
Sommerprüfung 2023

**Ausbildungsberuf**

Fachinformatiker/ Fachinformatikerin (VO 2020) Fachrichtung:

Anwendungsentwicklung

# Entwickeln einer Webanwendung für einen Router

**Auszubildender:** Andrii Dragozhynskyi
**Abgabetermin:** 15.05.2023

**Ausbildungsbetrieb:** Lutz und Grub AG
Frankenstraße 160
90461 Nürnberg

## Inhaltverzeichnis

# 1 Analysephase

Ich absolviere derzeit eine Umschulung in der Fachrichtung Anwendungsentwicklung an der LUTZ & GRUB AG Akademie. Die Lutz & Grub AG hat sich auf die Umschulung und Weiterbildung von Mitarbeitern im Bereich Microsoft Informationstechnologie spezialisiert. Das Unternehmen wurde 1996 von Martin Grub und Reinhard Lutz gegründet und hat Niederlassungen in Karlsruhe, Nürnberg, Heilbronn und Stuttgart. Aktuell bietet das Unternehmen drei Umschulungsrichtungen an:

- Fachinformatiker in der Anwendungsentwicklung (IHK)
- Fachinformatiker in der Systemintegration (IHK)
- Kaufmann/Kauffrau im E-Commerce (IHK)

Im Rahmen meiner Umschulung absolviere ich derzeit ein Praktikum bei der ASCEND GmbH.

Die ASCEND GmbH ist ein IT-Systemhaus mit Hauptsitz in Nürnberg, das im Jahr 2010 gegründet wurde. Das Unternehmen bietet hauptsächlich folgende Dienstleistungen an:

- IT-Lösungen für große Unternehmensnetzwerke, einschließlich Cloud-Anwendungen
- Kommunikationslösungen auf der Baustelle
- Herstellung einer Verbindung zu einem lokalen VPN für Ihr Unternehmen als ersten Schritt auf mobilen Geräten
- Kabellose Lösungen für Veranstaltungen jeglicher Art und Größenordnung
- Kabellose Gastlösungen.

## 1.1 Kundengespräch

Die ASCEND GmbH entwickelt derzeit ein eigenes mobiles Gerät mit integriertem Router, einem Akku und einem Minicomputer mit einer Bildschirmauflösung von 800x480 Pixel. Dieser Komplex wird für öffentliche Veranstaltungen und Events genutzt, unabhängig davon, ob sie drinnen oder draußen stattfinden. Das Gerät reduziert die Zeit für den Auf- und Abbau der Geräte, ihre Einrichtung und die Kosten für den Kauf zusätzlicher Geräte.

Die ASCEND GmbH hat einen internen Auftrag zur Entwicklung einer Webanwendung zur Routersteuerung mittels der Hersteller-API erstellt. Diese ermöglicht es, gewisse Parameter mittels eines Mini-Computers mit Touch-Screen schnell anzusehen und anzupassen und ich als Praktikant übernehme diese Aufgabe. Derzeit ist es nur umständlich, den Router über die interne Weboberfläche zu steuern.

## 1.2 Ist-Analyse

Während eines Kundegesprächs mit meinem Geschäftsführer Johannes Fickeis, der als interner Kunde im Rahmen dieses Projekts fungiert, stellte sich Folgendes heraus:

- interne Weboberfläche den Router zeigt alle erforderlichen Daten auf separaten Seiten an, wodurch es schwieriger wird, Daten in Echtzeit zu verfolgen und zu verwalten.
- der Zugriff auf die interne Weboberfläche des Routers zur Überwachung und Verwaltung ist nur über die Administratorzugriffsebene möglich

## 1.3 Lösungsmöglichkeiten darstellen/ Alternativen

Nach dem Kundengespräch und Analyse der Hauptprozesse haben sich folgenden Diagramme entwickelt und dargestellt:

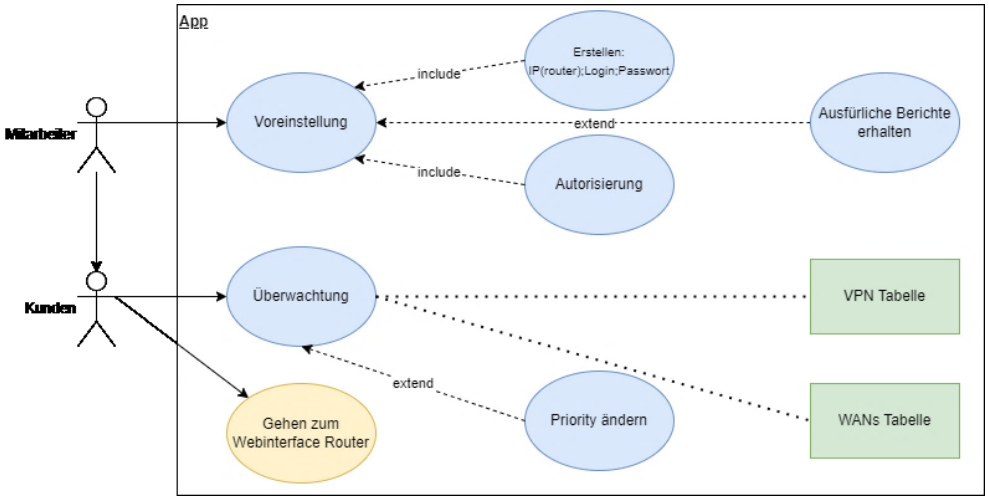



Abbildung 1: Use-Case Diagramm.

Der Prozess ist in zwei Teile unterteilt: Vorkonfiguration und die Verwendung der Anwendung.

Vorkonfiguration:

Der IT-Mitarbeiter konfiguriert die Anwendung vorab, indem er die IP-Adresse des Routers sowie den Benutzernamen und das Passwort auf der Konfigurationsseite "Support" eingibt. Die Anwendung authentifiziert sich auf der Konfigurationsseite "Support" und stellt dann eine Verbindung zum Router her. Wenn die Verbindung erfolgreich ist, wird ein Bericht über eine erfolgreiche oder nicht erfolgreiche Verbindung angezeigt. Es besteht die Möglichkeit, den Empfang von Berichten vom Router zu überprüfen. Wenn die Verbindung nicht erfolgreich ist, müssen die IP-Adresse, der Login und das Passwort auf ihre Richtigkeit überprüft werden. Nach erfolgreicher Verbindung mit dem Router gelangt man zur Hauptseite der Anwendung «Dashboard». Dort werden drei Datengruppen angezeigt:

- Die erste Datengruppe umfasst das Router-Modell und die IP-Adresse des Routers.
- Die zweite Datengruppe enthält die WAN-Sequenznummer der Verbindungen, den Verbindungsnamen, den Verbindungsstatus (Ein/Aus), den Verbindungsstatus der Verbindungen (connection, connected, disabled, SIM PIN required incorrect, obtaining IP Address.., No device detected), den Verbindungstyp (Ethernet, cellular, Wi-Fi), die Signalstärke (), den Prioritätsstatus dieser Verbindung (1, 2, 3, 4) und Tasten zum Ändern der Priorität (Set Priority 1/ Set Priority 2).
- Die dritte Datengruppe umfasst die Sequenznummer der VPN-Verbindungen, den Namen der VPN-Verbindungen, den Status der VPN-Verbindungen (connected/), den Namen der WAN-Verbindungen, die Qualitätsindikatoren für jede WAN-Verbindung (Latency, Drop Paket, Tx Geschwindigkeit, Rx Geschwindigkeit), allgemeine Daten für jede VPN-Verbindung wie Geschwindigkeit Tx und Rx). Nach einer visuellen Überprüfung wird überprüft, ob die drei Datengruppen vorhanden sind. Diese ändern sich, wenn sich der WAN-Prioritätsstatus der Verbindung ändert. Der Validierungsprozess gilt dann als abgeschlossen.

Verwendung der App:

- Der Mitarbeiter des Unternehmens oder der Kundenach der Vorkonfiguration über eine Verknüpfung auf dem mobilen Gerät zur Hauptseite der Anwendung "Dashboard". Falls keine Vorkonfiguration vorgenommen wurde, wird die Anwendung automatisch zur Konfigurationsseite "Support" weitergeleitet. Auf der Hauptseite der Anwendung kann der Mitarbeiter oder Kunde die WAN-Verbindungsdaten überwachen, den Prioritätsstatus der WAN-Verbindungen manuell ändern und die VPN-Verbindungsdaten in Echtzeit verfolgen.
- Um detailliertere Berichte zu erhalten oder die Einstellungen für die Verbindung zum Router zu ändern, kann man zur "Support"-Seite gehen, wofür eine Authentifizierung erforderlich ist.

```
start
-> if (Vorkonfiguration durch IT-Mitarbeiter) then
   -> Konfigurationsseite "Support" aufrufen
   -> Authentifizierung durchführen
   -> Verbindung zum Router herstellen
   -> if (Verbindung erfolgreich) then
      -> Bericht über erfolgreiche Verbindung anzeigen
      -> Datengruppen auf der Hauptseite "Dashboard" anzeigen (Router-Modell,
            IP-Adresse des Routers, WAN-Verbindungsdaten, VPN-Verbindungsdaten)
      -> Validierung abschließen
   -> else
      -> Überprüfung von IP-Adresse, Login und Passwort
   endif
-> else
   -> Weiterleitung zur Konfigurationsseite "Support"
endif
-> Mitarbeiter/Kunde gelangt zur Hauptseite "Dashboard"
-> Mitarbeiter/Kunde kann WAN-Verbindungsdaten überwachen
-> Mitarbeiter/Kunde kann Prioritätsstatus von WAN-Verbindungen manuell ändern
-> Mitarbeiter/Kunde kann VPN-Verbindungsdaten in Echtzeit verfolgen
-> if (detaillierte Berichte oder Router-Einstellungen ändern) then
   -> Weiterleitung zur Seite "Support" oder Konfigurationsseite "WebAdmin"
   -> Authentifizierung durchführen
   -> Zugang zu Berichten oder Webinterface des Routers ermöglichen
   -> Logout durchführen
endif
-> Anwendung schließen oder Webbrowser beenden
```

- Wenn die Router-Einstellungen geändert werden sollen, kann man zur Konfigurationsseite der Anwendung "WebAdmin" gehen, wo sich ein Link zum Webinterface des Routers befindet. Um sich auf der Webinterface-Seite des Routers anzumelden, ist eine Authentifizierung erforderlich.

Um die Verwendung der Anwendung abzuschließen, muss der Kunde die Anwendungsseite schließen oder den Webbrowser beenden.

*Abbildung 2: Anwendungsalgorithmus*

## 1.4 Projektziele definieren

Ziel des Projektes ist es, eine Webanwendung zu entwickeln, die lokal auf einem Raspberry PI mit einem Touchscreen, mit einer Auflösung von 800x480 Pixeln, läuft. Der logische Teil des Projekts wird mit der Programmiersprache PHP entworfen und erstellt, ohne eine Datenbank auf der Grundlage der Kundenanforderungen zu erstellen. Der grafische Teil des Projekts wird über den Webbrowser Mozilla Firefox präsentiert. Die Webanwendung zeigt unter anderem den aktuellen Status von WAN Verbindungen, die Mobilfunk Signalstärke und den Status der VPN Tunnel, in Echtzeit, an. Die Webanwendung kann nur vom Terminal aus aufgerufen werden und ist sowohl für Mitarbeiter des Unternehmens, die bei Veranstaltungen oder Events eine Internetverbindung bereitstellen, als auch für Personen, die keine Experten auf diesem Gebiet sind, gedacht. Daher soll das von ASCEND GmbH entwickelte grafische Frontend eine schnelle Statusübersicht und grundlegende Entstörungsmöglichkeiten geben. Bei Problemen die darüber hinaus gehen, soll über diesen Zugang ein einfacher Zugang zum Webinterface des Routers möglich sein. Bei der Entwicklung der Anwendung muss ich mich nicht zusätzlich mit der Konfiguration des Routers, des Servers befassen. Der Standort der Anwendung wird vom Kunden bestimmt. Die Grundfarben und die Schriftart müssen den Anforderungen des Kunden entsprechen.

