BakaSub Design Document

BakaSub é uma ferramenta de engenharia de tradução e muxagem de legendas via terminal (TUI). O projeto transcende o conceito de um simples script, rejeitando a estética moderna "web" em favor de uma identidade visual **"High-Fidelity ASCII"** (inspirada na Demoscene, `btop` e interfaces industriais/retro).

O core da aplicação é escrito em **Go (Golang)** para garantir performance nativa, binários estáticos leves e concorrência robusta. Utiliza Inteligência Artificial (Arquitetura Modular) para realizar traduções contextuais de alta precisão, mantendo a integridade técnica de formatos complexos (ASS/SSA), gerenciando custos e garantindo sincronia perfeita.

🎨 UX e Design System ("Native Neon TUI")

A interface é construída sobre a arquitetura **The Elm Architecture** usando as bibliotecas **Bubble Tea** (lógica) e **Lip Gloss** (estilo). Rejeita terminantemente botões estilo web, mouse e CSS.

1. Identidade Visual

- **Geometria Pura:** Uso estrito de caracteres Unicode Box Drawing (`│`, `─`, `╔`, `╗`) para criar painéis conectados.
- **Densidade "Btop":** Layouts compactos, uso de gráficos em Braille e barras de progresso em blocos (`▓`, `░`). Sem padding desnecessário.
- **Navegação Keyboard-Centric:** A interação é 100% via teclado.
    - **Sem Botões Clicáveis:** Ações são indicadas por teclas de atalho entre colchetes (ex: `[ c ] CONFIG`, `[ q ] QUIT`).
    - **Navegação:** Setas para listas, `Tab` para campos, `Esc` para voltar.

- **Zero System Windows:** O sistema nunca abre janelas nativas. Inputs de arquivos e diálogos são Modais ASCII sobrepostos (Overlays).

2. Dashboard (Tela Inicial)

Um painel de controle denso, sem espaços vazios:

- **Header:** Logo ASCII e Status de Dependências/API.
- **Input & Mode:** Seleção de Caminho e alternância entre *Full Process* e *Watch Mode*.
- **Modules (1-4):** Atalhos diretos para Extração, Tradução, Muxagem e **Manual Review**.
- **Toolbox (5-8):** Atalhos para ferramentas MKV (Header Editor, Attachments, Remuxer, Glossary).
- **System & AI:** Resumo do Provider/Modelo atual, Custo estimado da sessão e Temperatura.

3. Visualização de Processos ("The Log View")

Feedback visual técnico durante a execução:

- **Painel de Logs:** Stream de eventos em tempo real.
- **Progresso Físico:** Barras de bloco para progresso do arquivo atual e do lote total.
- **Stats em Tempo Real:** Velocidade (tokens/sec) e contagem de erros tratados.

⚙️ Fluxos de Trabalho (Workflows)

1. Modos de Execução

- **Input Manual:** Digitar/Colar caminho.
- **Detecção de Contexto:** Se for pasta, pergunta: *Process Batch* ou *Select Single File*.
- **Watch Mode (Monitoramento):** Utiliza `fsnotify` para vigiar pastas. Ao detectar um novo arquivo `.mkv` (e aguardar o fim da cópia), inicia o fluxo *Touchless* automaticamente.

2. Full Mode Strategy ("Job Setup")

Tela de preparação "Pre-Flight" crítica antes de gastar tokens:

- **Extração:** Seleção de faixas (Legenda Fonte / Áudio Ref).
- **Resolução de Conflitos (Blocker):** Se houver múltiplas faixas do mesmo idioma (ex: 2x 'eng'), o botão de "Start" fica **BLOQUEADO**. O usuário deve teclar `[ r ] RESOLVE` para abrir um Modal e escolher a faixa correta.

- **Simulação (Dry Run):** Tecla `[ d ]` gera um relatório de "Bill of Materials" (Custo Estimado, Tokens, Permissões de Escrita) sem chamar a API paga.
- **Configuração de Contexto:** Define o Perfil de Prompt (Anime, Filme, etc.).

3. Automação Touchless

Para execução sem supervisão (Watch Mode ou Scripts).

- **Configuração:** Um modal específico define as regras de decisão "cegas":
  - *Conflito:* Auto-selecionar Maior arquivo, Menor arquivo ou Pular.
  - *Contexto Default:* Qual perfil assumir (ex: Sempre Anime).
  - *Muxing:* Substituir ou Novo Arquivo.

4. Tradução (Core Engine)

- **Modular AI Providers:** Arquitetura agnóstica. Suporta:
  - **OpenRouter:** (Recomendado - Claude, GPT-4, etc).
  - **Google Gemini:** (API Nativa).
  - **OpenAI:** (API Nativa).
  - **Local LLM:** (Ollama/LMStudio via URL base).
- **Contextual Pre-flight (Smart NER):** Scan prévio para identificar Nomes Próprios e criar um *Glossário Volátil*.
- **Engenharia de Prompts (Profile System):**
  - **Factory Profiles (🔒):** Prompts otimizados imutáveis (Anime, Movie, Series).
  - **User Profiles (👤):** O usuário clona um Factory para editar. Permite personalização extrema (ex: "Anime - Gírias BR").
- **Tier-Aware Throttling:** Ajuste dinâmico de concorrência baseada no Provider.

