SWT3 - Introdução à IA no Desenvolvimento de Software - Projeto Final

1. CONTEXTUALIZAÇÃO

O manual do proprietário representa um artefato crucial no contexto veicular, servindo como a fonte primária e mais detalhada de informações técnicas e operacionais, ratificadas pela montadora. Sua função é detalhar a composição, funcionalidades e procedimentos essenciais para a correta utilização e manutenção do veículo, visando maximizar sua vida útil e garantir a segurança do condutor e passageiros.

Entretanto, observa-se que uma parcela significativa dos usuários negligencia a leitura integral desses documentos. Essa negligência é frequentemente atribuída à sua extensão considerável e à utilização de uma linguagem excessivamente técnica, o que resulta em dificuldades na compreensão e, consequentemente, pode levar a incidentes operacionais e danos prematuros ao veículo.

Partindo desta problemática, o presente projeto propõe o desenvolvimento de uma ferramenta de interação que visa otimizar a experiência de consumo das informações contidas nos manuais do proprietário. O cerne da solução reside na aplicação da metodologia **RAG (*Retrieval-Augmented Generation* ou Geração Aumentada por Recuperação)**. Esta abordagem permitirá a extração assertiva e eficiente de informações específicas dos documentos, respondendo às consultas dos usuários de maneira dinâmica e contextualizada, transformando o manual em uma base de conhecimento acessível via *chat*.

2. METODOLOGIA

Inicialmente, algumas possíveis topologias para a aplicação foram consideram, sendo elas:

- Lovable + OpenAI Agent
- Lovable Standalone
- Telegram + Open AI Agent

2.1 - Open AI Agent

Para o desenvolvimento do agente inteligente integrado, utilizou-se o OpenAI AgentKit, um framework integrado de ferramentas para o ciclo de vida completo de aplicações. A escolha desta plataforma fundamentou-se no curto prazo disponível e sua rápida curva de aprendizado. Assim, foi necessário apenas submeter o manual do proprietário da Fiat Toro 2022/2023, desenvolver um prompt otimizado e realizar a conexão com a ferramenta de backend para que a sua utilização se tornasse viável.

2.2 - Supabase

Trata-se de uma plataforma *Backend as a Service* (BaaS) de código aberto, frequentemente apresentada como uma alternativa ao Firebase. Sua arquitetura é baseada em PostgreSQL e oferece um conjunto robusto de funcionalidades integradas, essenciais para o desenvolvimento ágil e escalável. Estas funcionalidades incluem: **banco de dados relacional, autenticação de usuários, APIs em tempo real, armazenamento de arquivos, funções serverless.**

A escolha desta ferramenta também se justifica pela sua compatibilidade e fácil integração com a plataforma Lovable.

2.3 - Lovable

Consiste em uma plataforma de desenvolvimento de software baseada em inteligência artificial (IA) que permite criar aplicações web funcionais, sites e ferramentas internas a partir de descrições textuais (prompts), sem a necessidade de escrever código manualmente.

2.4 - Implementação

A etapa de implementação foi seccionada de acordo com o conteúdo adquirido durante a disciplina de Introdução à IA, dessa maneira, o projeto foi norteado pelo conceito de "*user stories".*