# 2 Planungsphase

## 2.1 Projektplanung

| Thema | Zeit |
|---|---|
| **1. Analysephase** | |
| Kundengespräch | 2 Std. |
| Ist-Analyse | 2,0 Std. |
| Soll-Analyse | 2,0 Std. |
| Lösungsmöglichkeiten darstellen/ Alternativen | 1,0 Std. |
| Projektziele definieren | 1,0 Std. |
| **2. Planungsphase** | |
| Projektplanung | 2,0 Std. |
| Kostenkalkulation | 2,0 Std. |
| **3. Aus-/Durchführung** | |
| Information strukturieren und entwickeln | 4,0 Std. |
| Programmlogik zur Ansprache der API entwickeln | 12,0 Std. |
| Webanwendung entwickeln | 8,0 Std. |
| Verarbeitung die Darstellung der empfangenen Informationen von Router | 8,0 Std. |
| Entwicklung eines Parameterverwaltungssystems zwischen der Webanwendung und dem Router | 9,0 Std. |
| Konfigurierung den Empfang und die Anzeige von Informationen in Echtzeit. | 8,0 Std. |
| Umfangreiche Testung | 3,0 Std. |
| **4. Projektergebnisse** | |
| Abschlusstest | 3 Std. |
| Soll-Ist-Vergleich | 3 Std. |
| Übergabe an den Kunden | 2 Std. |
| **5. Projektabschlussphase** | |
| Projektdokumentation | 8 Std. |
| **Gesamtzeit** | **80,0 Std.** |

## 2.2 Kostenkalkulation

In diesem Projekt, das von mir als Praktikant durchgeführt wird, habe ich Herrn Obed Klein als meinen Projektbetreuer, der hauptsächlich eine beratende Funktion hat. Für die Kostenkalkulation des Projekts verwende ich einen pauschalen Stundensatz.

| Vorgang | | Zeit in Stunden | Kosten 1 Stunden | Gesamtkosten |
|---|---|---|---|---|
| Lohnkosten | Andrii Dragozhynskyi | 80 | 13,00 Euro | 1.040,00 Euro |
| Gemeinkosten | 95% des Lohnkosten | | | 988,00 Euro |
| Lohnkosten | Herr Obed Klein | 16 | 50,00 Euro | 800,00 Euro |
| Insgesamt : | | | | 2.828,00 Euro |

# 3 **Aus/Durchführung**

## 3.1 Information strukturieren und entwickeln

Für die Durchführung des Projekts habe ich einen Arbeitsplatzrechner mit Windows 11 Pro verwendet. Darauf waren bereits vorinstalliert: Microsoft Visual Studio Code 2023, MIB Browser, XAMPP und FileZilla. Bei der Entwicklung der Software habe ich mich auf folgende Informationsquellen gestützt: die offizielle PHP-Dokumentation (https://www.php.net), Stack Overflow (https://stackoverflow.com/) und Metanit (https://metanit.com/), sowie die Dokumentationen zur SNMP-Erweiterung (https://www.php.net/manual/de/intro.snmp.php) und zur cURL-Erweiterung (https://www.php.net/manual/de/intro.curl.php). Zudem habe ich die Peplink-Router-API-Dokumentation (https://knowhow.peplink.ninja/documentation/peplink-router-api-documentation/) und das Peplink-Router-API-Handbuch (https://download.peplink.com/resources/Peplink-Router-API-Documentation-for-Firmware-8.1.1.pdf) verwendet. Zusätzlich habe ich die MIB-Dateien für SNMP von der Peplink-Website heruntergeladen (https://www.peplink.com/support/downloads/miscellaneous-downloads/#snmp-mibs).

Nach einer ausführlichen Analyse des Anwendungsalgorithmus habe ich in einem Gespräch mit meinem Projektbetreuer, Herrn Obed Klein, die nächsten Entwicklungsschritte für die App identifiziert:

- Entwicklung einer Funktion, die den Zugriff auf die API über eine benutzerfreundliche Benutzeroberfläche ermöglicht. Dabei sollten Konfigurationsoptionen für HTTP-Anfragen wie IP-Adresse, Login, Passwort sowie GET- und POST-Parameter bereitgestellt werden;
- Entwicklung von Tools für die sichere Eingabe, Speicherung und Weitergabe von Parametern (API-Request);
- Entwicklung von Tools zum Abrufen, Anzeigen und Speichern von API-Antworten (API-Response);
- Entwicklung von Funktionen zur dynamischen Erstellung von Tabellen basierend auf den erhaltenen API-Responses vom Router;
- Entwicklung von Verwaltungstools zur Änderung des Prioritätsstatus der WAN-Verbindung;
- Entwicklung von Tools zur Echtzeit-Anzeige von Tabellendaten (Aktualisierungsintervall: 5-10 Sekunden).

Für die Umsetzung ich werde zwei Arten von Anfragen über die Router-API verwendet, sowie eine SNMP-Abfrage zur Datenerfassung. Die Router-API stellt jedoch sicher, dass Sie genügend Informationen erhalten, indem ich nur vier Arten von POST- und GET- Abfragen verwende wie POST /api/login, GET /api/status.wan.connection, GET /api/status.pepvpn, POST /api/config.wan.connection.priority, bietet aber keine Latency-, Drop Paket, Upload- und Download-Daten, die über SNMP mit der MIB-Datei des Herstellers abgerufen werden können.

Um die Router-Zugriffsfunktion zu implementieren, verwende ich die Dokumentation für Peplink Router API. Laut "Peplink Router API Documentation"[1] :

"The API is a set of HTTP endpoints. Each endpoint is an HTTP GET requests or POST requests with JSON arguments and JSON responses. The access port is same as that configured for Web Admin access. For security reason, however, the API should always be used under Secure HTTP (HTTPS) access" und mithilfe der Einstellungen der cURL-Bibliothek die Anfragen den Router verwalten.

## 3.2 Programmlogik zur Ansprache der API entwickeln

Um die Funktion "api()" zu entwickeln, erstelle ich die Datei api_peplink_connect.php (Abbildung 3). Diese Funktion sendet eine HTTP-Anfrage an eine bestimmte URL und gibt die Antwort zurück. Die "api()"-Funktion akzeptiert eine URL, eine Aktion und optional Parameter als Eingabe.

Um die HTTP-Anfrage auszuführen, verwende ich die cURL-Bibliothek und konfiguriere die entsprechenden cURL-Optionen. Die Funktion unterstützt sowohl GET- als auch POST-Anfragen und ermöglicht das Senden von Daten im JSON-Format.

Es wird auch überprüft, ob ein Cookie-Dateipfad vorhanden ist. Falls nicht, wird eine leere Datei erstellt. Die Antwort der Anfrage wird in den Sitzungsvariablen gespeichert, wenn die Anfrage auf spezifische API-Endpunkte gerichtet ist.

Um die cURL-Bibliothek verwenden zu können, muss die php.ini-Datei konfiguriert werden. Dazu suchen und kommentiere ich die Zeichenfolge ";extension=curl" in der php.ini-Datei aus. Anschließend speichere ich die Datei und starte den lokalen Apache-Server neu.

Für die korrekte Konfiguration der cURL-Optionen arbeite ich eng mit meinem Projektbetreuer Herrn Obed zusammen, der mir dabei hilft, die richtigen Einstellungen vorzunehmen.

Weitere Informationen zur Aktivierung von cURL in PHP findest du unter folgendem Link:
[https://www.geeksforgeeks.org/how-to-enable-curl-in-php/].

[1] https://knowhow.peplink.ninja/documentation/peplink-router-api-documentation/

```php
<?php
function api($url, $action, $parameters = NULL, $debug = FALSE)
{
    if (!file_exists($parameters['cookie'])) {
        $fh = fopen($parameters['cookie'], "w");
        fwrite($fh, "");
        fclose($fh);
    }

    $useragent  = isset($parameters['useragent']) ? $parameters['useragent'] : 'Mozilla/5.0
    (Windows NT 6.1; WOW64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2';

    if (!empty($parameters['data'])) {
        $post_isset = TRUE;
        $parameters['data'] = json_encode($parameters['data']);
    } else $post_isset = FALSE;

    $curl_options = array(
        CURLOPT_CONNECTTIMEOUT => (isset($parameters['timeout']) ? $parameters['timeout'] : 5),

        CURLOPT_COOKIEJAR      => $parameters['cookie'],
        CURLOPT_COOKIEFILE     => $parameters['cookie'],
        CURLOPT_RETURNTRANSFER => TRUE, //zurück bekommen Antwort
        CURLOPT_POST           => $post_isset, //POST:TRUE, GET:FALSE
        CURLOPT_URL            => $url . $action, //query data with url
        CURLOPT_USERAGENT      => $useragent,

        CURLOPT_SSL_VERIFYHOST => FALSE,
        CURLOPT_SSL_VERIFYPEER => FALSE,

        CURLOPT_ENCODING       => "",
        CURLOPT_FOLLOWLOCATION => TRUE,
        CURLOPT_HEADER         => FALSE,
        CURLINFO_HEADER_OUT    => TRUE,
        CURLOPT_HTTPHEADER     => array(
            'Content-Type: application/json'
        ),

        CURLOPT_AUTOREFERER    => TRUE,
        CURLOPT_HTTP_VERSION   => CURL_HTTP_VERSION_1_1,
    );
    # Actual Request
    $curl_handle =  curl_init();

    curl_setopt_array($curl_handle, $curl_options);
    if (isset($parameters['data'])) {
        ## DEBUG
        // print_r($parameters['data']);
        curl_setopt($curl_handle, CURLOPT_POSTFIELDS, $parameters['data']);
    }

    $result = curl_exec($curl_handle);

    require __DIR__ . '/../../config/config.php';
    if ($action == "/api/status.wan.connection") {
        file_put_contents($config['file'], $result, LOCK_EX);
        $_SESSION['wan_data']  = $result;
    }
    if ($action == "/api/status.pepvpn") {
        file_put_contents($config['file2'], $result, LOCK_EX);
        $_SESSION['vpn_data']  = $result;
    }

    curl_close($curl_handle);
    return $result;
```

*Abbildung 3: Funktion "api()".*

Dann in der Datei app.php erstelle ich Parameter für die Funktion "api()". Zuerst weise ich den Wert von $config['file_api'] der Variablen $file_api zu, um die Datei einzuschließen, deren Pfad in der Variablen $file_api gespeichert ist. Anschließend weise ich den Wert von $config['login_file'] der Variablen $login_file zu. Danach initialisiere ich das Array $data_auth mit dem Wert von $cookie und einem leeren Benutzernamen und Passwort. Diese Variable wird zur Authentifizierung des Benutzers und zum Zugriff auf die Funktion "api()" verwendet (Abbildung 4).

Um eine sichere Eingabe und Speicherung von Parametern und deren Weitergabe (API-Request) zu gewährleisten, entwickle ich eine Funktion, die mit einer Bedingung beginnt. Es wird überprüft, ob die Variablen "ip", "login" und "password" im POST-Array gesetzt sind. Wenn dies der Fall ist, wird die IP-Adresse des Geräts auf ihre Gültigkeit als IPv4-Adresse überprüft. Wenn die IP-Adresse gültig ist, wird sie in die Variable $ip geschrieben (Abbildung 5).

Dann wird die Funktion "edit_config_file()" dreimal aufgerufen, um die Werte der Variablen "ip", "login" und "password" in der Konfigurationsdatei zu aktualisieren. Dabei werden auch das Verschlüsselungsverfahren und der Schlüssel verwendet. Schließlich wird überprüft, ob bestimmte Dateien existieren, und falls ja, werden sie mit der Funktion "unlink()" gelöscht. Wenn die IP-Adresse ungültig ist, wird die Variable "$ip" auf den String "IP: Not valid!" gesetzt.

