



NT

# ENTWICKLER DOKUMENTATION

NetTools Suite

---

## Inhalt:

- Architektur & Projektstruktur
- UI-Komponenten & Design-System
- Backend-Module & Tools
- Neue Tools hinzufuegen
- Best Practices & Patterns

Version 2.0  
Stand: Dezember 2024

## Inhaltsverzeichnis

-1. Projektübersicht	3
-2. Architektur	4
-3. Verzeichnisstruktur	5
-4. Technologie-Stack	6
-5. Hauptkomponenten	7
-6. UI-Module	9
-7. Tools-Module	10
-8. Design-System	11
-9. Konfiguration & Persistenz	12
-10. Neue Tools hinzufügen	13
-11. Best Practices	14
-12. Bekannte Probleme	15

## 1. Projektuebersicht

Die NetTools Suite ist eine Desktop-Anwendung fuer Netzwerk-Administration, entwickelt mit Python und CustomTkinter. Die Anwendung folgt einer modularen Architektur, die einfache Erweiterung und Wartung ermoeglicht.

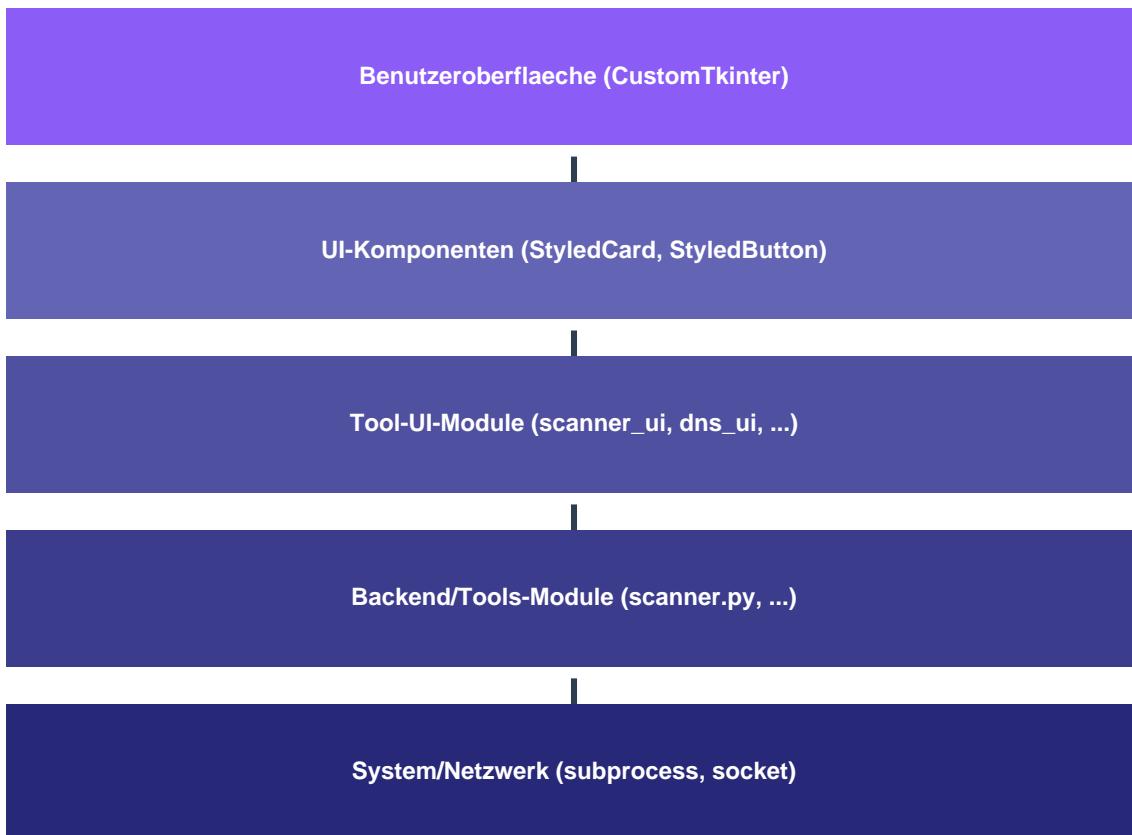
### Kernziele

- Moderne, benutzerfreundliche GUI mit Dark Theme
- Modularer Tool-Architektur fuer einfache Erweiterung
- Plattformuebergreifend (primaer Windows)
- Einfache Erweiterbarkeit durch klare Strukturen

## 2. Architektur

### Schichtenmodell

Die Anwendung folgt einem klaren Schichtenmodell:



### Kommunikationsfluss

- Benutzer-Aktion -> UI-Modul (z.B. scanner\_ui.py)
- Validierung der Eingabe im UI-Modul
- Backend-Modul führt Operation aus (Threading)
- System-Aufruf (ping, tracert, etc.)
- Ergebnis-Callback zurück zum UI
- UI-Update im Haupt-Thread via app.after()

#### WICHTIG: Threading-Regel

Lange Operationen IMMER in separatem Thread ausführen. UI-Updates NUR über self.app.after(0, callback)!