2.4.1. Documentação e Repositório (Documentar Prompt e Processo)

Documentação

2.4.2. Post Mortem

Post Mortem

2.4.4. Links adicionais do projeto

- **Github:**

  https://github.com/Gabrielhxavier/smart-car-manual

- **Supabase Project URL:**

  https://iijaqloachmqgkqwpphn.supabase.co

- **JWT (Anônimo):**

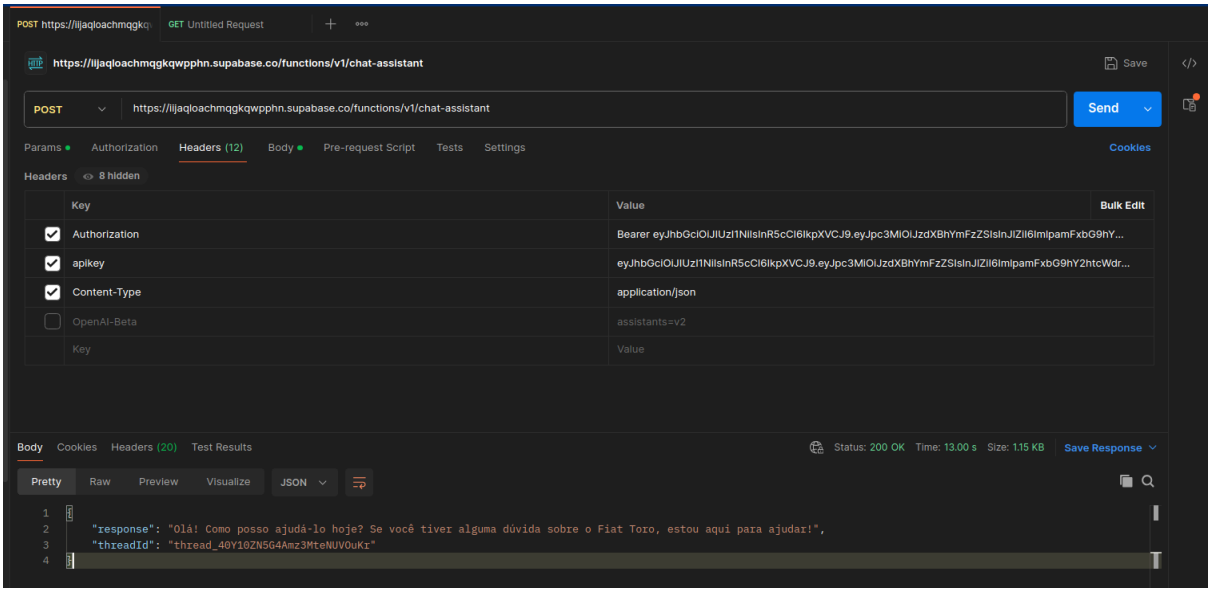
  ```
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsI
  nJlZiI6ImlpamFxbG9hY2htcWdrcXdwcGhuIiwicm9sZSI6ImFub24iLCJpYXQ
  iOjE3NzAwMzY5OTgsImV4cCI6MjA4NTYxMjk5OH0.qyB4V0MO8orfVa-3gwrBC
  ```

madodU-g-Zilz_zFj84UnE

- **Comando para API Key (Exemplo):**

npx supabase secrets set OPENAI_API_KEY=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

Postman



URL: https://iijaqloachmqgkqwpphn.supabase.co/functions/v1/chat-assistant

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImlpamFxbG9hY2htcWdrcXdwcGhuIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NzAwMzY5OTgsImV4cCI6MjA4NTYxMjk5OH0.qyB4V0MO8orfVa-3gwrBCmadodU-g-Zilz_zFj84UnE

apikey: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImlpamFxbG9hY2htcWdrcXdwcGhuIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NzAwMzY5OTgsImV4cCI6MjA4NTYxMjk5OH0.qyB4V0MO8orfVa-3gwrBCmadodU-g-Zilz_zFj84UnE