Um die Sicherheit der Benutzereingaben zu gewährleisten, verwende ich die Funktionen "filter_var()", "htmlentities()", "htmlspecialchars()" und "strip_tags()", um die Eingaben zu filtern und potenziell schädlichen Code zu entfernen, bevor sie in die Konfigurationsdatei geschrieben werden.

```php
<?php
require(__DIR__ . '/../../config/config.php');
require('encrypt.php');

$file_api = $config['file_api'];
include $file_api;
$cookie = $config['cookie'];
$login_file = $config['login_file'];

$data_auth = array(
    "cookie" => $cookie,

    "data" => [
        "username" => "",
        "password" => ""
    ]
);
$data_wan_status = array(
    "cookie" => $cookie,
    "method" => "GET"
);
$data_vpn_status = array(
    "cookie" => $cookie,
    "method" => "GET"
);
```

*Abbildung 4: Codeausschnitt von app.php (Teil 1)*

```php
function edit_config_file($for_edit, $my_own, $login_file, $method, $key_schluessel, $iv)/*($func1) */
{
    if (file_exists($login_file)) {
        $fopen = json_decode(file_get_contents($login_file), TRUE);
        if (array_key_exists($for_edit, $fopen)) {
            require_once('encrypt.php'); //required encryption file
            $fopen[$for_edit] = textencryption($my_own, $method, $key_schluessel, $iv); //encryption text
            file_put_contents($login_file, json_encode($fopen)); //src/api/login.json
        }
    } else {
        $array = array(
            'ip' => '',
            'login' => '',
            'password2' => ''
        );
        $json_data = json_encode($array);
        file_put_contents($login_file, $json_data);
        $fopen = json_decode(file_get_contents($login_file), TRUE);
        require_once('encrypt.php'); //required encryption file
        $fopen[$for_edit] = textencryption($my_own, $method, $key_schluessel, $iv); //encryption text
        file_put_contents($login_file, json_encode($fopen));
    }
}
###
if (isset($_POST["ip"]) && isset($_POST["login"]) && isset($_POST["password"])) {
    if (filter_var(htmlentities(strip_tags($_POST["ip"])), FILTER_VALIDATE_IP, FILTER_FLAG_IPV4)) {
        $ip = htmlspecialchars($_POST["ip"]);
        edit_config_file('ip', $ip, $login_file, $config["method"], $config['key_schluessel'], $config['iv']);

        $login = htmlentities(strip_tags($_POST["login"]));
        edit_config_file("login", $login, $login_file, $config["method"], $config['key_schluessel'], $config['iv']);

        $password = htmlentities(strip_tags($_POST["password"]));
        edit_config_file("password2", $password, $login_file, $config["method"], $config['key_schluessel'], $config['iv']);


    } else {
        $ip = "Not valid";

        $login = htmlentities(strip_tags($_POST["login"]));
        edit_config_file("login", $login, $login_file, $config["method"], $config['key_schluessel'], $config['iv']);

        $password = htmlentities(strip_tags($_POST["password"]));
        edit_config_file("password2", $password, $login_file, $config["method"], $config['key_schluessel'], $config['iv']);
    }
}
```

*Abbildung 5: Codeausschnitt von app.php (Teil 2)*

Die Funktion "edit_config_file()" (Abbildung 5) akzeptiert sechs Argumente: $for_edit, $my_own, $login_file, $method, $key_schluessel, und $iv. Sie wird verwendet, um bestimmte Parameter in einer JSON-Konfigurationsdatei zu ändern:

- Zunächst wird überprüft, ob die Datei $login_file existiert. Falls sie existiert, wird der Inhalt der Datei als JSON-Array dekodiert und in der Variable $fopen gespeichert;
- Dann wird überprüft, ob der Schlüssel $for_edit im Array $fopen vorhanden ist. Wenn ja, wird die Datei encrypt.php eingebunden und die Funktion textencryption mit den Argumenten $my_own, $method, $key_schluessel und $iv aufgerufen, um den Text $my_own zu verschlüsseln;
- Das verschlüsselte Ergebnis wird dann als Wert für den Schlüssel $for_edit im Array $fopen gespeichert;
- Zum Schluss wird das Array $fopen wieder in JSON-Format umgewandelt und in die Datei $login_file geschrieben;
- Falls $login_file nicht existiert, wird ein neues JSON-Array mit den Schlüsseln 'ip', 'login' und 'password2' erstellt und in die Datei $login_file geschrieben. Dann wird der Inhalt von $login_file wie oben beschrieben bearbeitet.

Nächste Codeblock (Abbildung 6) wird überprüft, ob die Anmeldedatei vorhanden ist. Wenn ja, werden die JSON-Daten aus der Datei entschlüsselt und der Benutzernamen, das Passwort und die IP-Adresse mithilfe der Funktion "textdecryption()" aus der Datei encrypt.php auf die entschlüsselten Werte gesetzt. Anschließend wird die Funktion "api()" aufgerufen, um zwei API-Aufrufe an die angegebene IP-Adresse durchzuführen und die WAN-Verbindungs- und VPN-Statusdaten abzurufen.

Als nächstes Punkt mache ich Bedienung die prüft ob die IP, das Login und das Passwort über die POST-Methode bereitgestellt werden, erstellt die Funktion eine Antwortnachricht mit den Werten und ruft die Funktion "api()" auf, um einen Login-API-Aufruf mit den bereitgestellten Daten durchzuführen.

```php
}
if (file_exists($login_file)) {
    $fopen = json_decode(file_get_contents($login_file), TRUE);
    $data_auth["data"]["username"] = textdecryption($fopen["login"], $config['method'], $config['key_schluessel'], $config['iv']);
    $data_auth["data"]["password"] = textdecryption($fopen["password2"], $config['method'], $config['key_schluessel'], $config['iv']);
    $url_1 = 'https://' . textdecryption($fopen["ip"], $config['method'], $config['key_schluessel'], $config['iv']);
}
if (isset($_POST["ip"]) && isset($_POST["login"]) && isset($_POST["password"])) {
    $_antwort_1 = " IP: $ip<br> Login: $login <br> Password: $password";
    $_antwort_2 = api($url_1, '/api/login', $data_auth, FALSE);
}
```

*Abbildung 6: Codeausschnitt von app.php (Teil 3)*

Nächste Codeausschnitt (Abbildung 7) reagiert auf POST-Anfragen, die von der "Support" Webseite gesendet werden. Je nach ausgewählter Aktion ruft das Skript verschiedene API-Endpunkte auf. Nachdem es sich bei dem entfernten Gerät authentifiziert hat, führt es die gewünschte Aktion aus. Die verfügbaren Aktionen umfassen das Erstellen oder Löschen von Clients, das Abrufen von Clientinformationen, das Setzen von Prioritäten für Clients sowie das Abrufen von Statusinformationen für das LAN, WAN und das Mobilfunknetz. Die Antwort wird im HTML-Format zurückgegeben, um sie auf der Benutzeroberfläche anzuzeigen. Wenn die Aktion "clear" ist, wird der Benutzer auf die Netzwerkseite weitergeleitet.

```php
if ($_POST) {

    api($url_1, '/api/login', $data_auth, FALSE);

    switch ($_POST['action']) {
        case 'create_client': ···
        case 'delete_client': ···
        case 'priority_client': ···
        case 'fetch_clients': ···
        case 'client_status': ···
        case 'lan_status': ···
        case 'wan_status':
            $_antwort_1 = '<h2>WAN STATUS</h2>' . api($url_1, '/api/status.wan.connection', $data_wan_status, false);
            $_antwort_2 = '<h2>VPN STATUS</h2>' . api($url_1, '/api/status.pepvpn', $data_vpn_status, FALSE);
            break;
        case 'cellular_status':
            $_antwort_1 =  '<h2>SCAN STATUS</h2>';
            $_antwort_2 =  api($url_1, '/api/cmd.carrier.scan?connId=4&reference=yes', $data_cellular_status, FALSE);
            break;
        case "clear":
            header('Location:network.php');
            die();
            break;
    }
}

## view_antwort
if (isset($_antwort_1)) {
    $viewantwort = '<pre>' . $_antwort_1 . '</pre>';
} else {
    $viewantwort = null;
}
if (isset($_antwort_2)) {
    $viewantwort = '<pre>' . $_antwort_1 . '</pre>' . '</br>' . '<pre>' . $_antwort_2 . '</pre>';
} else {
    $viewantwort = null;
}
```

*Abbildung 7: Codeausschnitt von app.php (Teil 4)*

Anschließend binde ich app.php mit config.php und encrypt.php Datei (Abbildung 4) in die enthält Konfigurationseinstellungen und Funktionen zur Verschlüsselung und Entschlüsselung entsprechend.

Für die Verschlüsselung und Entschlüsselung (Abbildung 8) meine IP-Adresse, Login und Password in Datei encrypt.php zunächst füge ich eine Konfigurationsdatei config.php ein, die die Verschlüsselungsmethode, den Schlüssel und den Initialisierungsvektor für die Ver- und Entschlüsselung des Textes enthält. Dann erstelle ich zwei Funktionen, "textencryption()" und "textdecryption()", die den übergebenen Text mit den in der Konfigurationsdatei angegebenen Werten ver- und entschlüsseln können. Die Verschlüsselung wird mit "openssl_encrypt()" und die Entschlüsselung mit "openssl_decrypt()" durchgeführt.

```php
<?php
require(__DIR__ . '/../../config/config.php');

$method = $config['method'];
$key_schluessel = $config['key_schluessel'];
$iv = $config['iv'];
function textencryption($text, $method, $key_schluessel, $iv)
{
    $contentsEncrypted = openssl_encrypt($text, $method, $key_schluessel,  0, $iv);
    return $contentsEncrypted;
};
function textdecryption($text, $method, $key_schluessel, $iv)
{
    $contentsDesrypted = openssl_decrypt($text, $method, $key_schluessel, 0, $iv);
    return  $contentsDesrypted;
};
```

*Abbildung 8: Ver- und Entschlüsselung*

## 3.3 Webanwendung entwickeln

Für die Eingabe von Einstellungen, die Ausgabe von Berichten für den internen Gebrauch und für Kunden, die Verfolgung von Daten in Echtzeit und die Umstellung auf die Webschnittstelle des Routers muss ich 4 Seiten erstellen:

- Die Hauptseite namens "Dashboard", die in der Datei index.php enthalten sein wird. Diese Seite wird eine Navigationsleiste, Informationen zum Router-Modell und seiner IP-Adresse, zwei Tabellen für die Anzeige von Daten zu WAN-Verbindungen und VPN-Verbindungen sowie Schaltflächen zur Änderung des Prioritätsstatus von WAN-Verbindungen enthalten;
- Die Seite "Support", die in der Datei network.php enthalten sein wird. Auf dieser Seite können Benutzer Router-Dateneingaben wie Aktivitätsauswahl und detaillierte Berichte anzeigen und ändern;
- Die Authentifizierungsseite, die in der Datei support_login_view.php enthalten sein wird. Diese Seite wird für die Anmeldung zur "Support"-Seite verwendet;
- Die "WebAdmin"-Seite, die in der Datei web_admin.php enthalten sein wird. Auf dieser Seite werden Links, Login- und Passwortinformationen für den Router angezeigt, und Benutzer können zur Webschnittstelle des Routers navigieren.

Außerdem werde ich separat für alle Seiten eine Navigationsleiste, die sich in der Datei navbar.php befinden wird, erstellen, die sich mit jeder Seite im <body>-Tag verbindet. Das Einrichtung des Darstellungsformats der Webseiten, der Stile, der Position auf der Seite , der Schriftgröße und der Farbe entsprechend den Anforderungen des Kunden wird in separaten Dateien, mit CSS-Code und mit dem Bootstrap-Clientframework beschrieben.

Die Hauptwebseite "Dashboard" (Abbildung 9) beginnt mit dem Einbinden der erforderlichen Dateien wie Konfigurationsdateien und Bibliotheken. Anschließend werden einige Funktionen wie "checkfiles()", "versionprueft()" und "api()" aufgerufen, um die Daten zu überprüfen und abzurufen, die auf der Seite angezeigt werden sollen. Danach wird die HTML-Seite erstellt und gestaltet, wobei Bootstrap und eigene CSS-Dateien verwendet werden. Die Seite enthält eine Navigation, die oben auf der Seite angezeigt wird, gefolgt von einem Container mit verschiedenen Informationen über das Gerät und seine Verbindungen. Diese Informationen werden in Form von Tabellen präsentiert, die mit JavaScript dynamisch aktualisiert werden können.

