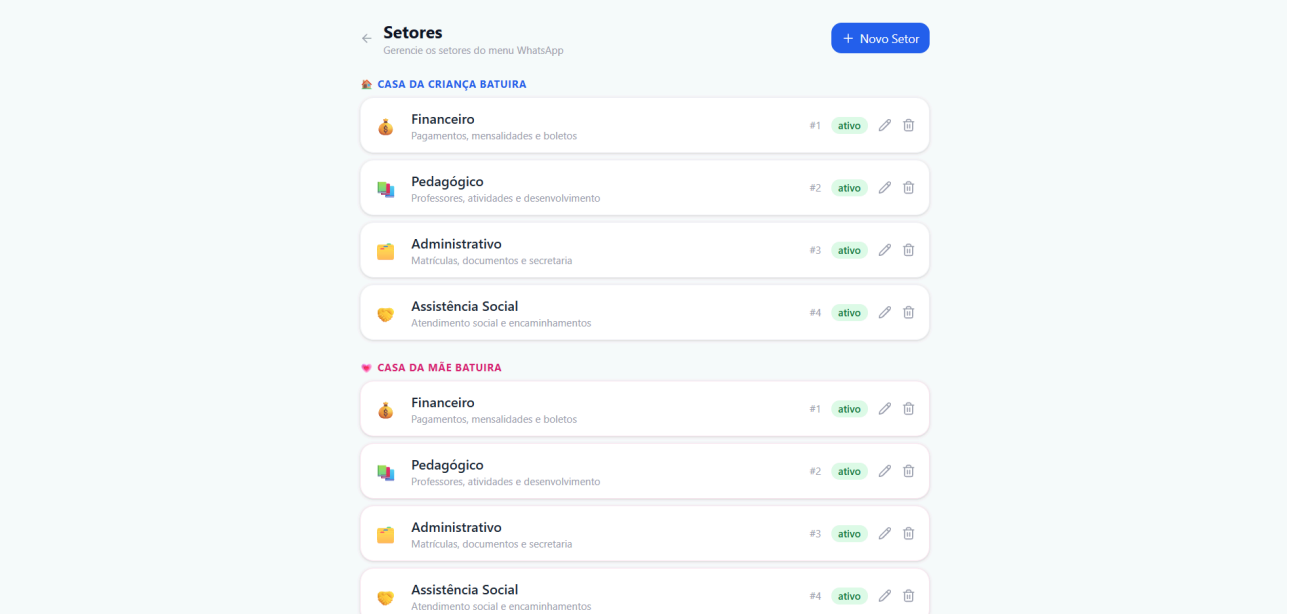


Imagem 4: Tela de Configurações de Setores.



4.3 Engine de Fluxo Visual

Um dos principais avanços do projeto foi a criação da FlowEngine, que permite controlar o bot por meio de fluxos visuais desenhados no painel web, substituindo mensagens fixas no código por nós configuráveis. Os tipos de nós suportados são:

- **TriggerNode** — Ponto de entrada do fluxo (gatilho de nova mensagem)
- **MessageNode** — Envia mensagem de texto com suporte a variáveis dinâmicas ({nome}, {setor}...)
- **ConditionNode** — Avalia condições e bifurca o fluxo (arestas 'Sim'/'Não')
- **ActionNode** — Executa ações no sistema (atribuir setor, encerrar conversa)

O FlowEngine interpreta os nós e arestas salvos no banco de dados (tabela flows, formato JSON), renderizando templates com contexto dinâmico. Isso permite que a equipe da instituição altere o comportamento do bot sem necessidade de programação.

4.4 Banco de Dados

O sistema utiliza SQLite como banco de dados, armazenado em um único arquivo local (data/batuira.db). A escolha por SQLite foi intencional: a escala atual da organização não justifica um servidor de banco separado, e o SQLite oferece zero configuração, backup simples (copia o arquivo) e SQL padrão compatível com migração futura para PostgreSQL.

Tabela	O que armazena
sectors	Setores do menu (nome, emoji, ordem, instituição, status ativo)

Tabela	O que armazena
attendants	Atendentes vinculados a setor, com e-mail e user_id Supabase
conversations	Cada conversa WhatsApp (status, setor, atendente, contato, timestamps)
messages	Histórico de mensagens com direção (in/out) e timestamps
api_keys	Chaves externas com prefixo btr_ e hash SHA-256
flows	Fluxos visuais serializados em JSON (nodes + edges)

O seed automático insere os 4 setores padrão (Financeiro, Pedagógico, Administrativo, Assistência Social) na primeira execução do sistema.

5. UX/UI — Interface do Painel

Para atingir as expectativas de um produto simples, funcional e resolutivo, foram aplicados conceitos de UX (User Experience) e UI (User Interface) no desenvolvimento do painel web. A interface foi projetada para ser intuitiva, permitindo que atendentes sem conhecimento técnico operem o sistema com facilidade.

