🚀 GUIA COMPLETO DE SETUP - CP2B MAPS V3 (NEWLOOK)

Data de Início: Segunda-feira, 16 de Novembro de 2025

📋 ÍNDICE DO SETUP

Tempo Total Estimado: ~3.5 horas

---

1. PREPARAÇÃO DO WORKSPACE (30min)

PASSO 1.1: Criar Estrutura de Diretórios

Abrir terminal e executar:

```
# 1. Criar workspace principal
mkdir -p ~/cp2b-workspace
cd ~/cp2b-workspace

# 2. Clonar repositórios
echo "📥 Clonando project_map (referência)..."
git clone https://github.com/aikiesan/project_map.git

echo "📥 Clonando NewLook (desenvolvimento)..."
git clone https://github.com/aikiesan/NewLook.git

# 3. Criar pastas auxiliares
mkdir -p docs-planning
mkdir -p backups

# 4. Verificar estrutura criada
```

```
tree -L 1 .
# Deve mostrar:
# .
# ├── project_map/
# ├── NewLook/
# ├── docs-planning/
# └── backups/

echo "✅ Workspace preparado em ~/cp2b-workspace"
```

PASSO 1.2: Configurar project_map como READ-ONLY

```
cd ~/cp2b-workspace/project_map

# Criar branch de referência (não modificar)
git checkout -b reference-v2
git config core.fileMode false

# Documentar estado atual
echo "# CP2B Maps V2 - Referência" > README_REFERENCE.md
echo "Este repositório é APENAS REFERÊNCIA." >> README_REFERENCE.md
echo "Não fazer commits aqui." >> README_REFERENCE.md
echo "Desenvolvimento ativo em: ../NewLook" >> README_REFERENCE.md
echo "Data snapshot: $(date)" >> README_REFERENCE.md

cd ..
echo "✅ project_map configurado como READ-ONLY"
```

PASSO 1.3: Configurar NewLook para Desenvolvimento

```
cd ~/cp2b-workspace/NewLook

# Configurar Git
git config user.name "Seu Nome Completo"
git config user.email "seu.email@exemplo.com"

# Criar branch de desenvolvimento
git checkout -b develop
git push -u origin develop

# Criar .gitignore inicial
```

```
cat > .gitignore << 'EOF'
# Python
__pycache__/
*.py[cod]
*$py.class
*.so
.Python
env/
venv/
ENV/
.venv

# Streamlit
.streamlit/secrets.toml

# IDE
.vscode/
.idea/
*.swp
*.swo
.cursor/
.DS_Store

# Data (grandes arquivos)
data/raw/*
data/rasters/*.tif
*.sqlite
*.db

# Exceções (manter alguns dados)
!data/raw/.gitkeep
!data/database/.gitkeep

# Environment
.env
.env.local

# Logs
logs/
*.log

# Temporary
tmp/
temp/
```

```
*.tmp

# OS
Thumbs.db
.DS_Store

# Backups
*.bak
*~
EOF

git add .gitignore
git commit -m "chore: Add comprehensive .gitignore"
git push origin develop

echo "✅ NewLook configurado para desenvolvimento"
```

---

2. ESTRUTURA DE PASTAS NEWLOOK (15min)

PASSO 2.1: Criar Estrutura Completa

```
cd ~/cp2b-workspace/NewLook

# Criar toda estrutura de uma vez
mkdir -p
{config,data/{database,shapefile,rasters,raw,processed},docs,logs,scripts,tests}
mkdir -p
src/{auth,ui/{pages,components,styles},data/{loaders,processors},core,ai,utils}

# Criar arquivos .gitkeep para manter pastas vazias no Git
find . -type d -empty -exec touch {}/.gitkeep \;

# Criar estrutura visual
cat > STRUCTURE.md << 'EOF'
# CP2B Maps V3 - Estrutura do Projeto



NewLook/ ├── app.py # 🚀 Entry point Streamlit ├── requirements.txt # 📦
Dependências Python ├── .env.example # 🔧 Template variáveis ambiente ├──
README.md # 📖 Documentação principal ├── CHANGELOG.md # 📝 Histórico
de versões │ ├── config/ # ⚙️ Configurações │ ├── settings.py # Configurações
```