Die Funktion "checkfiles()" (Abbildung 10) überprüft, ob bestimmte Dateien auf dem Server existieren, die für den Login-Vorgang und die Cookie-Verwaltung benötigt werden. Wenn eine der Dateien nicht vorhanden ist, wird überprüft, ob im übergebenen Datenarray $data_auth ein Cookie und ein Benutzername gespeichert sind. Falls nicht, wird der Nutzer auf die Login-Seite weitergeleitet.

Die Funktion "versionprueft()" (Abbildung 11) überprüft die Firmware-Version des Geräts. Wenn die Firmware-Version im Vergleich zur in der Sitzung gespeicherten Firmware-Version neu ist, wird der API-Cookie gelöscht, der Benutzer wird ausgeloggt und die API-Authentifizierung wird erneut durchgeführt. Andernfalls wird nichts unternommen. Die Funktion später während der Testphase entwickelt wurde.

```php
<?php
session_start();
require __DIR__ . '/config/config.php';
require_once __DIR__ . "/src/libs/checkfiles.php";
require __DIR__ . '/src/libs/app.php';
require __DIR__ . "/src/libs/versioprueft.php";
require  __DIR__ . '/src/libs/snmp.php';
require_once __DIR__ . "/src/libs/snmpname.php";
checkfiles($data_auth, $config);
api($url_1, '/api/status.wan.connection', $data_wan_status, FALSE);
api($url_1, '/api/status.pepvpn', $data_vpn_status, FALSE);
versionprueft($deviceFirmwareVersion_neue, $url_1, $data_auth);
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Dashboard</title>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js" integrity="sha384-oBqDVmMz9ATKxIep9tiCxS/Z9fNfEXiDAYTujMAeBAsjFuCZSmKbSSUnQlmh/jp3" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js" integrity="sha384-IDwe1+LCz02ROU9k972gdyvl+AESN10+x7tBKgc9I5HFtuNz0wWnPclzo6p9vxnk" crossorigin="anonymous"></script>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous" />
    <meta charset="UTF-8" />
    <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="src/libs/script_dynamischetabelle.js"></script>
    <script src="src/libs/script_tabelle_vpn.js"></script>
    <style>
        <?php
        require("./public/css/index.css");  ?>
    </style>
</head>

<body>
    <?php require __DIR__ . "/src/inc/navbar.php"; ?>

    <div id="block-3" class="container-fluid">
        <!-- Geräte und IP-->
        <div style="margin:0px 5px 10px 0px">
            <table>
                <tr>
                    <th>  <?= $sysName;  ?>  </th>
                    <th>  IP : <?= $lanIp; ?>  </th>
                </tr>
            </table>
        </div>
        <!--WAN -->
        <div id="demo"></div>
        <!--VPN -->
        <div id="block-4" class="container-fluid"></div>
    </div>
</body>

</html>
```

*Abbildung 9: Hauptwebseite "Dashboard" index.php*

```php
function checkfiles($data_auth, $config)
{
    if (!file_exists($config['cookie']) || !file_exists($config['login_file'])) {

        if ($data_auth["cookie"] == "" || $data_auth['data']["username"] == "") {
            echo
            '<script>location.replace("/../peplink-connect3/public/support_login_view.php");</script>';
            exit;
        }
    }
}
```

*Abbildung 10: Funktion "checkfiles()".*

```php
<?php
function versionprueft($deviceFirmwareVersion_neue, $url_1, $data_auth)
{
    if ($_SESSION['deviceFirmwareVersion'] !== $deviceFirmwareVersion_neue[".1.3.6.1.4.1.23695.200.1.1.1.1.3.0"]) {
        unlink(__DIR__ . '/../api/data/api_cookie.txt');
        api($url_1, '/api/login', $data_auth, FALSE);
        $_SESSION['deviceFirmwareVersion'] = $deviceFirmwareVersion_neue[".1.3.6.1.4.1.23695.200.1.1.1.1.3.0"];
        session_write_close();
    } else {
        // echo 'ALTEVersion!!!' . '</br>';
    }
}
```

*Abbildung 11: Funktion "versionprueft()".*

Auf der Seite "Support" (Abbildung 12) baue ich eine visuelle Darstellung erstellt, um Einstellungen wie "IP", "Login" und "Password" anzugeben und eine Aktion auszuwählen. Darüber hinaus wird das Ergebnis der Einstellungen oder Berichte angezeigt. Dazu habe ich die Seite in drei Teile unterteilt: eine Navigationsleiste oben auf der Seite, ein Bereich zur Dateneingabe und Aktivitätsauswahl links und eine Anzeige für das Ergebnis oder den Bericht rechts. Die Navigationsleiste habe ich in einer separaten Datei definiert. Zwei Formulare werden hinzugefügt: eines für die Verbindung mit dem Webadministrator und eines für den Zugriff auf Peplink Connect, ein Netzwerkgerät zur Verwaltung von WAN-, LAN- und Mobilfunkverbindungen. Dann habe ich Bootstrap, eine CSS-Bibliothek für die Gestaltung von Webseiten, und Popper.js, eine JavaScript-Bibliothek für Popups und Menüs, importiert. Die Formulardaten werden per POST an ein anderes Skript auf dem Server gesendet, das eine Verbindung herstellt und die gewünschte Aktion ausführt. Die Variablendeklarationen und Importe für die PHP-Bibliotheken und CSS-Dateien werden im <head>-Bereich der HTML-Seite durchgeführt. Der Hauptteil der HTML-Seite wird im <body>-Tag definiert und enthält die Formulare für den Benutzerzugriff auf die Netzwerkgeräte sowie eine Anzeige für die Antwort des Servers.

```php
<body>
    <?php require(__DIR__ . "/../inc/navbar.php"); ?>
    <div id="block-1" class="container-fluid">
        <div>
            <h1>WebAdmin</h1>
            <form name="action" method="POST">
                <p>IP : <input type="text" name="ip" autocomplete="on" required /></p>
                <p>Login : <input type="text" name="login" autocomplete="off" required /></p>
                <p>Password : <input type="password" name="password" autocomplete="off" required /></p>
                <p><input type="submit" name="action" value="Verbinden" /></p>
            </form>
        </div>
        <div>
            <h1>Peplink Connect</h1>
            <form method="POST">
                <select name="action" id="">
                    <option value="create_client">Create client</option>
                    <option value="delete_client">DELETE Client</option>
                    <option value="priority_client">PRIORITY Client</option>
                    <option value="fetch_clients">Fetch clients</option>
                    <option value="client_status">Status of clients</option>
                    <option value="lan_status">Status LAN</option>
                    <option value="wan_status">Status WAN</option>
                    <option value="cellular_status">Status Cellular</option>
                </select>
                <input type="submit" value="Execute">
                <button type="submit" name="action" value="clear">Clear</button>
            </form>
        </div>
    </div>
    <div id="block-2"> <?= $viewantwort; ?></div>
</body>

</html>
```

*Abbildung 12: Codeausschnitt Webseite "Support"*

```css
table,
table td,
th,
caption {
  border: 1px solid black;
  border-collapse: separate;
  text-align: center;
  grid-template-columns: auto;
  grid-template-rows: auto;
  font-family: 'Montserrat', sans-serif;
  font-size: 15;
  white-space: nowrap;
}
.block-3.td,
th {
  height: 60px;
}
#table-1 {
  margin: auto;
}
.container-fluid {
  padding-left: 1px;
  padding-right: 1px;
}

#block-3 {
  top: 56px;
  position: fixed;
  padding-top: 1%;
  padding-left: 1%;
  padding-right: 1%;
  height: 100%;
  overflow: auto;
  overflow-y: scroll;
}

#block-4 {
  position: absolute;
  padding-top: 1%;
  height: 100%;
  overflow: auto;
  overflow-y: scroll;
}
.navbar-nav li a,
.nav-link,
.nav-link.active {
  color: #efefef;
  font-family: 'Montserrat', sans-serif;
  font-weight: 900;
}

.navbar {
  position: fixed;
  left: 0;
  right: 0;
  min-width: 800px;
  padding-left: 1px;
  padding-right: 1px;
  background-color: #30305b;
}
```

*Abbildung 13: CSS Einrichtung für index.php (Codeausschnitt)*

```css
h1,
p {
  font-size: 20px;
  font-family: 'Montserrat', sans-serif;
  font-weight: 900;
  color: #30305b;
  text-decoration: underline;
  margin-top: 0;
  margin-bottom: 1rem;
}

#block-1 {
  top: 56px;
  padding-top: 1%;
  position: fixed;
  left: 0;
  /* top: 50px; */
  width: 50%;
  height: 80%;
  /* background: lightcyan; */
  color: black;
  overflow: auto;
  border-right: 2px solid #30305b;

  overflow-y: scroll;
}
#block-2 {
  top: 56px;
  padding-top: 1%;
  position: fixed;
  right: 0;
  width: 50%;
  height: 100%;
  color: black;
  overflow-y: scroll;
}
.navbar-nav li a,
.nav-link,
.nav-link.active {
  color: #efefef;
  font-family: 'Montserrat', sans-serif;
  font-weight: 900;
}

.navbar {
  position: fixed;
  left: 0;
  right: 0;
  min-width: 800px;
  padding-left: 1px;
  padding-right: 1px;
  background-color: #30305b;
}
```

*Abbildung 14: CSS Einrichtung für Webseite "Support" (Codeausschnitt)*

Jetzt sieht das Aussehen der Webseite "Support" so aus:


*Abbildung 15: Webseite "Support"*

Danach erstelle ich die Login-Seite "Authentication" (Abbildung 16), die sich in der Datei "support_login_view.php" befindet. Die Seite hat ein Bootstrap-Design und verwendet PHP, um die Benutzerdaten zu validieren. In der <head>-Sektion lade ich Skripte und Stylesheets, einschließlich Popper.js und Bootstrap. Im <body>-Tag erstelle ich eine Navigationsleiste und einen Container mit der Klasse "container-fluid". Innerhalb des Containers befindet sich ein Formular, das einen Benutzernamen und ein Passwort erfordert. Wenn der Benutzer auf den Submit-Button klickt, wird die Funktion "checkpass()" aus der Datei "checkpass_support.php" aufgerufen (Abbildung 17). Die Funktion "checkpass()" überprüft, ob das eingegebene Passwort und der eingegebene Benutzername korrekt sind, indem sie die Hashes der Passwörter und Benutzernamen vergleicht. Wenn die Überprüfung erfolgreich ist, wird der Benutzer zur Netzwerkseite weitergeleitet. Andernfalls wird der Login-Status als "access_denied" zurückgegeben.

```
<body>
    <?php require __DIR__ . "/../src/inc/navbar.php"; ?>

    <div id="block-5" class="container-fluid">
        <h1>Login Seite</h1>
        <p>
        <form method="post">
            <table class="login">
                <tr>
                    <td>Login</td>
                    <td><input type="text" name="login" autocomplete="off" required></td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td><input type="password" name="password" autocomplete="off" required></td>
                </tr>
                <th colspan="2">
                    <input type="submit" value="Einloggen" name="btn">
                </th>
            </table>
        </form>
        </p>
    </div>
</body>

</html>
<?php
require_once __DIR__ . "/checkpass_support.php";
checkpass();
?>
```

*Abbildung 16: Login-Seite "Authentication"*

```
<?php
function checkpass()
{
    if (isset($_POST['login']) && isset($_POST['password'])) {
        $login = $_POST['login'];
        $password = $_POST['password'];
        $paswordhash = password_hash("admin", PASSWORD_DEFAULT);
        $loginhash = password_hash("admin", PASSWORD_DEFAULT);
        if (password_verify($password, $paswordhash) && password_verify($login, $loginhash)) {

            echo '<script>location.replace("/../peplink-connect3/src/api/network.php");</script>';
            exit;
        } else {
            $data["login_status"] = "access_denied";
        }
    } else {
        $data["login_status"] = "";
    }
}
```

*Abbildung 17: Funktion "checkpass()".*

Nächste Webseite "WebAdmin" (Abbildung 18) habe ich mit PHP, HTML und JavaScript geschrieben. Zu Beginn habe ich einige externe Bibliotheken und Ressourcen, wie Bootstrap und PopperJS, in die Seite eingebunden. Danach habe ich die Zeichenkodierung, die Browserkompatibilität und die Ansicht der Seite festgelegt.