Content-Type: application/json

Body: {

"message": "Olá, teste!"

```
   }
```

CodePen

```
// Follow this setup guide to integrate the Deno language server with your editor:

// https://deno.land/manual/getting_started/setup_your_environment

// This enables autocomplete, go to definition, etc.


// Setup type definitions for built-in Supabase Runtime APIs

import { serve } from "https://deno.land/std@0.168.0/http/server.ts";

import OpenAI from "npm:openai@latest";


const corsHeaders = {

  "Access-Control-Allow-Origin": "*",

        "Access-Control-Allow-Headers":    "authorization,    x-client-info,    apikey,
content-type",

  "Content-Type": "application/json",

};


// ✅ Inicializa cliente OpenAI uma vez (reutilizado entre requests)

const apiKey = Deno.env.get("OPENAI_API_KEY");

const assistantId = Deno.env.get("OPENAI_ASSISTANT_ID");


if (!apiKey || !assistantId) {

  console.error("Missing OPENAI_API_KEY or OPENAI_ASSISTANT_ID");

}


const openai = apiKey ? new OpenAI({

  apiKey,

  timeout: 25000, // ✅ Timeout de 25s (edge functions têm limite de 30s)

  defaultHeaders: {

    "OpenAI-Beta": "assistants=v2",
```

```
  },
}) : null;

// ✅ Helper otimizado para JSON response
function json(data: unknown, status = 200) {
  return new Response(JSON.stringify(data), {
    status,
    headers: corsHeaders,
  });
}

serve(async (req) => {
  // ✅ CORS preflight - resposta imediata
  if (req.method === "OPTIONS") {
    return new Response("ok", { headers: corsHeaders });
  }

  try {
    // ✅ Valida env vars logo no início
    if (!openai || !assistantId) {
      return json({ error: "Missing environment variables" }, 500);
    }

    // ✅ Parse URL e body em paralelo (não bloqueia)
    const url = new URL(req.url);
    const action = url.searchParams.get("action") ?? "sync";

    // ✅ Parse JSON com error handling inline
    const body = await req.json().catch(() => ({}));
```

```
const message: string | undefined = body?.message;
let activeThreadId: string | undefined = body?.threadId ?? body?.thread_id;
const runId: string | undefined = body?.runId ?? body?.run_id;

// -------------------------
// ACTION: STATUS
// -------------------------
if (action === "status") {
  if (!activeThreadId || !runId) {
    return json({ error: 'threadId and runId required' }, 400);
  }

  // ✅ Busca run e messages em paralelo se completed
  const run = await openai.beta.threads.runs.retrieve(activeThreadId, runId);

  if (run.status !== "completed") {
    return json({
      status: run.status,
      threadId: activeThreadId,
      runId,
    });
  }

  // ✅ Busca apenas últimas mensagens (limit reduzido)
  const messages = await openai.beta.threads.messages.list(activeThreadId, {
    limit: 5, // ✅ Reduzido de 20 para 5 (mais rápido)
    order: "desc"
  });
```

```
  const lastAssistant = messages.data.find((m) => m.role === "assistant");
  const response = lastAssistant?.content?.[0]?.type === "text"
    ? lastAssistant.content[0].text.value
    : "";

  return json({
    status: "completed",
    response,
    threadId: activeThreadId,
    runId,
  });
}

// ✅ Valida message para start/sync
if (!message) {
  return json({ error: "Message is required" }, 400);
}

// -------------------------
// Create or reuse thread
// -------------------------
if (!activeThreadId) {
  const thread = await openai.beta.threads.create();
  activeThreadId = thread.id;
}

// ✅ Adiciona mensagem e cria run em sequência otimizada
await openai.beta.threads.messages.create(activeThreadId, {
```

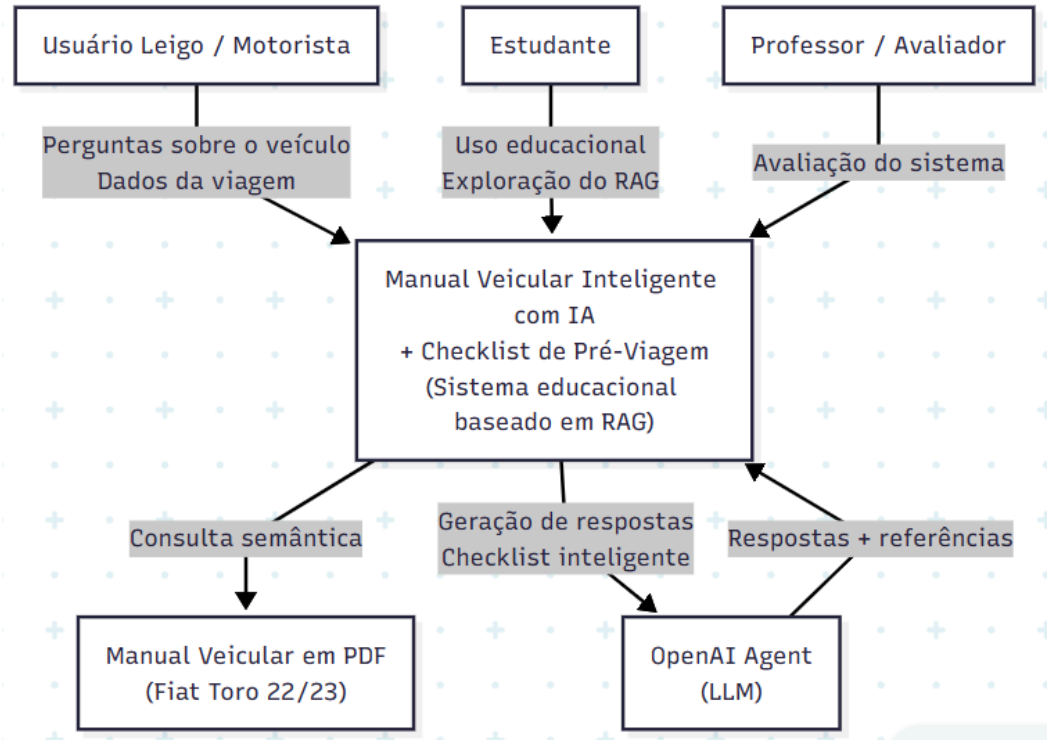
```
  role: "user",
  content: message,
});

// -------------------------
// ACTION: START (async)
// -------------------------
if (action === "start") {
  const run = await openai.beta.threads.runs.create(activeThreadId, {
    assistant_id: assistantId,
  });

  return json({
    status: run.status,
    threadId: activeThreadId,
    runId: run.id,
  });
}

// -------------------------
// ACTION: SYNC (polling otimizado)
// -------------------------
const run = await openai.beta.threads.runs.create(activeThreadId, {
  assistant_id: assistantId,
});

// ✅ Polling com timeout e intervalo otimizado
const maxAttempts = 20; // ✅ Reduzido de infinito
const pollInterval = 1000; // ✅ 1s (era implícito antes)
```

```
    let attempts = 0;

    while (attempts < maxAttempts) {
          const currentRun = await openai.beta.threads.runs.retrieve(activeThreadId,
run.id);

      if (currentRun.status === "completed") {
        const messages = await openai.beta.threads.messages.list(activeThreadId, {
          limit: 5,
          order: "desc"
        });

        const lastAssistant = messages.data.find((m) => m.role === "assistant");
        const response = lastAssistant?.content?.[0]?.type === "text"
          ? lastAssistant.content[0].text.value
          : "";

        return json({
          status: "completed",
          response,
          threadId: activeThreadId,
          runId: run.id,
        });
      }

      if (["failed", "cancelled", "expired"].includes(currentRun.status)) {
        return json({
          error: `Run ${currentRun.status}`,
          status: currentRun.status,
```