gerais │ ├── database.py # Config banco de dados │ └── supabase.py # Config Supabase │ ├── src/ # 💻 Código fonte │ ├── auth/ # 🔐 Autenticação │ │ ├── **init**.py │ │ ├── supabase_auth.py # Integração Supabase │ │ ├── session_manager.py # Gestão de sessões │ │ └── permissions.py # Sistema de permissões │ │ │ ├── ui/ # 🎨 Interface usuário │ │ ├── pages/ # Páginas principais │ │ │ ├── **init**.py │ │ │ ├── home.py # Landing page │ │ │ ├── login.py # Login/Registro │ │ │ ├── dashboard.py # Dashboard principal │ │ │ ├── map_analysis.py # Análise geoespacial │ │ │ ├── data_explorer.py # Explorar dados │ │ │ ├── advanced_analysis.py # Análises avançadas │ │ │ ├── proximity.py # Análise proximidade │ │ │ ├── mcda.py # Análise MCDA │ │ │ ├── assistant.py # Bagacinho IA │ │ │ ├── references.py # Referências científicas │ │ │ ├── profile.py # Perfil usuário │ │ │ └── about.py # Sobre o projeto │ │ │ │ ├── components/ # Componentes reutilizáveis │ │ │ ├── **init**.py │ │ │ ├── navbar.py # Barra de navegação │ │ │ ├── sidebar.py # Barra lateral │ │ │ ├── cards.py # Cards de métricas │ │ │ ├── maps.py # Componentes de mapa │ │ │ ├── charts.py # Gráficos │ │ │ ├── filters.py # Filtros │ │ │ └── modals.py # Modais/Dialogs │ │ │ │ └── styles/ # Estilos e temas │ │ ├── **init**.py │ │ ├── theme.py # Tema CP2B (verde) │ │ ├── custom.css # CSS customizado │ │ └── wcag.py # Estilos acessibilidade │ │ │ ├── data/ # 📊 Camada de dados │ │ ├── loaders/ # Carregadores │ │ │ ├── **init**.py │ │ │ ├── municipal_loader.py # Dados municipais │ │ │ ├── spatial_loader.py # Dados geoespaciais │ │ │ └── research_loader.py # Dados FAPESP │ │ │ │ └── processors/ # Processadores │ │ ├── **init**.py │ │ ├── biogas_calculator.py # Cálculos biogás │ │ └── spatial_processor.py # Processar geometrias │ │ │ ├── core/ # 🧱 Lógica de negócio │ │ ├── **init**.py │ │ ├── mcda.py # Algoritmo MCDA │ │ ├── proximity_analyzer.py # Análise proximidade │ │ └── optimization.py # Otimização (futuro) │ │ │ ├── ai/ # 🤖 Inteligência Artificial │ │ ├── **init**.py │ │ ├── gemini_client.py # Cliente Gemini API │ │ ├── rag_system.py # Sistema RAG │ │ └── rate_limiter.py # Controle de uso │ │ │ └── utils/ # 🛠️ Utilitários │ ├── **init**.py │ ├── logger.py # Sistema de logs │ ├── cache_manager.py # Gestão de cache │ ├── validators.py # Validações │ └── helpers.py # Funções auxiliares │ ├── data/ # 💾 Dados da aplicação │ ├── database/ # Banco SQLite │ │ └── municipios.db │ ├── shapefile/ # Shapefiles │ │ └── municipios_sp.shp │ ├── rasters/ # Rasters MapBiomas │ │ └── mapbiomas_sp.tif │ ├── raw/ # Dados brutos │ │ └── Dados_Por_Municipios_SP.xls │ └── processed/ # Dados processados │ ├── docs/ # 📚 Documentação │ ├── ARCHITECTURE.md # Arquitetura técnica │ ├── METHODOLOGY.md # Metodologia científica │ ├── USER_GUIDE.md # Guia do usuário │ ├── WCAG_COMPLIANCE.md # Acessibilidade │ └── API.md # Documentação API (futuro) │ ├── tests/ # 🧪 Testes │ ├── **init**.py │ ├── test_auth.py │ ├── test_calculations.py │ └── test_ui.py │ ├── scripts/ # 📜 Scripts utilitários │ ├── setup_database.py # Setup inicial DB │ ├── sync_data.py # Sync dados project_map │ └── deploy.sh # Script de deploy │ └── logs/ # 📋 Logs da aplicação └── app.log

```
EOF

echo "✅ Estrutura de pastas criada. Ver STRUCTURE.md"
```

---

3. CONFIGURAÇÃO DO CURSOR + CLAUDE CODE (20min)

PASSO 3.1: Instalar e Configurar Cursor

```
# 1. Baixar Cursor (se ainda não tiver)
# https://cursor.sh/
# Instalar conforme seu sistema operacional

# 2. Abrir NewLook no Cursor
cd ~/cp2b-workspace/NewLook
cursor .

# Ou se preferir abrir manualmente:
# Cursor → File → Open Folder → Selecionar NewLook
```

PASSO 3.2: Configurações Recomendadas do Cursor

No Cursor, criar `.cursor/settings.json`:

```
mkdir -p .cursor
cat > .cursor/settings.json << 'EOF'
{
  "cursor.chat.model": "claude-3.5-sonnet",
  "cursor.autocomplete.enabled": true,
  "cursor.autocomplete.suggestOnTriggerCharacters": true,

  "python.analysis.typeCheckingMode": "basic",
  "python.linting.enabled": true,
  "python.linting.pylintEnabled": true,
  "python.formatting.provider": "black",

  "editor.formatOnSave": true,
  "editor.codeActionsOnSave": {
    "source.organizeImports": true
  },

  "files.exclude": {
```

```
    "**/__pycache__": true,
    "**/*.pyc": true,
    ".venv": false
  }
}
EOF
```

PASSO 3.3: Extensões Recomendadas

Instalar no Cursor (Cmd/Ctrl + Shift + X):

1. **Python** (Microsoft) - Obrigatório
2. **Pylance** (Microsoft) - Type checking
3. **Black Formatter** - Formatação código
4. **GitLens** - Histórico Git visual
5. **Better Comments** - Comentários coloridos
6. **Error Lens** - Erros inline

PASSO 3.4: Configurar Claude Code (Composer)

Atalhos importantes:

- `Cmd+K` (Mac) ou `Ctrl+K` (Windows): Abrir Claude chat inline
- `Cmd+L` (Mac) ou `Ctrl+L` (Windows): Abrir painel lateral Claude
- `Cmd+Shift+L`: Composer (multi-file editing)

Criar arquivo de instruções para Claude:

```
cat > .cursorrules << 'EOF'
# Instruções para Claude Code - CP2B Maps V3

## Contexto do Projeto
- Projeto: Plataforma de análise de potencial de biogás (São Paulo)
- Stack: Python 3.10+, Streamlit, Supabase, PostGIS
- Prazo: MVP em 4 semanas
- Objetivo: Sistema com autenticação + 8 módulos de análise

## Padrões de Código
- Type hints obrigatórios em todas funções
- Docstrings estilo Google
- Imports organizados (stdlib → third-party → local)
- Nomes em inglês (código), comentários em português
```

- Logging estruturado (não usar print)
- Error handling robusto (try/except com logs)

Estrutura de Funções
```python
def nome_funcao(param: tipo) -> tipo_retorno:
    """Breve descrição.

    Args:
        param: Descrição do parâmetro

    Returns:
        Descrição do retorno

    Raises:
        ErroTipo: Quando ocorre X
    """
    try:
        # Implementação
        pass
    except Exception as e:
        logger.error(f"Erro em nome_funcao: {e}")
        raise
```

Princípios

- WCAG 2.1 AA compliance sempre
- Performance: cache com @st.cache_data quando apropriado
- Segurança: nunca hardcode credentials
- Modularidade: funções pequenas, responsabilidade única
- Testabilidade: código deve ser testável

Referências

- Código V2 em: ../project_map/
- Dados municipais: 645 municípios SP
- Metodologia: Papers MCDA (Kaynak & Gümüş 2025)

Quando Criar Arquivos

- Sempre pergunte antes de criar novos módulos
- Siga estrutura em STRUCTURE.md
- Imports relativos dentro de src/ EOF

---

4. CÓPIA DE DADOS DO PROJECT_MAP (20min)

PASSO 4.1: Script de Cópia Inteligente

```bash
cd ~/cp2b-workspace/NewLook

# Criar script de sync
cat > scripts/sync_data.py << 'EOF'
#!/usr/bin/env python3
"""
Script para copiar dados essenciais do project_map para NewLook.
Execução: python scripts/sync_data.py
"""

import shutil
import os
from pathlib import Path
from datetime import datetime

# Paths
PROJECT_MAP = Path("../project_map")
NEWLOOK = Path(".")

# Mapeamento: origem → destino
COPY_MAP = {
    # Database
    "data/database": "data/database",

    # Shapefiles (apenas essenciais)
    "data/shapefile": "data/shapefile",

    # Rasters otimizados
    "data/rasters": "data/rasters",

    # Dados municipais
    "data/Dados_Por_Municipios_SP.xls": "data/raw/Dados_Por_Municipios_SP.xls",
}

def copy_data():
    """Copia dados seletivamente."""
```

```
print("🔄 Iniciando cópia de dados do project_map...\n")

total_size = 0
copied_files = 0

for src_rel, dst_rel in COPY_MAP.items():
    src = PROJECT_MAP / src_rel
    dst = NEWLOOK / dst_rel

    if not src.exists():
        print(f"⚠️  Origem não existe: {src}")
        continue

    # Criar diretório destino
    dst.parent.mkdir(parents=True, exist_ok=True)

    # Copiar
    if src.is_file():
        print(f"📄 Copiando arquivo: {src.name}")
        shutil.copy2(src, dst)
        size = dst.stat().st_size
        total_size += size
        copied_files += 1
        print(f"   ✅ {size / 1024 / 1024:.2f} MB")

    elif src.is_dir():
        print(f"📁 Copiando diretório: {src.name}")
        if dst.exists():
            shutil.rmtree(dst)
        shutil.copytree(src, dst)

        # Calcular tamanho
        dir_size = sum(f.stat().st_size for f in dst.rglob('*') if f.is_file())
        file_count = len(list(dst.rglob('*')))
        total_size += dir_size
        copied_files += file_count
        print(f"   ✅ {dir_size / 1024 / 1024:.2f} MB ({file_count} arquivos)")

print(f"\n✅ Cópia concluída!")
print(f"   Total: {total_size / 1024 / 1024:.2f} MB")
print(f"   Arquivos: {copied_files}")

# Criar arquivo de metadados
metadata = f"""# Sync Metadata
```

```
Data: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}
Origem: {PROJECT_MAP.absolute()}
Tamanho Total: {total_size / 1024 / 1024:.2f} MB
Arquivos: {copied_files}

## Arquivos Copiados:
"""
    for src_rel in COPY_MAP.keys():
        metadata += f"- {src_rel}\n"

    (NEWLOOK / "data" / "SYNC_INFO.md").write_text(metadata)
    print(f"\n📋 Metadados salvos em: data/SYNC_INFO.md")

if __name__ == "__main__":
    copy_data()
EOF

chmod +x scripts/sync_data.py

# Executar cópia
python scripts/sync_data.py

echo "✅ Dados copiados. Verificar data/SYNC_INFO.md"
```