Der Inhalt der Variablen wird vom Router abgerufen und enthält den Benutzernamen und das Passwort für das Webinterface des Routers. Darunter befinden sich zwei Schaltflächen, um den Benutzernamen und das Passwort in die Zwischenablage zu kopieren, damit der Benutzer sie dann im Webinterface des Routers einfügen kann. Darüber hinaus wird ein Hyperlink zur Verfügung gestellt, der den Benutzer direkt zum Webinterface des Routers führt.

Die JavaScript-Funktionen habe ich hergestellt, um den Benutzernamen und das Passwort in die Zwischenablage zu kopieren. In der Funktion wird die ausgewählte ID des Benutzernamens oder des Passworts verwendet, um das entsprechende Textfeld zu identifizieren und seinen Wert zu kopieren. Der kopierte Text wird dann in der Zwischenablage des Benutzers gespeichert. Schließlich wird eine Bestätigungsmeldung angezeigt, um den Benutzer darüber zu informieren, dass der Text erfolgreich in die Zwischenablage kopiert wurde.

```
<body>
    <?php
    require("navbar.php");
    ?>
    <div id="block-6" class="container-fluid">
        <h2>Zum Router Webadmin</h2>
        <p>
            Mit dem nachfolgenden Link gelangen Sie direkt auf das Administrationsinterface des Routers.
            Zur Anmeldung benötigen Sie einen Benutzernamen und ein Passwort.
            Sie können die Daten einzeln in die Zwischenablage kopieren und im Webinterface einfügen
        </p>
        <p>Login : <?php echo $data_auth["data"]["username"]; ?>
            <button onclick="copyToClipboard('username')">
                <img src="./public/images/copy_83918.png" alt="copy_83918">
            </button>
        </p>
        <input style="display: none" type="text" value="<?php echo $data_auth["data"]["username"]; ?>" id="username" readonly>
        <div class="copy1">
            <p>Password : <?php echo   $data_auth["data"]["password"]; ?>
                <button onclick="copyToClipboard('password')">
                    <img src="./public/images/copy_83918.png" alt="copy_83918">
                </button>
            </p>
            <input style="display: none" value="<?php echo $data_auth["data"]["password"]; ?>" id="password" readonly>
        </div>
        <p><a href=<?php echo $url_1 ?> target="_blank" style="text-decoration: underline">Zum Webinterface hier klicken : <?php echo $url_1 ?></a></p>
    </div>
    <script>
        function copyToClipboard(name) {
            if (name == "username") {
                var copyText = $("#username").val();
            } else if (name == "password") {
                var copyText = $("#password").val();
            }
            console.log(copyText)
            navigator.clipboard.writeText(copyText);
            alertTimeout("Copied the text: " + copyText, 1000);
            function alertTimeout(mymsg, mymsecs) {
                var myelement = document.createElement("div");
                myelement.setAttribute("style", "background-color: #c8cbe3;color:#30305b; width: 450px;height: 100px;position: absolute;top:0;bottom:0;le
                border: 4px solid black;font-family:arial;font-size:25px;font-weight:bold;display: flex; align-items: center; justify-content: center; te
                myelement.innerHTML = mymsg;
                setTimeout(function() {
                    myelement.parentNode.removeChild(myelement);
                }, mymsecs);
                document.body.appendChild(myelement);
            }
        }
    </script>
</body>
</html>
```

*Abbildung 18: Webseite "WebAdmin"*

Im Navigationsleiste (Abbildung 19) der eine Navigation auf einer Webseite darstellt, verwende ich das Bootstrap-Framework, um das Design und die Funktionalität der Navigation zu gestalten.

Die Navigation wird durch das <nav>-Tag definiert, das eine Bootstrap-Klasse navbar enthält, die ein responsives Verhalten auf kleineren Bildschirmen ermöglicht. Das navbar-expand-sm-Attribut gibt an, dass die Navigation auf kleineren Bildschirmen automatisch erweitert wird. Innerhalb des nav-Tags befindet sich ein Container, der durch das -Tag definiert wird und eine Bootstrap-Klasse container-fluid enthält. Dies sorgt dafür, dass der Container den verfügbaren Platz auf der Seite ausfüllt. Die Navigation enthält ein Logo, das durch das <a>-Tag mit einer Bootstrap-Klasse navbar-brand und einem Link zur Homepage der Webseite eingebettet ist. Das Logo selbst ist ein Bild, dass durch das <img>-Tag eingebunden wird.

Darunter befindet sich ein Button, der durch das <button>-Tag definiert wird und eine Bootstrap-Klasse navbar-toggler enthält. Dieser Button wird auf kleineren Bildschirmen angezeigt, wenn die Navigation erweitert wird. Wenn der Button geklickt wird, wird der Inhalt der Navigation durch das collapse-Attribut und die Bootstrap-Klasse navbar-collapse ausgeblendet oder eingeblendet.

Innerhalb des -Tags befindet sich eine <ul>-Liste mit drei <li>-Elementen, die jeweils einen Link zu verschiedenen Seiten der Webseite enthalten. Die Bootstrap-Klasse "navbar-nav" sorgt für das richtige Styling der Liste und ihrer Elemente.


*Abbildung 19: Navigationsleiste*

## 3.4 Verarbeitung die Darstellung der eingegebenen Informationen.

### Implementierung Funktionen für Erste Tabelle

In der Datei snmpname.php (Abbildung 20) implementiere ich zwei Funktionen, um Informationen über das Router Modell und seine IP-Adresse zu verarbeiten. Diese Informationen werden anschließend an die Datei index.php weitergegeben. Zunächst definiere ich eine Variable namens $host, die mithilfe der Funktion textdecryption() entschlüsselt wird. Diese Funktion erfordert die Parameter $fopen["ip"], $method, $key_schluessel und $iv. Die Variable $community wird auf den Wert 'public' gesetzt. Um die Art und Weise zu ändern, wie die SNMP-Daten abgerufen und dargestellt werden, rufe ich die Funktionen "snmp_set_oid_output_format()" und "snmp_set_valueretrieval()" auf.

Mithilfe der Funktion "implode()" erhalte ich den Wert bestimmter SNMP-Objekte, indem ich den Host, die Community und das OID an die Funktion "snmp2_real_walk()" übergebe. Diese Funktion gibt alle Instanzen eines SNMP-Objekts und deren Werte in einem bestimmten Bereich zurück, der durch das OID definiert ist. Der Wert des Objekts mit dem OID ".1.3.6.1.2.1.1.5.0" (SysName) wird in der Variable $sysName gespeichert, und der Wert des Objekts mit dem OID ".1.3.6.1.4.1.23695.200.1.3.1.1.1.0" (LanIp) wird in der Variable $lanIp gespeichert.

```php
<?php
$host = textdecryption($fopen["ip"], $method, $key_schluessel, $iv);

$community = 'public';
snmp_set_oid_output_format(SNMP_OID_OUTPUT_NUMERIC); //
snmp_set_valueretrieval(SNMP_VALUE_PLAIN);
$sysName = implode(snmp2_real_walk("$host", "$community", ".1.3.6.1.2.1.1.5.0")); // sysName
$lanIp = implode(snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.3.1.1.1.0")); // lanIp
```

*Abbildung 20: snmpname.php*

### Implementierung Funktionen für Zweite Tabelle.

Um die Daten, die von der Router-API zurückgegeben werden, besser zu lesen und zu verstehen, und um eine dynamische Tabelle zu erstellen, habe ich den Code in zwei Funktionen aufgeteilt – "dynamischtabelle()" und "tabelleBauen()". Zunächst wird die Konfigurationsdatei "config.php" eingebunden, und dann rufe ich die Funktion "api()" zweimal auf, um den aktuellen Status der WAN-Verbindungen und die Daten der VPN-Tunnel vom Router abzurufen.

### Implementierung Funktion "dynamishetabelle()".

Die Funktion "dynamischtabelle()" (Abbildung 21) akzeptiert zwei Argumente: $_count als Ganzzahl und $json_decoded als Array, das aus einer JSON-Zeichenfolge dekodiert wurde. Innerhalb der Funktion werden zwei API-Aufrufe durchgeführt, um Daten abzurufen und in Variablen zu speichern. Die Funktion durchläuft das Array "json_decoded" und erstellt für jedes Element im Array eine HTML-Tabellenzeile. Wenn das Element einen "message"-Schlüssel hat, der nicht gleich "Disabled (Activation Required)" ist, wird eine weitere Funktion namens "tabelleBauen()" aufgerufen, um den Inhalt der Tabellenzelle zu erstellen. Wenn das Element auch einen "priority"-Schlüssel hat, wird eine Optionsfeldgruppe erstellt, mit der der Benutzer die Priorität des Elements festlegen kann. Ein JavaScript-Skript ist enthalten, das erkennt, wenn eine der Optionsfelder angeklickt wird, und die Priorität für die entsprechenden Daten aktualisiert.

```php
<?php
require __DIR__ . "/../../config/config.php";
api($url_1, '/api/status.wan.connection', $data_wan_status, FALSE);
api($url_1, '/api/status.pepvpn', $data_vpn_status, FALSE);

function dynamishetabelle($_count, $json_decoded)
{
    $colum_numer = 0;
    $btn_name = 1;
    $btn_radio1 = 1;
    $btn_radio2 = 2;
    for ($i = 0; $i < $_count; $i++) { ?>
        <tr>
            <?php if (array_key_exists('message', $json_decoded['response'][$i + 1])) {
                if ($json_decoded['response'][$i + 1]['message'] != "Disabled (Activation Required)") { ?>
                    <th><?= $colum_numer += 1; ?></th>
                    <?php tabelleBauen($json_decoded, $i); ?>
                    <td>
                        <?php ##bauen radio button
                        if (!empty($json_decoded['response'][$i + 1]['priority'])) { ?>
                            <div class="btn-group" role="group" aria-label="Basic radio toggle button group">
                                <form action="priority.php" method="POST">
                                    <?php ### Visible nur eine Knopf
                                    if ($json_decoded['response'][$i + 1]['priority'] == 1) { ?>
                                        <input type="radio" class="btn-check" name="<?= $btn_name ?>" id="<?= $btn_radio1 ?>" value="1" autocomplete="off" data-id="<?= $i + 1; ?>" style=" display: none;" checked />
                                        <label id="<?= $btn_radio1 ?>" class="btn btn-primary" style="display: none;" for="<?= $btn_radio1  ?>">Set Priority</br>1</label>

                                        <input type="radio" class="btn-check" name="<?= $btn_name ?>" id="<?= $btn_radio2  ?>" value="2" autocomplete="off" data-id="<?= $i + 1; ?>" />
                                        <label id=" <?= $btn_radio2  ?>" class="btn btn-outline-primary" for="<?= $btn_radio2  ?>">Set Priority</br>2</label>
                                    <?php
                                    } elseif ($json_decoded['response'][$i + 1]['priority'] == 2) { ?>
                                        <input type="radio" class="btn-check" name="<?= $btn_name ?>" id="<?= $btn_radio1 ?>" value="1" autocomplete="off" data-id="<?= $i + 1; ?>" />
                                        <label id=" <?= $btn_radio1 ?>" class="btn btn-primary" for="<?= $btn_radio1  ?>">Set Priority</br>1</label>

                                        <input type="radio" class="btn-check" name="<?= $btn_name ?>" id="<?= $btn_radio2  ?>" value="2" autocomplete="off" style="display: none;" checked />
                                        <label id="<?= $btn_radio2  ?>" class="btn btn-outline-primary" style="display: none;" for="<?= $btn_radio2  ?>" data-id="<?= $i + 1; ?>">Set Priority</br>2</label>
                                    <?php }
                                    ?>
                                    <script src="/peplink-connect3/src/libs/script_priority.js"></script>
                                </form>
                            </div>
                        <?php $btn_name += 1;
                            $btn_radio1 += 2;
                            $btn_radio2 += 2;
                        } ?>
                    </td><?php
                        }
                    } else { ?>
                <th><?php echo $colum_numer += 1; ?></th>
                <?php tabelleBauen($json_decoded, $i); ?>
                <td></td> <?php
                        } ?>
        </tr><?php
            }
        }; ?>
```

*Abbildung 21: Funktion "dynamishetabelle()".*

## Implementierung Funktion "tabelleBauen()".

Die Funktion "tabelleBauen()" (Abbildung 22) erstellt eine HTML-Tabelle basierend auf den JSON-Daten. Der Funktionsparameter $json_decoded ist das dekodierte JSON-Objekt, und $i wird als Indexwert verwendet, um auf die entsprechenden Daten im JSON-Objekt zuzugreifen.