Foram considerados três princípios fundamentais de design definidos por Donald Norman: visibilidade (o usuário consegue identificar o estado do sistema e suas opções de ação), consistência (botões e ícones representam exatamente suas funcionalidades) e pregnância (elementos visuais são auto-explicativos).

5.1 Análise de Similares

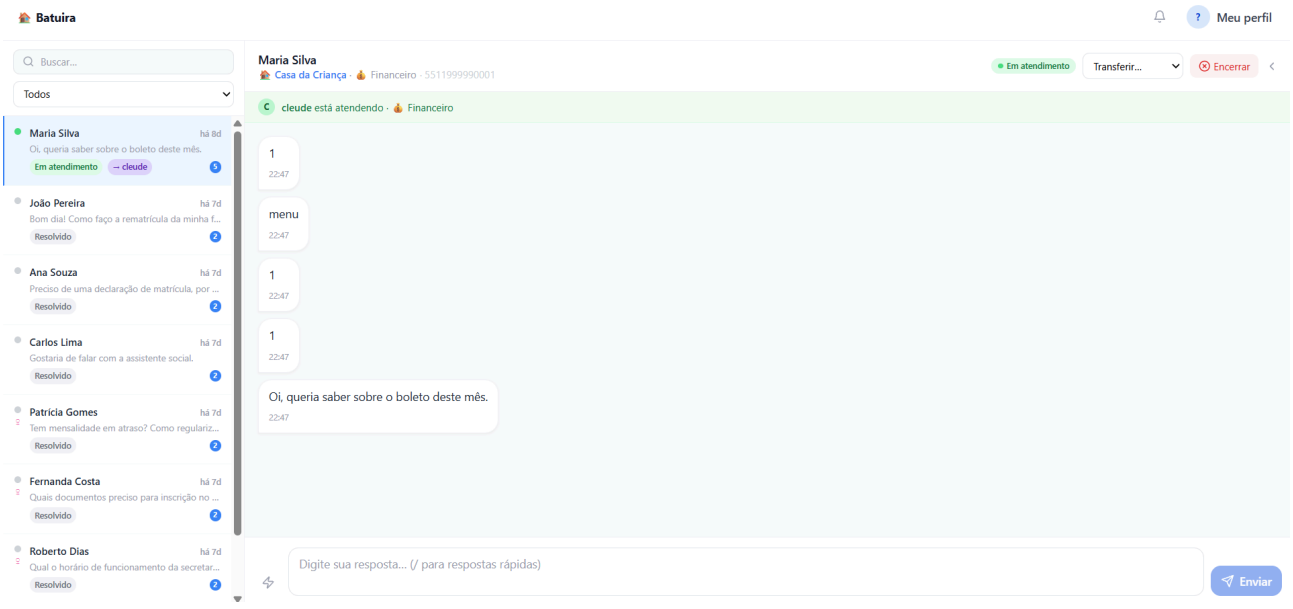
Foram analisadas soluções existentes no mercado de atendimento via WhatsApp, como plataformas de help desk e sistemas de chat, para identificar padrões de UX que melhor atendem ao perfil de organizações sociais. Foram observados aspectos como organização da fila de atendimento, visualização de conversas, facilidade de transferência entre setores e indicação clara de status.

5.2 Protótipos e Telas

A seguir são apresentados protótipos e capturas de tela do sistema desenvolvido. O design utiliza Tailwind CSS com uma paleta limpa e funcional, priorizando a legibilidade e a eficiência do atendente.

Imagem 7: Protótipo da tela de login.

Imagem 8: Painel de atendimento com chat aberto.



6. Deploy e Infraestrutura

O projeto contempla múltiplas opções de deploy para atender diferentes cenários:

- **Fly.io (Backend)** — Configuração em fly.toml para deploy do backend FastAPI com SQLite persistido em volume
- **Vercel (Frontend)** — Deploy automático do frontend Next.js via integração com GitHub
- **Docker Compose (VM)** — docker-compose.yml com Nginx como proxy reverso, scripts setup-vm.sh e deploy.sh para automação
- **Windows local** — Scripts iniciar.bat e parar.bat para execução local simplificada

O histórico de commits do repositório documenta a evolução do projeto:

Hash	Data	Descrição
617f567	18/05/2026	Batuira Bot — commit inicial limpo
8138d62	18/05/2026	Prepara projeto para deploy: Fly.io + Vercel
b4eea0c	22/05/2026	Adiciona engine de fluxo, melhorias no painel e scripts de deploy
5c30142	22/05/2026	Remove dependência de IA desnecessária do bot Batuira

7. Decisões Técnicas

A seguir, as principais decisões de arquitetura tomadas durante o projeto e suas justificativas:

SQLite em vez de PostgreSQL

A escala atual da organização não justifica um servidor de banco separado. SQLite é um arquivo local, zero configuração, backup simples. Se o volume crescer, a migração para PostgreSQL é direta, pois o código usa SQL padrão.