PASSO 4.2: Verificar Integridade dos Dados

```
# Criar script de verificação
cat > scripts/verify_data.py << 'EOF'
#!/usr/bin/env python3
"""Verifica integridade dos dados copiados."""

from pathlib import Path
import sqlite3

DATA_DIR = Path("data")

def verify():
    """Verifica se dados essenciais existem."""

    checks = {
        "Database": DATA_DIR / "database" / "municipios.db",
        "Excel Municipal": DATA_DIR / "raw" / "Dados_Por_Municipios_SP.xls",
```

```
        "Shapefile": DATA_DIR / "shapefile" / "municipios_sp.shp",
        "Raster MapBiomas": DATA_DIR / "rasters",
    }

    print("🔍 Verificando integridade dos dados...\n")

    all_ok = True
    for name, path in checks.items():
        exists = path.exists()
        status = "✅" if exists else "❌"
        print(f"{status} {name}: {path}")

        if exists and name == "Database":
            # Verificar tabelas
            try:
                conn = sqlite3.connect(path)
                cursor = conn.cursor()
                cursor.execute("SELECT COUNT(*) FROM municipios")
                count = cursor.fetchone()[0]
                print(f"   → {count} municípios na database")
                conn.close()
            except Exception as e:
                print(f"   ⚠️  Erro ao ler database: {e}")
                all_ok = False

        if not exists:
            all_ok = False

    print(f"\n{'✅ Todos dados OK!' if all_ok else '❌ Faltam dados essenciais'}")
    return all_ok

if __name__ == "__main__":
    verify()
EOF

chmod +x scripts/verify_data.py
python scripts/verify_data.py
```

---

5. ARQUIVOS BASE ESSENCIAIS (30min)

PASSO 5.1: requirements.txt

```
cat > requirements.txt << 'EOF'
# Core
streamlit==1.31.0
python-dotenv==1.0.0

# Autenticação
supabase==2.3.0
pydantic==2.5.0

# Dados e Análise
pandas==2.1.4
numpy==1.24.3
openpyxl==3.1.2
xlrd==2.0.1

# Geoespacial
geopandas==0.14.1
shapely==2.0.2
fiona==1.9.5
rasterio==1.3.9
folium==0.15.1
pyproj==3.6.1

# Visualização
plotly==5.18.0
matplotlib==3.8.2

# IA e RAG
google-generativeai==0.3.2
sentence-transformers==2.2.2

# Banco de Dados
sqlalchemy==2.0.23
psycopg2-binary==2.9.9

# Utilitários
jenkspy==0.3.2
pillow==10.1.0
psutil==5.9.6
requests==2.31.0

# Desenvolvimento
black==23.12.1
pylint==3.0.3
```

```
pytest==7.4.3
EOF
```

PASSO 5.2: .env.example

```
cat > .env.example << 'EOF'
# Supabase Configuration
SUPABASE_URL=your_supabase_project_url_here
SUPABASE_ANON_KEY=your_supabase_anon_key_here
SUPABASE_SERVICE_ROLE_KEY=your_service_role_key_here

# Google Gemini API
GEMINI_API_KEY=your_gemini_api_key_here

# Application Settings
APP_ENV=development
DEBUG=true
LOG_LEVEL=INFO

# Database (local SQLite)
DATABASE_PATH=data/database/municipios.db

# Session
SECRET_KEY=your_secret_key_for_sessions_here

# Rate Limiting
GEMINI_RATE_LIMIT_PER_DAY=5
EXPORT_RATE_LIMIT_PER_HOUR=10

# URLs
BASE_URL=http://localhost:8501

# Features Flags
ENABLE_ANALYTICS=false
ENABLE_MCDA=true
ENABLE_AI_ASSISTANT=true
EOF
```

PASSO 5.3: config/settings.py

```
cat > config/settings.py << 'EOF'
```

```
"""
Configurações centralizadas da aplicação.
"""

import os
from pathlib import Path
from dotenv import load_dotenv

# Carregar variáveis de ambiente
load_dotenv()

# Paths
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
LOGS_DIR = BASE_DIR / "logs"

# Supabase
SUPABASE_URL = os.getenv("SUPABASE_URL")
SUPABASE_ANON_KEY = os.getenv("SUPABASE_ANON_KEY")
SUPABASE_SERVICE_ROLE_KEY =
os.getenv("SUPABASE_SERVICE_ROLE_KEY")

# Gemini API
GEMINI_API_KEY = os.getenv("GEMINI_API_KEY")
GEMINI_RATE_LIMIT = int(os.getenv("GEMINI_RATE_LIMIT_PER_DAY", 5))

# Application
APP_ENV = os.getenv("APP_ENV", "development")
DEBUG = os.getenv("DEBUG", "false").lower() == "true"
LOG_LEVEL = os.getenv("LOG_LEVEL", "INFO")
SECRET_KEY = os.getenv("SECRET_KEY", "dev-secret-change-in-production")

# Database
DATABASE_PATH = DATA_DIR / "database" / "municipios.db"

# Validações
def validate_config():
    """Valida configurações essenciais."""
    required = {
        "SUPABASE_URL": SUPABASE_URL,
        "SUPABASE_ANON_KEY": SUPABASE_ANON_KEY,
    }

    missing = [k for k, v in required.items() if not v]
```

```
    if missing:
        raise ValueError(f"Missing required env vars: {', '.join(missing)}")

# Tema CP2B
THEME = {
    "primary_color": "#1E5128",
    "secondary_color": "#4E9F3D",
    "background_color": "#FFFFFF",
    "text_color": "#191A19",
    "accent_color": "#D8E9A8",
}
EOF
```