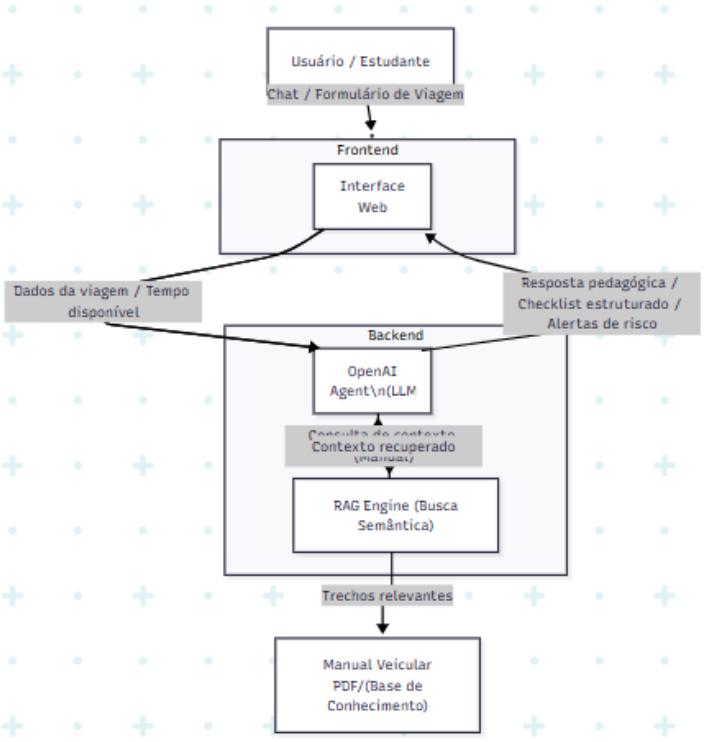
```
          threadId: activeThreadId,
          runId: run.id,
        }, 500);
      }

      // ✅ Aguarda antes de próxima tentativa
      await new Promise((resolve) => setTimeout(resolve, pollInterval));
      attempts++;
    }

    // ✅ Timeout no polling
    return json({
      error: "Polling timeout",
      status: "timeout",
      threadId: activeThreadId,
      runId: run.id,
    }, 408);

  } catch (error) {
    console.error("Error:", error);
    return json({
      error: error instanceof Error ? error.message : "Internal error"
    }, 500);
  }
});
```

2.5 - Diagramas de arquitetura

2.5.1. Diagrama de contexto



2.5.2. Diagrama de containers



2.5.2. Diagrama de componentes