Die Funktion gibt HTML-Code aus, um eine Tabelle mit verschiedenen Spalten zu erstellen, darunter WANNAME, STATUS, STATUS Connect, TYPE, Signal Stark und STATUS Priority. Jede Zeile der Tabelle enthält Daten für einen WAN-Anschluss. Bestimmte Spalten können auch Bilder enthalten, z. B. die Signalstärkeanzeige.

In der Spalte STATUS Connect wird basierend auf dem Verbindungsstatus der Text oder ein Bild angezeigt, um anzuzeigen, ob die Verbindung aktiviert oder deaktiviert ist und ob eine Verbindung hergestellt wird oder nicht. Wenn eine Verbindung hergestellt wird, wird auch das entsprechende Bild angezeigt, das den Verbindungsstatus symbolisiert.

Die Spalte Signal Stark zeigt ein Bild der Signalstärke an, das auf der Basis der Signalstärke des Cellular-Modems angezeigt wird. Wenn das Signal stark ist, werden beispielsweise fünf Balken angezeigt. Wenn das Signal schwach ist, werden weniger Balken angezeigt, und bei sehr schwachem Signal wird nur ein Balken angezeigt.

In der Spalte STATUS Priority wird die Priorität des WAN-Anschlusses angezeigt. Wenn der WAN-Anschluss eine höhere Priorität hat, wird der entsprechende Text angezeigt. Die letzte Spalte bietet die Möglichkeit, die Priorität des WAN-Anschlusses zu verwalten.

Die Funktion ermöglicht somit die Erstellung einer dynamischen Tabelle zur Darstellung von WAN-Anschlussinformationen mit verschiedenen visuellen Elementen.

```php
<?php
function tabelleBauen($json_decoded, $i)
{
?>
    <!--WANNAME -->
    <td><?= $json_decoded['response'][$i + 1]['name'] ?></td>
    <!-- STATUS-->
    <td><?= ($json_decoded['response'][$i + 1]['enable'] === true) ?  'enable' : 'disable';  ?>
    </td>
    <!-- STATUS Connect-->
    <td class="ausnahme"><?php if (!empty($json_decoded['response'][$i + 1]['message'])) {
                            if ($json_decoded['response'][$i + 1]['statusLed'] === "green") {
                            ?>
                <img src="./public/images/led-green.png" alt="led-green">
            <?php
                                echo $json_decoded['response'][$i + 1]['message'];
                            } elseif ($json_decoded['response'][$i + 1]['statusLed'] === "flash") { ?>
                <img src="./public/images/led-flash.gif" alt="led-flash">
            <?= $json_decoded['response'][$i + 1]['message'];
                            } elseif ($json_decoded['response'][$i + 1]['statusLed'] === "yellow") { ?>
                <img src="./public/images/led-orange.png" alt="led-yellow">
            <?= $json_decoded['response'][$i + 1]['message'];
                            } elseif ($json_decoded['response'][$i + 1]['message'] === "No SIM Card Detected" || $json_decoded['response'][$i + 1]['message'] === "No Device Detected") { ?>
                <img src="./public/images/led-red.png" alt="led-red">
        <?= $json_decoded['response'][$i + 1]['message'];
                            } else {
                                echo $json_decoded['response'][$i + 1]['message'];
                            }
                        } ?>
    </td>
    <!-- TYPE-->
    <td><?php if (!empty($json_decoded['response'][$i + 1]['type'])) echo $json_decoded['response'][$i + 1]['type']  ?></td>
    <!-- Signal Stark-->
    <td><?php if (!empty($json_decoded['response'][$i + 1]['type'])) {
            if ($json_decoded['response'][$i + 1]['type'] == "cellular" & $json_decoded['response'][$i + 1]['statusLed'] === "green") {
                $_arraynum = 0;
                for ($l = 0; $l < 1; $l++) {
                    $_min = 0;
                    $json_decoded['response'][$i + 1]['cellular']['rat'][0]['band'][$_arraynum]['signal']['rssi'] < $_min ?
                        $_min = $json_decoded['response'][$i + 1]['cellular']['rat'][0]['band'][$_arraynum]['signal']['rssi'] : $_min;
                    $_arraynum = $_arraynum + 1;
                }
                if ($_min >= -59) { ?>
                    <img src="./public/images/signal5-5.gif" alt="signal5">
                <?php
                    echo $_min . " " . "dB";
                } elseif ($_min >= -69) { ?>
                    <img src="./public/images/signal5-4.gif" alt="signal4">
                <?php
                    echo $_min . " " . "dB";
                } elseif ($_min >= -79) { ?>
                    <img src="./public/images/signal5-3.gif" alt="signal3">
                <?php
                    echo $_min . " " . "dB";
                } elseif ($_min >= -89) { ?>
                    <img src="./public/images/signal5-2.gif" alt="signal2">
                <?php
                    echo $_min . " " . "dB";
                } elseif ($_min >= -105) { ?>
                    <img src="./public/images/signal5-1.gif" alt="signal1">
                <?php
                    echo $_min . " " . "dB";
                } elseif ($_min < -105) { ?>
                    <img src="./public/images/signal5-0.gif" alt="signal0">
                <?php
                    echo $_min . " " . "dB";
                }
            } elseif ($json_decoded['response'][$i + 1]['type'] == "cellular" & $json_decoded['response'][$i + 1]['statusLed'] === "flash") {
                if ($json_decoded['response'][$i + 1]['message'] === "Obtaining IP Address...") { ?>

                    <img src="./public/images/signal5-0.gif" alt="signal0">
        <?php
                };
            }
        }
        ?>
    </td>
    <!-- STATUS Priority-->
    <td><?php if (!empty($json_decoded['response'][$i + 1]['priority'])) echo $json_decoded['response'][$i + 1]['priority']  ?></td>
```

*Abbildung 22: Funktion "tabellebauen()".*

## Implementierung Funktionen für Dritte Tabelle

In dieser Tabelle werden Informationen über den Zustand der VPNTunnel angezeigt, die aus mehreren Quellen stammen: über die API des Routers mit der Funktion "vpnwanverbindung()" und über SNMP-Requests mit der Funktion "getsnmpdaten()". Die Daten werden in den Funktionen "vpnwanverbindung()" und "getsnmpdaten()" verarbeitet. Die verarbeiteten Daten werden dann in der Datei view_tabelle_vpn.php zusammengeführt und in einer gemeinsamen Tabelle angezeigt.

Die Funktion durchläuft dann eine Schleife über die WAN-Verbindungen und überprüft, ob eine Verbindung aktiv ist und ob deren Status "Connected" oder "Standby" ist. Wenn das enable-Attribut von $json_decoded['response'][$i + 1] auf true gesetzt ist und die message entweder den Text "Connected" enthält oder "Standby" ist, wird $count_active_verbindung um eins erhöht.

Anschließend wird eine HTML-Tabelle generiert, die den Namen der VPN-Verbindung und das Status-LED-Symbol enthält (grün oder orange, abhängig von $json_decoded['response'][$i + 1]['statusLed']).

Die Daten werden somit entsprechend verarbeitet und in der Tabelle dargestellt, um den Zustand der VPNTunnel anzuzeigen.

```php
<?php
api($url_1, '/api/login', $data_auth, FALSE);
api($url_1, '/api/status.wan.connection', $data_wan_status, FALSE);
api($url_1, '/api/status.pepvpn', $data_vpn_status, FALSE);



function vpnwanverbindung($_count, $json_decoded/*, $count_active_verbindung*/)
{
    $count_active_verbindung = 0;
    for ($i = 0; $i < $_count; $i++) {
        if ($json_decoded['response'][$i + 1]['enable'] === true and (str_contains($json_decoded['response'][$i + 1]['message'],
                                                                                   "Connected") || $json_decoded['response'][$i + 1]['message'] === "Standby")) {
            $count_active_verbindung += 1;
?> <th>
                <?php if ($json_decoded['response'][$i + 1]['statusLed'] === "green") {
                ?>
                    <img src="./public/images/led-green.png" alt="led-green">
                <?php
                } elseif ($json_decoded['response'][$i + 1]['statusLed'] === "yellow") { ?>
                    <img src="./public/images/led-orange.png" alt="led-yellow">
                <?php
                }
                ?>
                <?= $json_decoded['response'][$i + 1]['name'];
                ?>
            </th>
    <?php
        }
    }
    ?>
    <th>Ingesamt</th> <?php    };
```

*Abbildung 23: Funktion "vpnwanverbindung()".*

Die Funktion "getsnmpdaten()" (siehe Anhang, Abbildung 24) erhält mehrere Parameter und liefert HTML-Code zurück. Dieser Code dient zur Generierung einer Tabelle mit VPN-Verbindungsinformationen und den dazugehörigen SNMP-Daten. Die Funktion beginnt mit einer Schleife, die für jede VPN-Verbindung eine neue Tabellenzeile erstellt und Informationen wie Verbindungsnummer, Name und Status ausgibt. Anschließend folgt eine weitere Schleife, die die SNMP-Daten für jede Netzwerkverbindung durchläuft. Wenn die Verbindung aktiv und verbunden ist, werden Werte wie Latenz, Anzahl der verworfenen Pakete sowie die Übertragungsraten (Tx und Rx) in kbps oder Mbps ausgegeben.

Danach erfolgt die Ausgabe der gesamten SNMP-Daten für die jeweilige VPN-Verbindung, einschließlich der insgesamt übertragenen Datenmengen für Tx und Rx. Letztendlich liefert die Funktion den generierten HTML-Code für die Tabelle zurück.

In der Datei view_tabelle_vpn.php wird eine HTML-Tabelle mit VPN-Verbindungen und deren Status generiert. Die Tabelle besteht aus Spalten für die Nummer der Verbindung, den Namen der VPNs und den Status der VPNs. Der Inhalt der Spalte "Status VPNs" wird von der Funktion "vpnwanverbindung()" generiert. Diese Funktion akzeptiert zwei Argumente, $_count und $json_decoded, und erstellt den entsprechenden Inhalt basierend auf diesen Daten. Die anderen Spalten der Tabelle werden von der Funktion "getsnmpdaten()" generiert. Diese Funktion akzeptiert mehrere Argumente, einschließlich $_count_vpn, $_count, $json_decoded, $json_decoded_vpn, $latency, $drop_paket_different, $tx_different, $rx_different, $vpn_Tx_ingesamt, und $vpn_Rx_ingesamt. Sie verwendet diese Daten, um den Inhalt der jeweiligen Spalten zu generieren und zu aktualisieren. Durch die Kombination dieser Funktionen wird die HTML-Tabelle mit den VPN-Verbindungen und den entsprechenden Informationen in der Datei view_tabelle_vpn.php erstellt.