PASSO 5.4: README.md

```
cat > README.md << 'EOF'
# 🌱 CP2B Maps V3 - Plataforma de Análise de Potencial de Biogás

**Status:** 🚧 Em Desenvolvimento (MVP - 4 semanas)
**Versão:** 3.0.0-alpha
**Início:** 16 de Novembro de 2025

---

## 🎯 Sobre o Projeto

Plataforma web para identificação e análise de localizações ótimas para plantas de biogás no Estado de São Paulo, utilizando:

- 📊 **Análise Multicritério Espacial (MCDA)**
- 🗺️ **Dados Georreferenciados** (MapBiomas, IBGE, FAPESP)
- 🤖 **IA Generativa** (Assistente Bagacinho com RAG)
- ♿ **Acessibilidade** (WCAG 2.1 Level AA)

---

## 🚀 Quick Start

### Pré-requisitos
- Python 3.10+
- Git
- Conta Supabase (gratuita)
```

Instalação

```bash
# Clone
git clone https://github.com/aikiesan/NewLook.git
cd NewLook

# Ambiente virtual
python -m venv venv
source venv/bin/activate  # Linux/Mac
# venv\Scripts\activate  # Windows

# Dependências
pip install -r requirements.txt

# Configuração
cp .env.example .env
# Editar .env com suas credenciais

# Rodar
streamlit run app.py
```

Acesse: http://localhost:8501

📊 Features (MVP - 4 Semanas)

✅ Semana 1 (Em Andamento)

- [ ] Sistema de autenticação (Supabase)
- [ ] Landing page institucional
- [ ] Dashboard principal
- [ ] 3 perfis de usuário (Visitante, Autenticado, Admin)

⏳ Semana 2 (Planejado)

- [ ] Migração de 8 módulos de análise
- [ ] Integração Bagacinho IA
- [ ] Sistema de referências científicas

⏳ Semana 3 (Planejado)

- [ ] Análise MCDA configurável
- [ ] Integração MapBiomas
- [ ] 5 Layers de restrições ambientais

⏳ Semana 4 (Planejado)

- [ ] WCAG 2.1 Level AA compliance
- [ ] Responsividade mobile
- [ ] Deploy Railway
- [ ] Documentação completa

🏗️ Arquitetura

```
NewLook/
├── app.py              # Entry point
├── config/             # Configurações
├── src/
│   ├── auth/           # Autenticação
│   ├── ui/             # Interface
│   ├── data/           # Dados
│   ├── core/           # Lógica negócio
│   └── ai/             # IA
├── data/               # Dados aplicação
└── docs/               # Documentação
```

Ver STRUCTURE.md para detalhes.

🤝 Desenvolvimento

Branch Strategy

- `main`: Produção (protegido)
- `develop`: Desenvolvimento ativo
- `feature/*`: Features específicas

Commits

Seguimos Conventional Commits:

feat: Nova funcionalidade
fix: Correção de bug

docs: Documentação
refactor: Refatoração
test: Testes
chore: Manutenção

📄 Licença

MIT License - Ver LICENSE

📞 Contato

- **Repositório:** https://github.com/aikiesan/NewLook
- **Issues:** https://github.com/aikiesan/NewLook/issues

Desenvolvido com ❤️ para o futuro energético sustentável de São Paulo EOF

PASSO 5.5: CHANGELOG.md

```bash
cat > CHANGELOG.md << 'EOF'
# Changelog - CP2B Maps V3

Todas mudanças notáveis serão documentadas aqui.

Formato baseado em [Keep a Changelog](https://keepachangelog.com/).

---

## [Unreleased]

### 🚀 Planejado
- Sistema de autenticação com Supabase
- Landing page institucional
- Dashboard com dados municipais
- 8 módulos de análise migrados
- MCDA configurável
```

- WCAG 2.1 AA compliance

[3.0.0-alpha] - 2025-11-16

✨ Adicionado
- Estrutura inicial do projeto
- Configuração de ambiente
- Cópia de dados do project_map
- Documentação base

🔧 Configuração
- Setup Cursor + Claude Code
- Git workflow definido
- Scripts de sync e verificação

[2.0.0] - 2025-10-13 (project_map - Referência)

Ver: https://github.com/aikiesan/project_map/blob/main/CHANGELOG.md

Base para migração V3:
- 8 módulos funcionais
- Bagacinho IA com RAG
- Sistema WCAG 2.1 Level A
- 20+ referências científicas
- Dados validados FAPESP

Convenções:
- ✨ Adicionado: Novas features
- 🔧 Configuração: Mudanças de config
- 🐛 Corrigido: Bug fixes
- 📚 Documentação: Docs
- ♻️ Refatoração: Code refactor
- 🚀 Performance: Melhorias
EOF

---

6. PRIMEIRO PROMPT CLAUDE CODE (1h)

PASSO 6.1: Criar Ambiente Virtual Python

cd ~/cp2b-workspace/NewLook

Criar venv
python3 -m venv venv

Ativar
source venv/bin/activate # Mac/Linux
venv\Scripts\activate # Windows

Instalar dependências
pip install --upgrade pip
pip install -r requirements.txt

echo "✅ Ambiente Python configurado"

PASSO 6.2: Primeiro Prompt no Cursor (Composer)

Abrir Cursor:

cursor .

