📋 Controle de Versão - Licenciamento Ambiental Frontend

> Estratégias e práticas para versionamento da aplicação

🎯 Estratégia Adotada

Abordagem Híbrida: package.json + Arquivo de Configuração ☑ IMPLEMENTADO

Combinamos o versionamento semântico do NPM com um arquivo de configuração centralizado para metadados.

Implementação Atual:

☑ 1. Versão no package.json

```
{
  "name": "licenciamento-ambiental-frontend",
  "version": "1.0.0",
  "description": "Sistema de Licenciamento Ambiental - Integração"
}
```

☑ 2. Arquivo de Configuração Arquivo: src/config/version.ts

```
import packageJson from '../../package.json';

export const APP_VERSION = `v${packageJson.version}`;
export const APP_NAME = 'Licenciamento Ambiental - Integração';
export const BUILD_DATE = new Date().toISOString().split('T')[0];

export const VERSION_INFO = {
  version: APP_VERSION,
  name: APP_NAME,
  buildDate: BUILD_DATE,
  fullName: `${APP_NAME} ${APP_VERSION}`
};

export function logVersionInfo() {
  console.log(`
╔═══════════════════════════════════════════════════════╗
║  ${APP_NAME}
║  Versão: ${APP_VERSION}
║  Build: ${BUILD_DATE}
╚═══════════════════════════════════════════════════════╝
  `);
}
```

☑ 3. Uso no Dashboard

```
import { APP_VERSION, APP_NAME } from '../config/version';

// No sidebar:
<div className="flex items-center justify-between w-full">
  <p className="text-xs text-gray-500">{APP_NAME}</p>
  <span className="px-2 py-0.5 text-xs font-medium bg-green-100 text-green-700
rounded">
    {APP_VERSION}
  </span>
</div>
```

☑ 4. Log no Console

```
// src/main.tsx
import { logVersionInfo } from './config/version';

logVersionInfo(); // Exibe banner no console ao iniciar
```

🗀 Estrutura de Versionamento

1. Version no package.json

```
{
  "name": "licenciamento-ambiental-frontend",
  "version": "1.0.0",
  "description": "Sistema de Licenciamento Ambiental - Integração"
}
```

2. Arquivo de Configuração (Proposto)

Arquivo: src/config/version.ts

```
import packageJson from '../../package.json';

export const APP_VERSION = `v${packageJson.version}`;
export const APP_NAME = 'Licenciamento Ambiental - Integração';
export const BUILD_DATE = new Date().toISOString().split('T')[0];

export const VERSION_INFO = {
  version: APP_VERSION,
  name: APP_NAME,
```

```
  buildDate: BUILD_DATE
};
```

3. Uso no Dashboard

```
import { APP_VERSION, APP_NAME } from '../config/version';

// No sidebar:
<div className="flex items-center justify-between">
  <p className="text-xs text-gray-500">{APP_NAME}</p>
  <span className="px-2 py-0.5 text-xs font-medium bg-green-100 text-green-700
rounded">
    {APP_VERSION}
  </span>
</div>
```

🔢 Versionamento Semântico

Seguimos o padrão **SemVer** (Semantic Versioning):

```
MAJOR.MINOR.PATCH

Exemplo: v1.2.3
         | | |
         | | └─ PATCH: Correções de bugs (bugfixes)
         | └─── MINOR: Novas funcionalidades (features)
         └───── MAJOR: Mudanças incompatíveis (breaking changes)
```

Regras de Incremento:

Tipo de Mudança	Versão	Exemplo
🐛 **Bugfix** (correção de erro)	PATCH	1.0.0 → 1.0.1
✨ **Feature** (nova funcionalidade)	MINOR	1.0.1 → 1.1.0
💥 **Breaking Change** (incompatível)	MAJOR	1.1.0 → 2.0.0
📝 **Docs** (documentação)	Não altera	1.0.0 → 1.0.0
🎨 **Style** (formatação)	Não altera	1.0.0 → 1.0.0

📌 Histórico de Versões

v1.0.0 - 2025-11-04 (Release Inicial)

Features Implementadas:

🏠 Busca de Imóvel

- Modal em 2 etapas (busca → confirmação)
- Debounce de 500ms
- API: `GET /imoveis/buscar?q={query}`
- Fallback para Supabase
- Exibição: Nome do Imóvel, Área Total, Endereço
- Tags: `#search` `#modal` `#property`

📋 Fluxo de Solicitação (6 Steps)

