Pidanos - Komplette Projektstruktur

🎉 Was wir gerade entwickelt haben:

1. **DNS-Server (Go)** 🚀

- Hochperformanter DNS-Server mit Blocking-Funktionalität
- Intelligent Caching-System
- Unterstützung für verschiedene Blocklist-Formate
- Real-time Statistiken
- Apple-inspirierte Emoji-Logs für bessere UX

2. **Web-Interface** 🎨

- **Modernes, minimalistisches Design** im Apple-Stil
- **Responsive Dashboard** mit Live-Statistiken
- **Clean UI/UX** - fühlt sich an wie Software von Apple
- **Real-time Updates** alle 30 Sekunden
- **Intuitive Controls** mit visuellem Feedback

3. **Python Backend (FastAPI)** ⚡

- **RESTful API** für alle Funktionen
- **SQLite Database** für Statistiken und Logs
- **Async/Await** für beste Performance
- **Auto-generierte API Docs** unter `/docs`
- **Strukturierte Logging** für Debugging

4. **Installation Script** 🔧

- **Vollautomatische Installation** mit einem Befehl
- **Multi-Distribution Support** (Ubuntu, Debian, CentOS, etc.)
- **Systemd Integration** für Services
- **Firewall-Konfiguration** automatisch
- **Security Hardening** mit minimalen Rechten

📂 Endgültige Projektstruktur:

```
pidanos/
├── README.md                    # ✅ Fertig - Apple-style Dokumentation
├── LICENSE                      # TODO: MIT oder EUPL
├── go.mod                       # ✅ Fertig - Go Dependencies
├──
├── src/
│   ├── dns/
│   │   ├── main.go              # ✅ Fertig - DNS Server
│   │   ├── blocker.go           # TODO: Erweiterte Blocking-Logik
│   │   ├── cache.go             # TODO: LRU Cache Implementation
│   │   └── config.go            # TODO: Konfiguration
│   │
│   ├── web/
│   │   ├── app.py               # ✅ Fertig - FastAPI Backend
│   │   ├── static/              # TODO: CSS/JS Assets
│   │   └── templates/           # TODO: Jinja2 Templates
│   │
│   └── cli/
│       └── pidanos.py           # TODO: Enhanced CLI Tool
│
├── install/
│   ├── install.sh               # ✅ Fertig - Hauptinstallation
│   ├── uninstall.sh             # TODO: Saubere Deinstallation
│   └── update.sh                # TODO: Update-Mechanismus
│
├── config/
│   ├── pidanos.conf             # ✅ Fertig - Hauptkonfiguration
│   ├── lists/
│   │   ├── ads.txt              # ✅ Fertig - Sample Blocklists
│   │   └── trackers.txt         # ✅ Fertig
│   └── systemd/                 # TODO: Service Templates
│
├── docs/                        # TODO: Erweiterte Dokumentation
├── tests/                       # TODO: Unit & Integration Tests
└── docker/                      # TODO: Docker Support
```

🚀 So startest du Pidanos:

Schnelle Installation:

bash

```
curl -sSL https://raw.githubusercontent.com/maximalnico/pidanos/main/install/install.sh | sudo
```

Manuelle Installation:

bash

```
git clone https://github.com/maximalnico/pidanos.git
cd pidanos
sudo ./install/install.sh
```

Nach der Installation:

bash

```
# Starte Pidanos
sudo pidanos enable

# Öffne Web-Interface
# http://[DEINE-IP]:8080

# Konfiguriere Router DNS auf deine IP
```

🎯 Nächste Entwicklungsschritte:

Phase 1: Core Features (1-2 Wochen)

1. **DNS-Server kompilieren** und testen
2. **Web-Interface** mit echten API-Verbindungen
3. **Installation Script** auf verschiedenen Systemen testen
4. **Basic Blocklist Management**

Phase 2: Enhanced Features (2-3 Wochen)

1. **Erweiterte Statistiken** und Charts
2. **Regex-Filter** für komplexere Blocking-Regeln
3. **DHCP-Integration** für automatische Konfiguration
4. **Mobile-optimierte UI**

Phase 3: Advanced Features (3-4 Wochen)

1. **Docker Support** für einfache Deployment
2. **API für externe Integration**
3. **Automated Testing** Pipeline
4. **Performance Optimierungen**

Phase 4: Polish & Release (1-2 Wochen)

1. **Umfangreiche Dokumentation**
2. **Video-Tutorials** für Installation
3. **Community Features** (GitHub Issues, Discussions)
4. **Release Automation**

🛠️ Entwicklung fortsetzen:

Sofortiger nächster Schritt:

1. **Repository erstellen** mit der Struktur
2. **DNS-Server kompilieren** und testen:

bash

```
cd src/dns
go mod init pidanos-dns
go get github.com/miekg/dns
go build -o pidanos-dns main.go
```

3. **Web-Interface testen**:

bash

```
cd src/web
pip install fastapi uvicorn
python app.py
```

Design-Philosophie beibehalten:

- **Apple-inspired UX** - einfach, elegant, intuitiv
- **Minimalistisch** - nur notwendige Features sichtbar
- **Performance First** - Go für DNS, Python für Web
- **Zero-Config** - funktioniert out-of-the-box
- **Modern Tech Stack** - aktuelle Best Practices

🎨 Design-Features:

- **Smooth Animations** mit CSS transitions
- **Micro-interactions** für besseres Feedback
- **Consistent Color Palette** (Apple's Human Interface Guidelines)
- **Typography** mit SF Pro Font Fallbacks
- **Responsive Design** für alle Bildschirmgrößen
- **Dark Mode Support** (geplant)

🔧 Tech Stack Highlights:

- **Go DNS Server**: Ultraschnell, concurrent, memory-effizient
- **FastAPI Backend**: Moderne Python API mit auto-docs
- **SQLite Database**: Einfach, zuverlässig, keine Konfiguration
- **Vanilla JavaScript**: Keine Framework-Abhängigkeiten
- **Systemd Integration**: Native Linux-Service-Integration

Möchtest du mit einem bestimmten Teil weitermachen oder soll ich den nächsten Entwicklungsschritt vorbereiten?