Pressionar `Cmd+Shift+L` (Composer) e colar:

CONTEXTO DO PROJETO:
Estou iniciando a migração do CP2B Maps V2 (Streamlit) para V3 com autenticação e arquitetura profissional.

OBJETIVO DESTA SESSÃO:
Criar a infraestrutura base da aplicação (app.py + componentes essenciais) para começar o desenvolvimento.

ESTRUTURA JÁ CRIADA:
- Pastas: ver STRUCTURE.md
- Config: config/settings.py (completo)
- Dados: data/ (copiados do project_map)
- Docs: README.md, CHANGELOG.md

TAREFA:

Crie os seguintes arquivos base:

1. app.py (entry point Streamlit):
   - Configuração de página (título, ícone, layout wide)
   - Importar settings de config/settings.py
   - Placeholder para sistema de navegação
   - Sistema de logging básico
   - Session state initialization
   - Comentários TODO para próximas etapas

2. src/utils/logger.py:
   - Configurar logging estruturado
   - Níveis: DEBUG, INFO, WARNING, ERROR
   - Output: console + arquivo logs/app.log
   - Formato: timestamp, level, message
   - Rotação de logs (10MB max)

3. src/__init__.py (e todos subdiretórios):
   - Arquivos vazios para tornar pacotes Python

4. src/ui/components/navbar.py:
   - Componente de barra de navegação
   - Menu: Home, Dashboard, Análises, Sobre
   - Placeholder para estado de autenticação (user: None por enquanto)
   - Estilo: verde gradient (#1E5128 → #4E9F3D)

REQUISITOS:
- Type hints em todas funções
- Docstrings estilo Google
- Imports organizados (stdlib → third-party → local)
- Logging em vez de print()
- TODO comments para features futuras

REFERÊNCIA:
Você pode consultar ../project_map/app.py para inspiração, mas modernize o código.

NÃO IMPLEMENTE AINDA:
- Autenticação (próxima sessão)
- Páginas completas (faremos módulo a módulo)
- Conexão com banco (ainda)

ENTREGA ESPERADA:

Estrutura mínima para rodar `streamlit run app.py` e ver uma página básica com navbar.

PASSO 6.3: Validar Resultado do Claude Code

Após Claude gerar os arquivos:

Testar se roda
streamlit run app.py

Deve abrir navegador em localhost:8501
Espera-se: Página básica com navbar e título "CP2B Maps V3"

Se funcionar:
git add .
git commit -m "feat: Add base application structure

- Create app.py entry point
- Implement structured logging
- Add navbar component
- Initialize project skeleton

Generated by: Claude Code (Composer)
Time: ~20min"

git push origin develop

---

7. CHECKLIST DETALHADO SEMANA 1 (16-22 Nov)

SEGUNDA-FEIRA 16/11 - DIA 1: SETUP + INFRAESTRUTURA ⚙️

Manhã (4h) - Setup Workspace

- [ ] 8:00-8:30: Executar todos scripts de setup (Seções 1-4)
- [ ] 8:30-9:00: Configurar Cursor + Claude Code
- [ ] 9:00-9:30: Copiar e verificar dados
- [ ] 9:30-10:30: Criar arquivos base (Seção 5)
- [ ] 10:30-11:00: Break + revisão
- [ ] 11:00-12:00: Primeiro prompt Claude (Seção 6)

Tarde (4h) - Autenticação Base

- [ ] 14:00-14:30: Criar conta Supabase, configurar projeto

[ ] 14:30-15:30: Criar schema database (SQL Editor):
-- PerfisCREATE TABLE profiles (id UUID REFERENCES auth.users PRIMARY KEY, full_name TEXT, institution TEXT, user_type TEXT DEFAULT 'visitante', verified BOOLEAN DEFAULT false, created_at TIMESTAMPTZ DEFAULT now());-- ProjetosCREATE TABLE projects (id SERIAL PRIMARY KEY, user_id UUID REFERENCES auth.users, name TEXT, description TEXT, config JSONB, created_at TIMESTAMPTZ DEFAULT now());-- RLSALTER TABLE profiles ENABLE ROW LEVEL SECURITY;ALTER TABLE projects ENABLE ROW LEVEL SECURITY;

-
- [ ] 15:30-16:30: Claude Code - Criar `src/auth/supabase_auth.py`
- [ ] 16:30-17:30: Claude Code - Criar `src/ui/pages/login.py`
- [ ] 17:30-18:00: Testar login básico, commit

Entregável Dia 1: App roda + Login page existe (pode não funcionar 100%)

TERÇA-FEIRA 17/11 - DIA 2: AUTENTICAÇÃO COMPLETA 🔐

Manhã (4h) - Sistema de Auth

- [ ] 8:00-9:00: Refinar login/logout
- [ ] 9:00-10:00: Claude - Criar página de registro
- [ ] 10:00-11:00: Integrar session state Streamlit
- [ ] 11:00-12:00: Criar middleware de proteção (@require_auth)