### 3. Verzeichnisstruktur

```
nettools/
|-- nettools_app.py          # Hauptanwendung & Entry Point
|-- design_constants.py      # Farben, Schriften, Abstaende
|-- ui_components.py         # Wiederverwendbare UI-Komponenten
|-- requirements.txt          # Python-Abhaengigkeiten
|
|-- ui/                      # UI-Module fuer jeden Tool
|   |-- dashboard_ui.py       # Dashboard-Ansicht
|   |-- scanner_ui.py        # IPv4-Scanner UI
|   |-- dns_ui.py            # DNS-Lookup UI
|   |-- traceroute_ui.py     # Traceroute UI
|   |-- settings_ui.py       # Einstellungen
|   +-- ...
|
|-- tools/                   # Backend-Logik
...
... (6 weitere Zeilen)
```

## 4. Technologie-Stack

### Hauptabhaengigkeiten

Bibliothek	Version	Verwendung
customtkinter	>= 5.2.0	Moderne GUI-Komponenten
tkinter	(builtin)	Basis-GUI-Framework
requests	>= 2.31.0	HTTP-Anfragen
dnspython	>= 2.4.0	DNS-Abfragen
beautifulsoup4	>= 4.12.0	HTML-Parsing
pyinstaller	>= 6.0.0	Executable-Erstellung
matplotlib	>= 3.7.0	Visualisierungen

### System-Tools (extern)

- ping / tracert / pathping - Windows-Netzwerktools
- arp - ARP-Tabelle auslesen
- nslookup - DNS-Abfragen (Fallback)
- iperf3 - Bandbreiten-Tests (optional)

## 5. Hauptkomponenten

### 5.1 nettools\_app.py - Hauptanwendung

Dies ist der zentrale Entry Point und enthaelt die NetToolsApp Klasse:

```
class NetToolsApp(ctk.CTk):
    def __init__(self):
        # Fenster-Initialisierung
        # Konfiguration laden
        # UI aufbauen

    def create_sidebar(self):
        # Erstellt die Navigation

    def switch_page(self, page_id):
        # Wechselt zwischen Tools

    def show_toast(self, message, type):
        # Zeigt Benachrichtigungen
```

### Wichtige Methoden

Methode	Beschreibung
__init__()	Initialisiert App, laedt Confi
create_sidebar()	Erstellt Navigation
switch_page(id)	Wechselt zu einem Tool
show_toast(msg, type)	Zeigt Benachrichtigung
get_enabled_tools()	Laedt aktivierte Tools
is_admin()	Prueft Admin-Rechte

### 5.2 design\_constants.py

Zentrale Definition aller visuellen Konstanten:

```
COLORS = {
    "bg_primary": ("#1A1B26", "#1A1B26"),
    "electric_violet": ("#8B5CF6", "#A78BFA"),
    "neon_cyan": ("#00D9FF", "#67E8F9"),
    "success": ("#22C55E", "#4ADE80"),
    "danger": ("#EF4444", "#F87171"),
}
```

```
SPACING = {"xs": 4, "sm": 8, "md": 16, "lg": 24, "xl": 32}
FONT_SIZES = {"title": 24, "heading": 18, "body": 12}
```

## 6. UI-Module

### Struktur eines UI-Moduls

```
class ToolNameUI:  
    def __init__(self, app, parent):  
        self.app = app  
        self.parent = parent  
        self.create_ui()  
  
    def create_ui(self):  
        main_frame = ctk.CTkScrollableFrame(self.parent)  
        main_frame.pack(fill="both", expand=True)  
  
        # Titel  
        title = ctk.CTkLabel(main_frame, text="Tool Name")  
        title.pack(anchor="w")  
  
        # Eingabebereich mit StyledCard  
        ... (6 weitere Zeilen)
```

#### TIPP: UI-Konsistenz

Immer COLORS und SPACING aus design\_constants verwenden. StyledCard fuer Container, StyledButton fuer Aktionen.