In der Datei snmp.php (Abbildung 25 und 26 im Anhang) wird das Simple Network Management Protocol (SNMP) verwendet, um Netzwerkstatistiken von einem Netzwerkgerät abzurufen. Durch die Verwendung von SNMP wird eine Verbindung zum Gerät hergestellt, und es werden verschiedene Management-Information-Base (MIB)-Variablen abgerufen. Diese MIB-Variablen sind hierarchisch organisiert und enthalten Informationen über verschiedene Aspekte des Netzwerkgeräts, wie z.B. Schnittstellen, Routing-Tabellen und virtuelle private Netzwerke (VPNs). In diesem speziellen Code werden Statistiken für VPN-Verbindungen abgerufen. Der Code verwendet SNMP-Abfragen, um Informationen wie den Verbindungsstatus, die Transferraten (Tx und Rx), den Paketverlust und die Latenzzeit abzurufen. Die Funktion "snmp2_real_walk()" wird verwendet, um eine SNMP-Abfrage für eine bestimmte OID (Object Identifier) durchzuführen. Diese Funktion liefert ein assoziatives Array zurück, in dem die OID als Schlüssel und der abgerufene Wert als Wert gespeichert sind. Einige der abgerufenen OIDs sind:

".1.3.6.1.4.1.23695.200.1.1.1.1.3.0": Firmware-Version des Geräts
".1.3.6.1.4.1.23695.200.1.10.1.1.2.1.2": VPN-Verbindungsprofilname
".1.3.6.1.4.1.23695.200.1.10.1.1.2.1.3": Verbindungsstatus für jede VPN-Verbindung
".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.1": ID jedes WAN-Interfaces
".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.2": Name jedes WAN-Interfaces
".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.3": Übertragungsrate (Tx) jedes WAN-Interfaces
".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.4": Übertragungsrate (Rx) jedes WAN-Interfaces
".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.5": Anzahl der verworfenen Pakete jedes WAN-Interfaces
".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.6": Latenzzeit jedes WAN-Interfaces

Die abgerufenen Daten werden zuerst mithilfe der Funktionen "array_combine()", "array_diff()", "verkurzoid()", "array_intersect_key()", "vpn_werte_ingesamt()" und dann in verschiedenen Variablen gespeichert, wie z.B. $latency, $drop_paket_different, $tx_different, $rx_different, $vpn_Tx_ingesamt, $vpn_Rx_ingesamt und $vpn_Tx_ingesamt. Diese Variablen werden dann zur weiteren Verarbeitung in der Funktion "getsnmpdaten()" verwendet.

Um die Transferraten (Tx und Rx) und den Paketverlust zu verfolgen und die Differenz zwischen neuen und zuvor gespeicherten Werten zu berechnen, wird eine Sitzung gestartet. Eine Sitzung ermöglicht es, Variablen über verschiedene Seiten oder Anforderungen desselben Benutzers hinweg zu speichern und abzurufen. Anschließend werden bedingte Anweisungen (if/else) und Array-Funktionen verwendet, um Werte zu speichern und zu vergleichen. Der Code prüft, ob für jede der Variablen $tx, $rx und $drop_paket bereits eine Sitzungsvariable festgelegt wurde. Wenn die Variable nicht festgelegt wurde, wird sie mit einem Array initialisiert, das den aktuellen Wert enthält. Wenn die Variable gesetzt wurde, wird die Differenz zwischen dem aktuellen Wert und dem zuvor gespeicherten Wert mithilfe einer benutzerdefinierten Funktion namens "differentwert()" berechnet, und der aktuelle Wert wird für zukünftige Vergleiche in der Sitzungsvariablen gespeichert.

## 3.5 Entwicklung eines Parameterverwaltungssystems zwischen der Webanwendung und dem Router.

Um das Prioritystatus für jeder WAN Anschluss zu ändern, erstelle ich im Datei script_priority.js ein jQuery-Skript (Abbildung 27). Dieses Skript wird ausgeführt, wenn die Seite index.php geladen wird und ein Benutzer auf ein Radio-Button-Element klickt. Wenn ein Benutzer auf das Radio-Button-Element klickt, wird die Schaltfläche ausgeblendet. Anschließend wird über AJAX die Datei "priority.php" aufgerufen und dabei zwei Parameter, "value" und "id", übergeben. Die AJAX-Anfrage erfolgt asynchron, und die Ergebnisse werden in der Funktion "success" verarbeitet. In der "success"-Funktion wird die Funktion "updateTable()" aufgerufen. Diese Funktion lädt den Inhalt der Datei "view_dynamischetabelle.php" asynchron über AJAX und aktualisiert den Inhalt des HTML-Elements mit der ID "Demo" mit dem zurückgegebenen Inhalt. Dadurch wird die Tabelle auf der Webseite aktualisiert, ohne die gesamte Seite neu zu laden, sobald die Funktion aufgerufen wird.

Die Funktion "updateTable()" lädt asynchron den Inhalt der Datei view_dynamischetabelle.php mittels AJAX und aktualisiert den Inhalt des HTML-Elements mit der ID Demo mit dem zurückgegebenen Inhalt. Das bedeutet, dass die Tabelle auf der Webseite ohne die Seite neu zu laden aktualisiert wird, sobald die Funktion aufgerufen wird.

```
function updateTable() {
  $.get(
    '/peplink-connect3/src/libs/view_dynamischetabelle.php',
    function (result) {
      $('#demo').html(result);
    }
  );
}

$(document).ready(function () {
  $('input[type="radio"]').click(function () {
    var id = $(this).attr('data-id');
    var label = $('label[for="' + $(this).attr('id') + '"]');
    var value = $(this).attr('value');
    label.hide();
    $.ajax({
      async: true,
      method: 'POST',
      url: 'src/libs/priority.php',
      cache: false,
      data: {
        value: value,
        id: id,
      },
      success: function (result) {
        updateTable();
      },
    });
  });
});
```

*Abbildung 27: Datei script_priority.js*

```
<?php
require(__DIR__ . '/../../config/config.php');
require __DIR__ . "/encrypt.php";

$file_api = $config['file_api'];
include $file_api;

$fopen = json_decode(file_get_contents($config['login_file']), TRUE);
$url_1 = 'https://' . textdecryption($fopen["ip"], $method, $key_schluessel, $iv);

####_____###
$value = isset($_POST['value']) ? $_POST['value'] : '';
$id = isset($_POST['id']) ? ($_POST['id']) : "";
###_____####
$data_client_priority_andern = array(
    "cookie" => $config['cookie'],
    "data" => [
        "instantActive" => true,
        "list" => [
            [
                "connId" => (int)$id,
                "priority" => (int)$value
            ]
        ]
    ]

);
api($url_1, '/api/config.wan.connection.priority', $data_client_priority_andern, FALSE); //POST priority
```

*Abbildung 28: Datei priority.php*

In der Datei priority.php (Abbildung 28) liest mein PHP-Script am Anfang Konfigurationsdaten, die sich unter dem Pfad "config/config.php" befindet. Dann ruft das Skript die Funktion "textdecryption()" aus der Datei encrypt.php auf, um verschlüsselte Daten zu entschlüsseln. Anschließend wird eine POST-Anfrage an eine API gesendet, um die Priorität einer Verbindung zu ändern. Die POST-Anfrage wird an eine URL gesendet, die in der Variablen $url_1 gespeichert ist, und die Daten, die an die API gesendet werden sollen, werden in der Variablen $data_client_priority_andern gespeichert. Die Daten enthalten die Verbindungs-ID und die neue Priorität. Die Funktion "api()" wird aufgerufen, um die POST-Anfrage an die API zu senden.

## 3.6 Konfigurierung den Empfang und die Anzeige von Informationen im Echtzeit.

Um alle Informationen zu aktualisieren und in Echtzeit in zwei Tabellen anzuzeigen, habe ich zwei Dateien namens script_dynamischetabelle.js und script_tabelle_vpn.js erstellt, in denen ich JavaScript-Code entwickelt habe.

Für die WANs-Verbindungsdaten habe die Funktion "updateTable()" definiert, die auf einen jQuery-basierten Ereignishandler aufgerufen wird, wenn das Dokument vollständig geladen ist. Die Funktion "updateTable()" sendet eine GET-Anfrage an ein PHP-Skript unter "/peplink-connect3/src/libs/view_dynamischetabelle.php" und aktualisiert den HTML-Inhalt eines Elements mit der ID Demo basierend auf dem Ergebnis der Anfrage. Der "get()" -Methode in jQuery wird verwendet, um eine asynchrone HTTP-GET-Anfrage an die angegebene URL zu senden, und das Ergebnis der Anfrage wird dann als Argument an eine Callback-Funktion übergeben, die den Inhalt des Demo-Elements mithilfe der jQuery -Methode aktualisiert. Zusätzlich dazu wird mithilfe der "setInterval()"-Methode ein Timer eingerichtet, der die "updateTable()"-Funktion in regelmäßigen Intervallen aufruft. Das Intervall wird durch das Attribut data-attr eines HTML-Elements mit der Klasse ".datawan" festgelegt.

```
$(document).ready(function () {
  function updateTable() {
    $.get(
      '/peplink-connect3/src/libs/view_dynamischetabelle.php',
      function (result) {
        $('#demo').html(result);
      }
    );
  }

  updateTable();
  var setIntervalwan = document
    .querySelector('.datawan')
    .getAttribute('data-attr');
  setInterval(updateTable, setIntervalwan);
});
```

*Abbildung 29: script_dynamischetabelle.js*

```
<?php
require __DIR__ . "/../../config/config.php";
require __DIR__ . "/app.php";
require __DIR__ . "/tabellebauen.php";
require __DIR__ . "/dynamishetabelle.php";
?>

<table id="table-1" class="container-fluid">
    <div class="box1">
        <tr>
            <th>#</th>
            <th>WANs</th>
            <th>WAN Status</th>
            <th>Status Connection</th>
            <th>Type Connection</th>
            <th>Balken Grafik(Cellular)</th>
            <th>Priority Status</th>
            <th>Set Priority</th>
        </tr>
        <div id="demo"><?= dynamishetabelle($_count, $json_decoded); ?></div>
    </div>
</table>
```

*Abbildung 30: view_dynamischetabelle.php*

In der Datei script_tabelle_vpn.js (Abbildung 31) habe ich eine JavaScript-Funktion entwickelt, die die VPN-Tabelle auf der Webseite "Dashboard" aktualisiert. Diese Funktion wird ausgelöst, wenn das Dokument vollständig geladen ist, und verwendet jQuery, um eine AJAX-Anfrage an ein PHP-Skript zu stellen, das in der Datei view_tabelle_vpn.php zu finden ist (Abbildung 32). Dieses PHP-Skript generiert den HTML-Code für die Tabelle. Der resultierende HTML-Code wird dann in das Element mit der ID "#block-4" auf der Webseite eingefügt. Die Funktion liest auch das "data-attr"-Attribut eines Elements mit der Klasse ".datavpn", um das Intervall zu bestimmen, in dem die Tabelle aktualisiert werden soll. Dieses Intervall wird mithilfe der Funktion "setInterval()" festgelegt, die die Funktion "updatevpnTable()" in regelmäßigen Intervallen aufruft.

```
$(document).ready(function () {
  function updatevpnTable() {
    $.get(
      '/peplink-connect3/src/libs/view_tabelle_vpn.php',
      function (result) {
        $('#block-4').html(result);
      }
    );
  }

  updatevpnTable();
  var setIntervalvpn = document
    .querySelector('.datavpn')
    .getAttribute('data-attr');

  setInterval(updatevpnTable, setIntervalvpn);
});
```

*Abbildung 31: script_tabelle_vpn.js*

```
<?php
require __DIR__ . '/../../config/config.php';
require __DIR__ . '/app.php';
require __DIR__ . '/getsnmp.php';
require __DIR__ . '/tabelle_vpn.php';
require __DIR__ . '/snmp.php';
?>

<table id="table_2">
    <tr>
        <th>#</th>
        <th>VPNs</th>
        <th>Status VPNs</th>
        <div id="mydiv" class="mydiv">
            <?php
            vpnwanverbindung($_SESSION['count'], $json_decoded);
            ?>
        </div>
    </tr>
    <div id="main">
        <?php
        getsnmpdaten($_SESSION['count_vpn'], $_SESSION['count'],  $json_decoded, $json_decoded_vpn, $latency, $drop_paket_different, $tx_different, $rx_different, $vpn_Tx_ingesamt, $vpn_Rx_ingesamt, $aktualisieren);
        // print_r(getdate());

        ?>
    </div>
</table>
```

*Abbildung 32: view_tabelle_vpn.php*

Die beiden Funktionen, "updateTable()" und "updatevpnTable()", bieten eine dynamische Möglichkeit, WAN- und VPN-Daten auf einer Webseite anzuzeigen und zu aktualisieren, ohne dass ein vollständiges Neuladen der Seite erforderlich ist. Dies wird durch den Einsatz von AJAX-Anfragen und regelmäßigen Aktualisierungsintervallen ermöglicht.