1. **Participantes** - Seleção de PF/PJ (Requerente, Procurador, Técnico)
2. **Imóvel** - Busca e seleção de imóvel cadastrado
3. **Atividade** - Seleção de atividade do empreendimento
4. **Formulário** - FormWizard com 7 sub-etapas
5. **Documentação** - Upload de documentos (UI pronta)
6. **Revisão** - Revisão final dos dados

🔄 Renomeações

- "Inscrição" → "Solicitação"
- "Empreendimento" → "Atividade"

🐛 Correções

- **Crítico:** Refatoração de URLs da API (timeout resolvido)
- Migração fetch → axios
- Cliente HTTP centralizado (`lib/api/http.ts`)
- Correção de tipos (processId: number → string UUID)
- **Fix:** Correção de endpoints (adição de `/` antes dos recursos)
  - ☑ `usoAguaService.ts` - `/consumo-de-agua`
  - ☑ `residuosService.ts` - `/residuos/*`
  - ☑ `outrasInformacoesService.ts` - `/processos/*/outras-informacoes`
  - ☑ `outorgasService.ts` - `/outorgas`

🎨 UI/UX

- Colunas adicionais na tabela de imóveis: Nome e Área Total
- Mensagem "Endereço não cadastrado" quando ausente
- Label atualizado: "Nome" → "Nome do Imóvel"
- Badge de versão no menu lateral

🚀 Como Atualizar a Versão

Passo 1: Decidir o tipo de mudança

- Bugfix? → Incrementa PATCH
- Feature? → Incrementa MINOR

- Breaking? → Incrementa MAJOR

Passo 2: Atualizar package.json

```
# Manualmente ou usando NPM:
npm version patch   # 1.0.0 → 1.0.1
npm version minor   # 1.0.0 → 1.1.0
npm version major   # 1.0.0 → 2.0.0
```

Passo 3: Atualizar CHANGELOG

Documentar no `CHANGELOG.md` (ou neste arquivo) as mudanças.

Passo 4: Commit e Tag

```
git add .
git commit -m "chore: bump version to v1.1.0"
git tag v1.1.0
git push origin main --tags
```

🔮 Roadmap de Versões Futuras

v1.1.0 (Planejado)

- ☐ Backend de upload de documentos (DocumentacaoPage)
- ☐ Refinamento do fluxo de processos
- ☐ Dashboard de processos ativos

v1.2.0 (Planejado)

- ☐ Notificações em tempo real
- ☐ Histórico de alterações por processo
- ☐ Relatórios em PDF

v2.0.0 (Futuro)

- ☐ Redesign completo da UI
- ☐ Novo sistema de autenticação
- ☐ API GraphQL

📊 Onde a Versão Aparece

1. **Menu Lateral (Dashboard)** - Badge com versão atual
2. **Footer da Aplicação** - Texto "v1.0.0"
3. **Sobre o Sistema** - Modal com detalhes completos

4. **Console do Navegador** - Log na inicialização
5. **Meta Tags HTML** - Para rastreamento

🛠️ Implementação Alternativa: Variável de Ambiente

Se preferir não usar `package.json`, pode criar:

`.env`:

```
VITE_APP_VERSION=v1.0.0
VITE_APP_NAME="Licenciamento Ambiental - Integração"
```

Uso:

```
const version = import.meta.env.VITE_APP_VERSION;
```

📝 Convenções de Commit (Opcional)

Para facilitar o versionamento automático:

```
feat: nova funcionalidade (MINOR)
fix: correção de bug (PATCH)
docs: documentação (nenhum)
style: formatação (nenhum)
refactor: refatoração (nenhum, ou PATCH se corrige bug)
perf: melhoria de performance (PATCH ou MINOR)
test: testes (nenhum)
chore: tarefas de build/CI (nenhum)

BREAKING CHANGE: mudança incompatível (MAJOR)
```

Exemplo:

```
git commit -m "feat: adicionar busca de imóvel com debounce"
git commit -m "fix: corrigir URLs da API com barra inicial"
git commit -m "feat!: novo sistema de autenticação

BREAKING CHANGE: AuthContext agora usa JWT em vez de session"
```

🎯 Estratégias de Deploy

Branch-based Versioning

- `main` → Produção (v1.0.0, v1.1.0, etc.)
- `develop` → Desenvolvimento (v1.1.0-beta.1)
- `feature/*` → Features específicas (v1.1.0-alpha)

Tag-based Versioning

```
git tag v1.0.0        # Release estável
git tag v1.1.0-rc.1   # Release Candidate
git tag v1.1.0-beta.2 # Beta
```

Última atualização: 04/11/2025
Versão atual: v1.0.0
Mantido por: Equipe de Desenvolvimento