## 7. Tools-Module

### Struktur eines Backend-Moduls

```
class ToolName:  
    @staticmethod  
    def run(target: str, options: dict = None) -> Dict:  
        try:  
            if not target:  
                return {"success": False, "error": "Kein Ziel"}  
  
            result = subprocess.run(  
                ["tool", target],  
                capture_output=True,  
                text=True,  
                timeout=60  
            )  
  
            return {  
... (8 weitere Zeilen)
```

## 8. Design-System

### Farbschema

Farbe	Hex	Verwendung
Electric Violet	#8B5CF6	Primaerfarbe, aktive Elemente
Neon Cyan	#00D9FF	Akzente, Links
Success Green	#22C55E	Erfolg, positive Werte
Danger Red	#EF4444	Fehler, negative Werte
Warning Yellow	#F59E0B	Warnungen
Background	#1A1B26	Haupt-Hintergrund

### Abstands-System

Name	Wert	Verwendung
xs	4px	Minimal (zwischen Icons)
sm	8px	Klein (verwandte Elemente)
md	16px	Standard (Padding in Cards)
lg	24px	Gross (zwischen Sektionen)
xl	32px	Extra gross (Seiten-Raender)

## 9. Konfiguration & Persistenz

### Konfigurations-Datei

Speicherort: `~/.nettools/config.json`

```
{  
    "favorite_tools": ["scanner", "dns", "traceroute"],  
    "enabled_tools": ["dashboard", "scanner", "dns"],  
    "window": {  
        "width": 1400,  
        "height": 900  
    },  
    "scan_profiles": {  
        "office": {  
            "ip_range": "192.168.1.0/24",  
            "ports": "22,80,443"  
        }  
    }  
}
```

### Verlaufs-Speicherung

```
~/.nettools/  
|-- config.json      # Haupt-Konfiguration  
|-- scans.json       # Scan-Verlauf  
|-- traceroutes.json # Traceroute-Verlauf  
|-- dns_lookups.json # DNS-Verlauf  
+-- port_scans.json # Port-Scan-Verlauf
```

## 10. Neue Tools hinzufuegen

### Schritt 1: Backend-Modul erstellen

Erstelle tools/neues\_tool.py:

```
class NeuesTool:  
    @staticmethod  
    def run(target: str) -> dict:  
        # Implementierung  
        return {"success": True, "output": "...."}
```

### Schritt 2: UI-Modul erstellen

Erstelle ui/neues\_tool\_ui.py:

```
from ui_components import StyledCard, StyledButton  
  
class NeuesToolUI:  
    def __init__(self, app, parent):  
        self.app = app  
        self.parent = parent  
        self.create_ui()  
  
    def create_ui(self):  
        # UI-Code hier  
        pass
```

### Schritt 3: In nettools\_app.py registrieren

- Import hinzufuegen: from ui.neues\_tool\_ui import NeuesToolUI
- In nav\_categories einfuegen: ('neues\_tool', 'Icon', 'Name', 'Tooltip')
- In switch\_page Handler hinzufuegen
- In get\_enabled\_tools Default hinzufuegen

## 11. Best Practices

### Threading

#### WICHTIG: Wichtig

IMMER lange Operationen in separatem Thread. UI-Updates NUR ueber self.app.after(0, callback). daemon=True setzen!

### Error Handling

```
try:  
    result = dangerous_operation()  
except SpecificError as e:  
    self.app.show_toast(f"Fehler: {e}", "error")  
except Exception as e:  
    print(f"Unerwarteter Fehler: {e}")  
    self.app.show_toast("Ein Fehler ist aufgetreten", "error")
```

### Encoding (Windows)

```
# Bei subprocess:  
result = subprocess.run(cmd, capture_output=True)  
# NICHT text=True bei moeglichen Sonderzeichen!  
# Manuell dekodieren:  
output = result.stdout.decode('utf-8', errors='replace')
```

## 12. Bekannte Probleme

Problem	Lösung
Emoji-Breite variiert	Fixe Breite mit width Parameter
Windows-Encoding	Bytes-Modus + manuelle Dekodierung
Admin-Rechte noetig	is_admin() pruefen, restart_as
PSEXEC Cross-Domain	Feature temporaer deaktiviert

### INFO: Weiterentwicklung

Geplant: Netzwerk-Topologie-Visualisierung, PDF-Export, Plugin-System, Dark/Light Mode Toggle