5. Gestão de Custos e DBLocal

- **DBLocal 2.0 (Fuzzy Match):** Banco SQLite (`bakasub.db`). Armazena `Hash(Original) -> Tradução`. Usa *Levenshtein Distance* para reutilizar traduções se a frase for >95% similar (ex: correção de pontuação não gasta tokens).
- **Payload Minification:** JSON estritamente minificado para reduzir Input Tokens.

🛡️ Engenharia: Blindagem e Resiliência

1. Blindagem Contra Dessincronia

- **JSON Binding:** Payload com IDs explícitos `[{"i":10, "t":"..."}]`. Reconstrução baseada em ID, nunca em ordem de lista.

- **Sliding Window:** Injeta as últimas 3 linhas do lote anterior como "contexto passivo" para manter a fluidez do diálogo.
- **Self-Healing:** Se a API falhar ou alucinar o JSON, o sistema divide o lote recursivamente (50 -> 25+25) e tenta novamente.

2. Smart Resume (Recuperação de Crash)

- **State File:** Salva um `.bakasub.temp` atômico a cada lote traduzido.
- **Crash Handler:** Se o app fechar abruptamente (queda de energia/erro), ao reabrir, ele detecta o arquivo temporário e exibe um **Modal de Retomada**: *"Sessão anterior detectada. Continuar do lote 14?"*.

3. Quality Gate (Linter)

Antes de muxar, roda uma verificação rápida:

- Tags ASS quebradas (`{\an8` sem fechar).
- Resíduos de texto original.
- Se falhar, oferece: *Auto-Fix*, *Manual Review* ou *Ignore*.

🛠️ Toolbox (Ferramentas)

Ferramentas integradas para evitar o uso de softwares externos:

1. **Manual Review Editor:** Um editor TUI (side-by-side) para revisar/corrigir legendas manualmente antes da muxagem final.
2. **Project Glossary:** Editor de tabela para definir termos fixos da série (ex: "Nakama" -> "Companheiro") que são injetados no prompt.
3. **Header Editor:** Altera flags (Default/Forced) e Título do MKV in-place (sem remuxar).
4. **Attachment Manager:** Adiciona/Extrai/Remove fontes e capas do MKV.
5. **Quick Remuxer:** Limpeza de faixas (remover dublagens/legendas indesejadas) gerando um arquivo novo.

⚙️ Menu de Configurações

- **AI PROVIDERS:** Seleção do serviço (OpenRouter, Gemini, Local) e input de chaves/URL.
- **GENERAL:**
    - *Idioma de Destino:* Lista com Top 5 + Input para Código ISO Customizado.

    - *Touchless Rules:* Botão para configurar regras de automação.
    - *HI Tags:* Checkbox para remover `[Sons]`, `(Música)`.
- **AI MODELS:** Busca e seleção de modelos (buscando da API do provider).
- **PROMPTS:** Gerenciador de Perfis (Factory vs User).
- **ADVANCED:** Limpeza de Cache Semântico e Arquivos Temporários. Logs de Debug.

🚀 Fluxo de Boas-Vindas (Onboarding)

Wizard executado na primeira abertura (sem `config.json`):

1. **Acesso:** Escolha do Provider e Input da API Key (ou URL Local).
2. **Dependências:** Checagem e download automático de binários estáticos (`ffmpeg`, `mkvtoolnix`) se não encontrados no PATH.
3. **Defaults:** Seleção do Modelo de IA inicial e Idioma de Destino.

📦 Distribuição e Ciclo de Vida

1. Estratégia "Binary First"

O foco principal de entrega são **Binários Estáticos Standalone**. O usuário final não deve precisar instalar Go, Python ou Git para usar o BakaSub.

- **Windows:** Executável portátil (`bakasub.exe`).
- **Linux:** Binário ELF estático (`bakasub`).
- **CI/CD:** GitHub Actions compila e publica releases automaticamente para ambas as plataformas.

2. Contribuidores & Power Users

O método de clonar o repositório (`git clone`) e compilar localmente (`go build`) é reservado estritamente para desenvolvimento, contribuição ou auditoria de código.

3. Documentação e Localização

A documentação deve ser acessível e traduzida, refletindo a natureza global da ferramenta. Os arquivos obrigatórios no repositório são:

- `README.md`: Inglês (Fonte da verdade).

- `README-pt.md`: Português Brasileiro.
- `README-es.md`: Espanhol.

4. Atualizações (Update Checker)

- O binário consulta a API do GitHub Releases (`GET /releases/latest`) de forma assíncrona ao iniciar.
- Se uma nova versão for detectada, exibe um alerta visual discreto e não obstrutivo na Dashboard (`[!] UPDATE AVAILABLE`).