Supabase apenas para autenticação

Gerenciar senhas, tokens JWT, confirmação de e-mail e reset de senha é complexo e arriscado. O Supabase resolve gratuitamente. O backend verifica apenas o token JWT via JWKS, sem depender do banco Supabase.

ES256 (assimétrico) em vez de HS256

O formato atual do Supabase usa ES256. A verificação é feita buscando a chave pública no endpoint JWKS, significando que o backend nunca precisa armazenar o segredo privado.

Polling no frontend (3-4s) em vez de WebSocket

WebSocket exige infraestrutura adicional (Redis pub/sub). Para o volume de atendimentos, polling a cada 3-4 segundos é imperceptível e muito mais simples de manter.

Settings em tempo real via JSON

O arquivo settings_override.json permite alterar configurações (nome da org, bot ativo/inativo, horário de atendimento, chaves de API) sem reiniciar o servidor. As alterações sobrescrevem as variáveis do .env.

API Keys com prefixo btr_ e hash SHA-256

O prefixo facilita auditoria e identificação. SHA-256 é suficiente para chaves geradas aleatoriamente. A chave completa só é exibida uma vez na criação.

8. Orçamento

O projeto foi desenvolvido utilizando exclusivamente tecnologias gratuitas e de código aberto, minimizando custos operacionais. Os custos estimados para manutenção em produção são:

Item	Custo Estimado	Observação
Evolution API (self-hosted)	Gratuito	Hospedada na mesma VM ou em Docker
Supabase Auth (plano gratuito)	Gratuito	Até 50.000 usuários ativos/mês
Hospedagem Backend (Fly.io)	~R$ 0 a R$ 25/mês	Plano gratuito ou Hobby
Hospedagem Frontend (Vercel)	Gratuito	Plano Hobby para projetos pessoais
VM dedicada (alternativa)	R$ 50 a R$ 150/mês	Cloud VPS com Docker Compose
Domínio (opcional)	~R$ 40/ano	Para painel web personalizado
Desenvolvimento	[valor a definir]	Horas de desenvolvimento da equipe

O investimento total para manter o sistema em produção pode ser praticamente zero utilizando os planos gratuitos das plataformas, ou entre R$ 50 e R$ 200/mês com infraestrutura dedicada para maior confiabilidade.

9. Considerações Finais

A proposta de desenvolvimento de um bot de atendimento via WhatsApp para a Casa da Criança Batuira trouxe um desafio e uma responsabilidade significativos. Foi pensado desde o princípio em entregar uma solução que fosse razoável, bem desenhada e que atendesse todas as exigências e necessidades da instituição.

O sistema entregue contempla: bot WhatsApp funcional com menu de instituição e setores; painel web completo com login, fila de atendimento e chat; CRUD de setores e atendentes; engine de fluxo visual para configuração sem código; sistema de API Keys para integrações externas; configurações dinâmicas em tempo real; scripts de deploy para múltiplas plataformas; pesquisa de satisfação pós-atendimento; e suporte a duas instituições.

O projeto demonstra viabilidade técnica e econômica para organizações da sociedade civil que buscam profissionalizar seu atendimento digital. A arquitetura modular e as decisões técnicas documentadas garantem que o sistema possa evoluir conforme as necessidades da instituição, com possibilidade de migração para tecnologias mais robustas caso o volume de atendimentos cresça.

Por fim, conclui-se o trabalho com a entrega de todos os componentes levantados durante as etapas de produção, de forma satisfatória aos requisitos apresentados, chegando a um sistema funcional e pronto para alavancar o atendimento da Casa da Criança Batuira e da Casa da Mãe Batuira.

Referências

EVOLUTION API. Documentação oficial da Evolution API. Disponível em <https://doc.evolution-api.com>. Acesso em maio de 2026.

FASTAPI. FastAPI framework, high performance, easy to learn, fast to code. Disponível em <https://fastapi.tiangolo.com>. Acesso em maio de 2026.

IPEA - Instituto de Pesquisa Econômica Aplicada. Mapa das Organizações da Sociedade Civil. Disponível em <https://mapaosc.ipea.gov.br/indicadores>. Acesso em maio de 2026.

MINISTÉRIO DA CASA CIVIL. 90% dos lares brasileiros já têm acesso à internet no Brasil, aponta pesquisa. Disponível em <https://www.gov.br/casacivil>. Setembro de 2022.

NEXT.JS. The React Framework for the Web. Disponível em <https://nextjs.org>. Acesso em maio de 2026.

SOARES, Michel dos Santos. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Revista Eletrônica de Sistemas de Informação, v. 3, n. 1, junho 2004.

SQLITE. SQLite Home Page. Disponível em <https://www.sqlite.org>. Acesso em maio de 2026.

SUPABASE. The Open Source Firebase Alternative. Disponível em <https://supabase.com>. Acesso em maio de 2026.

TAILWIND CSS. A utility-first CSS framework. Disponível em <https://tailwindcss.com>. Acesso em maio de 2026.