Tarde (4h) - Perfis de Usuário

- [ ] 14:00-15:00: Implementar 3 perfis (visitante, autenticado, admin)
- [ ] 15:00-16:00: Sistema de permissões (permissions.py)
- [ ] 16:00-17:00: Página "Minha Conta"
- [ ] 17:00-18:00: Testes end-to-end, commit

Entregável Dia 2: Auth 100% funcional (login, registro, perfis)

QUARTA-FEIRA 18/11 - DIA 3: LANDING PAGE 🏠

Manhã (4h) - Design Landing

- [ ] 8:00-9:00: Estrutura HTML/Streamlit da landing
- [ ] 9:00-10:30: Hero section com imagem de fundo
- [ ] 10:30-12:00: Features grid (3 colunas)

Tarde (4h) - Conteúdo e Estilo

- [ ] 14:00-15:00: Métricas impactantes (cards grandes)
- [ ] 15:00-16:00: Footer (logos, contato, links)
- [ ] 16:00-17:00: CSS customizado (gradientes verdes)
- [ ] 17:00-18:00: WCAG audit landing, commit

Entregável Dia 3: Landing page profissional e acessível

QUINTA-FEIRA 19/11 - DIA 4: DASHBOARD BASE 📊

Manhã (4h) - Estrutura Dashboard

- [ ] 8:00-9:00: Layout sidebar + main area
- [ ] 9:00-10:30: Migrar municipal_loader.py do project_map
- [ ] 10:30-12:00: 4 cards de métricas (dados reais)

Tarde (4h) - Visualizações

- [ ] 14:00-15:30: Mapa Folium choropleth municipal
- [ ] 15:30-17:00: Gráficos Plotly (distribuição + ranking)
- [ ] 17:00-18:00: Sistema de filtros (sidebar)

Entregável Dia 4: Dashboard com dados reais dos 645 municípios

---

SEXTA-FEIRA 20/11 - DIA 5: INTEGRAÇÃO E POLIMENTO 💎

Manhã (4h) - Integrar Auth + Dashboard

- [ ] 8:00-9:00: Conectar login → dashboard (fluxo completo)
- [ ] 9:00-10:00: Visitante vê demo limitado (10 municípios)
- [ ] 10:00-11:00: Autenticado vê tudo (645 municípios)
- [ ] 11:00-12:00: Admin panel básico (listar usuários)

Tarde (4h) - Refinamento Visual

- [ ] 14:00-15:00: Ajustes de estilo (consistência cores, fontes)
- [ ] 15:00-16:00: Loading states (spinners)

- [ ] 16:00-17:00: Navegação teclado (Tab order)
- [ ] 17:00-18:00: Commit final semana 1

Entregável Dia 5: Fluxo completo visitante → login → dashboard funcionando

---

SÁBADO 21/11 - DIA 6: VALIDAÇÃO ✅

Manhã (3h) - Testes

- [ ] 9:00-10:00: Testar fluxo como visitante
- [ ] 10:00-11:00: Testar fluxo como autenticado
- [ ] 11:00-12:00: Testar fluxo como admin

Tarde (4h) - Sessão Claude Chat (aqui)

- [ ] 14:00-15:30: Apresentar progresso da semana
- [ ] 15:30-16:30: Discutir issues encontradas
- [ ] 16:30-17:30: Planejar Semana 2
- [ ] 17:30-18:00: Criar Issues GitHub para próximos módulos

Entregável Dia 6: Retrospectiva documentada + Plano Semana 2

DOMINGO 22/11 - DIA 7: BUFFER / DESCANSO 🌴

- [ ] **Opcional:** Correção de bugs críticos
- [ ] **Recomendado:** Descansar para Semana 2
- [ ] Revisar checklist Semana 2 (se quiser se adiantar)

---

8. WORKFLOW GIT DIÁRIO

PADRÃO DE COMMITS

Tipos de Commit (Conventional Commits):

```
feat:     # Nova funcionalidade
fix:      # Correção de bug
docs:     # Documentação
style:    # Formatação (não afeta lógica)
refactor: # Refatoração de código
test:     # Adicionar testes
```

chore: # Manutenção (build, deps)

Formato:

<tipo>(<escopo>): <descrição curta>

<corpo opcional - detalhes>

<footer opcional - breaking changes, issues>

Exemplos Bons:

git commit -m "feat(auth): Add Supabase authentication integration

- Implement login/logout flow
- Add session management with Streamlit
- Create user profile fetching
- Add error handling for auth failures

Closes #1"

git commit -m "fix(dashboard): Correct municipality count in metrics card

Previously showed 640, now correctly shows 645.
Issue was in the SQL query filtering."

git commit -m "docs: Add architecture diagram to README

Include component interaction flowchart"

git commit -m "refactor(ui): Extract navbar to reusable component

Moved from app.py to src/ui/components/navbar.py for better modularity"

WORKFLOW DIÁRIO TÍPICO

Início do dia
git checkout develop
git pull origin develop
git checkout -b feature/nome-da-feature

Durante desenvolvimento (commits frequentes)

```
# ... fazer alterações ...
git add src/auth/login.py
git commit -m "feat(auth): Add login form UI"

# ... mais alterações ...
git add src/auth/session.py tests/test_auth.py
git commit -m "feat(auth): Implement session persistence

- Add session state management
- Store user data in st.session_state
- Add tests for session lifecycle"

# Fim do dia (ou feature completa)
git push origin feature/nome-da-feature

# No GitHub: Create Pull Request
# Revisar → Aprovar → Merge to develop

# Fim da semana
git checkout develop
git pull origin develop
git checkout main
git merge develop
git tag v3.0.0-week1
git push origin main --tags
```

TEMPLATE DE PULL REQUEST

Criar `.github/PULL_REQUEST_TEMPLATE.md`:

```
mkdir -p .github
cat > .github/PULL_REQUEST_TEMPLATE.md << 'EOF'
## Descrição
<!-- Descreva o que esta PR faz -->

## Tipo de Mudança
- [ ] 🐛 Bug fix
- [ ] ✨ Nova feature
- [ ] 📚 Documentação
- [ ] ♻️ Refatoração
- [ ] 🚀 Performance
- [ ] 🧪 Testes
```

```
## Checklist
- [ ] Código segue padrões do projeto
- [ ] Adicionei docstrings onde necessário
- [ ] Testei localmente
- [ ] Atualizei documentação (se aplicável)
- [ ] WCAG compliance (se UI)
- [ ] Sem console.log ou print() esquecidos

## Screenshots (se aplicável)
<!-- Cole screenshots aqui -->

## Issues Relacionadas
Closes #

## Tempo Estimado vs Real
Estimado: Xh | Real: Xh

## Observações
<!-- Qualquer contexto adicional -->
EOF
```

🎯 RESUMO EXECUTIVO - CHECKLIST SEGUNDA-FEIRA (16/11)

MANHÃ (8:00 - 12:00)

```
# Terminal 1: Setup workspace
cd ~
mkdir cp2b-workspace
cd cp2b-workspace
git clone https://github.com/aikiesan/project_map.git
git clone https://github.com/aikiesan/NewLook.git

# Terminal 2: Configurar NewLook
cd NewLook
# Executar TODOS os scripts das Seções 1-5 (copiar/colar)

# Terminal 3: Abrir Cursor
cursor .
# Configurar extensões
# Usar primeiro prompt da Seção 6
```

Resultado esperado às 12:00:
✅ Workspace estruturado
✅ Dados copiados
✅ app.py rodando (streamlit run app.py)
✅ Navbar básica aparecendo

TARDE (14:00 - 18:00)

Supabase:
1. Criar conta em https://supabase.com
2. Novo projeto "cp2b-maps"
3. SQL Editor → executar schema (Seção 7, Dia 1)
4. Copiar URL e keys para .env

Cursor (Claude Code):
Cmd+Shift+L → Prompt:
"Crie src/auth/supabase_auth.py com:
- Cliente Supabase configurado
- Funções: login(), logout(), get_user()
- Error handling robusto
- Type hints e docstrings"

Testar:
streamlit run app.py
Tentar fazer login (vai falhar, ok por enquanto)

Commit final do dia:
git add .
git commit -m "feat(auth): Add Supabase integration and login page"
git push origin develop

Resultado esperado às 18:00:
✅ Schema Supabase criado
✅ Auth skeleton existe
✅ Código commitado

📞 SUPORTE E PRÓXIMOS PASSOS

Se Tiver Problemas:

1. **Dados não copiaram:** Verificar caminhos em `scripts/sync_data.py`

2. **Cursor não abre:** Reinstalar de https://cursor.sh
3. **Supabase erro:** Verificar .env tem as keys corretas
4. **Streamlit não roda:** `pip install -r requirements.txt` novamente

Sessões de Validação (Sábados):

Todo sábado, venha aqui (Claude Chat) com:

- Código implementado na semana
- Issues encontradas
- Dúvidas técnicas
- Planejar próxima semana

PRÓXIMA SESSÃO PLANEJADA:

Sábado 21/11 - 14:00:

Tópicos:

1. Revisar auth implementado
2. Validar landing page e dashboard
3. Discutir migração dos 8 módulos (Semana 2)
4. Ajustar cronograma se necessário

✅ CONFIRMAÇÃO FINAL

Você está pronto para começar segunda-feira (16/11/2025) se:

- [ ] Leu todo este documento (30min)
- [ ] Entendeu estrutura de pastas (STRUCTURE.md)
- [ ] Cursor instalado
- [ ] Git configurado
- [ ] Conta Supabase criada (pode fazer segunda cedo)
- [ ] Terminal/bash funcionando
- [ ] Python 3.10+ instalado

Tempo estimado Dia 1 (segunda): 8 horas

Primeira ação segunda 8:00: Executar Seção 1 (Preparação Workspace)

BOA SORTE! 🚀

Qualquer dúvida antes de segunda, pode perguntar aqui que eu ajudo!

🎁 BÔNUS: ATALHOS DO CURSOR

Cmd+K / Ctrl+K: Chat inline (editar código diretamente)
Cmd+L / Ctrl+L: Painel lateral Claude
Cmd+Shift+L: Composer (multi-file editing) ⭐
Cmd+/: Toggle comment
Cmd+P: Quick open file
Cmd+Shift+P: Command palette
Cmd+B: Toggle sidebar
Cmd+`: Terminal
Cmd+Shift+F: Search in files

Mais produtivo: Composer (Cmd+Shift+L) para criar múltiplos arquivos de uma vez!