Anschließend konfiguriere ich die Datei config.php (Abbildung 33). Im ersten Block initialisiere ich ein Array namens $config mit mehreren Konfigurationsoptionen für Dateipfade, Verschlüsselungsschlüssel und andere Einstellungen. Im zweiten Block lege ich die Zeitintervalle für die Aktualisierung der WAN- und VPN-Tabellen fest. Im dritten Codeblock prüfe ich, ob bestimmte Dateien vorhanden sind und lese deren Inhalt in die Variablen $json_decoded und $json_decoded_vpn. Anschließend zähle ich die Anzahl der WAN- und VPN-Verbindungen und speichere die Anzahl in Sitzungsvariablen. In den vierten und fünften Codeblöcken überprüfe ich zuvor gespeicherte WAN- und VPN-Daten in den Sitzungsvariablen, dekodiere sie und aktualisiere die entsprechenden Zählwerte in der Sitzung.

```php
<?php
##_____Confuguration_data
$config = [
    'file' => __DIR__ . "/../src/api/data/wantest.json",
    'file2' => __DIR__ . "/../src/api/data/pepvpn.json",
    'priority' => __DIR__ . "/../src/api/data/priority.json",
    'file_api' => __DIR__ . '/../src/api/api_peplink_connect.php',
    'cookie' =>  __DIR__ . "/../src/api/data/api_cookie.txt",
    'login_file' =>  __DIR__ . "/../src/api/data/login.json",
    'method' => "AES-256-CBC",
    'key_schluessel' => "033760fa712ab328cdcf063f5b19c93f74673488b88824e9d1fe14d8d3b9a0b52b2855759a10ea2b",
    'iv' => "1234567812345678",
];
#_____WAN/VPN Tabellle_____
$aktualisieren = 5;
$setIntervalvpn = $aktualisieren * 1000;
$setIntervalwan =  10000;

###_____
if (file_exists($config['file'])) {
    $json_decoded = json_decode(file_get_contents($config['file']), true);
    if (isset($json_decoded) && $json_decoded['stat'] != "fail") {
        $_count = count($json_decoded['response']['order']); //12 rechnen wie viele Verbindung(WANs)
    }
}
if (file_exists($config['file2'])) {
    $json_decoded_vpn = json_decode(file_get_contents($config['file2']), true);
    if (isset($json_decoded_vpn) && $json_decoded_vpn['stat'] != "fail") {
        $_count_vpn = count($json_decoded_vpn['response']['profile']['order']);
    }
}
if (isset($_SESSION['wan_data'])) {
    $json_decoded = json_decode($_SESSION['wan_data'], true);
    if ($json_decoded['stat'] != "fail") {
        $_count = count($json_decoded['response']['order']); //12 rechnen wie viele Verbindung(WANs)
        $_SESSION['count'] = $_count;
    }
}
if (isset($_SESSION['vpn_data'])) {
    $json_decoded_vpn = json_decode($_SESSION['vpn_data'], true);
    if ($json_decoded_vpn['stat'] != "fail") {
        $_count_vpn = count($json_decoded_vpn['response']['profile']['order']);
        $_SESSION['count_vpn'] = $_count_vpn;
    }
}
?>
<!DOCTYPE html>

<html>
<div class="datavpn" data-attr="<?= $setIntervalvpn; ?>"></div>
<div class="datawan" data-attr="<?= $setIntervalwan; ?>"></div>

</html>
```

*Abbildung: 33 config.php*

## 3.7 Umfangreiche Testung.

Zum Testen habe ich die Anwendung auf einem Raspberry Minicomputer installiert und die grundlegenden visuellen Einstellungen überprüft, wie das richtige Platzieren des Hauptfensters, die Anordnung der Tabellen, die Schriftgröße in der Tabelle und die Größe der Steuertasten. Diese wurden angepasst (Abbildung 34, 35, 36).

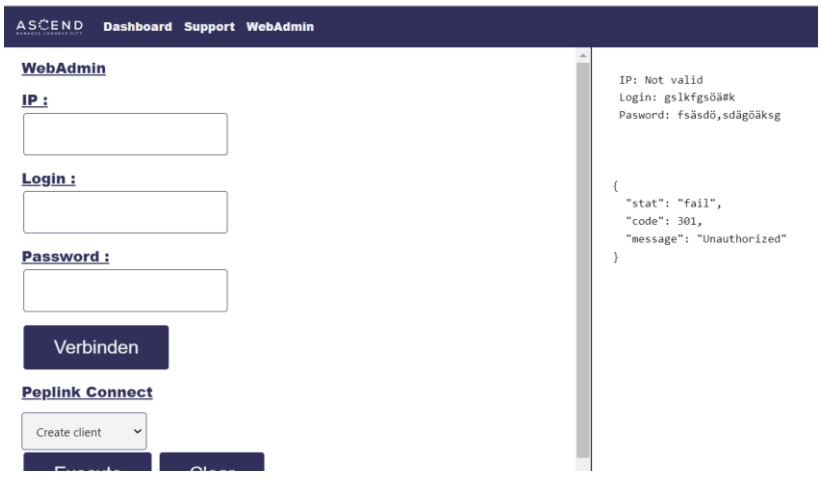



*Abbildung 34: Eingabe Falsche Daten (IP, Login und Password) auf "Support" Seite*

# 4 Projektergebnisse

## 4.1 Abschlusstest

Bei der Übergabe an den Kunden wurden die Anwendungen auf die Einhaltung der Anforderungen des Kunden getestet, indem sie an verschiedene Router-Modelle der Marke Peplink angeschlossen wurden:

- ✓ der Konfigurationsprozess wurde bei der Eingabe von IP, Login und Router-Pass überprüft;
- ✓ die Ausgabe und Änderung von Daten in Tabellen wurde in Echtzeit überprüft;
- ✓ der Datenänderungsmodus wurde überprüft, wenn sich der Wan-Status von Verbindungen ändert.

## 4.2 Soll-Ist-Vergleich

| Thema | Soll(Zeit) | Ist(Zeit) |
|---|---|---|
| **1. Analysephase** | | |
| Kundengespräch | 2,0 Std. | 2,0 Std. |
| Ist-Analyse | 2,0 Std. | **1,0 Std.** |
| Soll-Analyse | 2,0 Std. | 2,0 Std. |
| Lösungsmöglichkeiten darstellen/ Alternativen | 1,0 Std. | 2,0 Std. |
| Projektziele definieren | 1,0 Std. | 1,0 Std. |
| **2. Planungsphase** | | |
| Projektplanung | 2,0 Std. | 2,0 Std. |
| Kostenkalkulation | 2,0 Std. | 2,0 Std. |
| **3. Aus-/Durchführung** | | |
| Information strukturieren und entwickeln | 4,0 Std. | 4,0 Std. |
| Programmlogik zur Ansprache der API entwickeln | 12,0 Std. | **16,0 Std.** |
| Webanwendung entwickeln | 8,0 Std. | 8,0 Std. |
| Verarbeitung die Darstellung der empfangenen Informationen von Router | 8,0 Std. | **10,0 Std.** |
| Entwicklung eines Parameterverwaltungssystems zwischen der Webanwendung und dem Router | 9,0 Std. | 9,0 Std. |
| Konfigurierung den Empfang und die Anzeige von Informationen in Echtzeit. | 8,0 Std. | **4,0 Std.** |
| Umfangreiche Testung | 3,0 Std. | **1,0 Std.** |
| **4. Projektergebnisse** | | |
| Abschlusstest | 3,0 Std. | **2,0 Std.** |
| Soll-Ist-Vergleich | 3,0 Std. | **1,0 Std.** |
| Übergabe an den Kunden | 2,0 Std. | 2,0 Std. |
| **5. Projektabschlussphase** | | |
| Projektdokumentation | 8,0 Std. | **11,0 Std.** |
| **Gesamtzeit** | **80,0 Std.** | **80,0 Std.** |

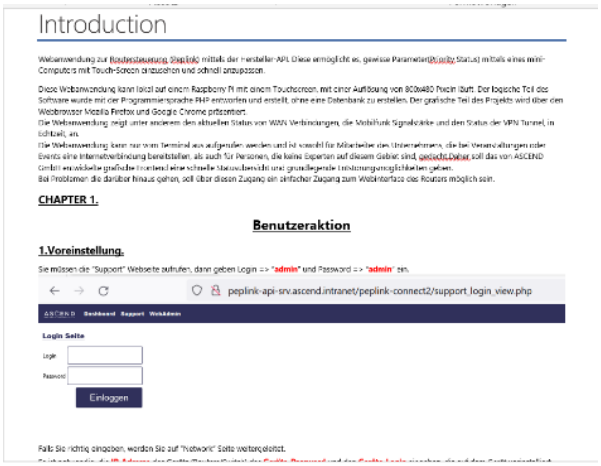
## 4.3 Übergabe an den Kunden

Bei der Übergabe der Anwendung wurde der Kunde darauf hingewiesen, dass die Anwendung Daten aus der geöffneten JSON-Datei liest und sie in einer HTML-Tabelle anzeigt. Es besteht das Risiko, dass ein Angreifer den Inhalt der JSON-Datei verändert, was dazu führen kann, dass in der HTML-Tabelle falsche Informationen angezeigt werden. Dies kann besonders gefährlich sein, wenn die Anwendung diese Daten verwendet, um Entscheidungen zu treffen oder Aktionen durchzuführen.

Obwohl die Anwendung keine Datenbank verwendet und nur Daten aus der geöffneten JSON-Datei liest und in einer HTML-Tabelle anzeigt, besteht das Hauptrisiko darin, dass die JSON-Datei von einem Eindringling verändert werden kann. Um dieses Risiko zu verringern, wird Folgendes empfohlen:

- Beschränken Sie den Zugriff auf die JSON-Datei auf schreibgeschützt. Dies kann z. B. durch die Verwendung von Berechtigungen im Dateisystem geschehen.
- Stellen Sie sicher, dass die Daten in der JSON-Datei keine bösartigen Skripte oder HTML-Tags enthalten. Dazu können Sie die Funktion "htmlspecialchars()" in PHP verwenden, die HTML-Sonderzeichen in ihre Entsprechung in Form von Zeicheneinheiten umwandelt.

Bei der Übergabe der Anwendung an den Kunden wurde ϕrcp eine technische Dokumentation erstellt, die die Betriebsanleitung, die Hauptfunktionsmerkmale des Programms und die Funktionslogik der Hauptfunktionen beschreibt.


Introduction

CHAPTER 1.

Benutzeraktion

1.Voreinstellung.

peplink-api-srv.ascend.intranet/peplink-connect2/support_login_view.php

Login Seite

Einloggen

*Abbildung 37: Technische Dokumentation*

# 5 Projektabschlussphase

## 5.1 Fazit

Obwohl ich im Herbst 2022 mit dem Erlernen der Programmiersprache PHP begonnen habe, konnte das Projekt innerhalb der geplanten Frist abgeschlossen werden. Die Hauptschwierigkeiten entstanden während der Konfigurationsphase der cURL-Bibliothek bei der Entwicklung der "api()" -Funktion, bei der Verarbeitung des Sendesignals zur Änderung des Prioritätsstatus beim Klicken auf die Schaltfläche. Außerdem dauerte es länger wie ich geplant habe, um die Darstellung der empfangenen Informationen vom Router zu verwalten. Aus diesem Grund hatte ich nicht genug Zeit, um einen Unit-Test zu entwickeln und durchzuführen. Bei der Codeentwicklung habe ich strukturierte Programmierung verwendet, das Programm ist in Funktionen, Bedingungen und Schleifen organisiert, im Wesentlichen habe ich einen prozeduralen Ansatz verwendet, nicht einen objektorientierten. Beim Erstellen der Anwendung habe ich festgestellt, dass der Hersteller die IP des Routers aktualisiert hat und danach die Verarbeitung einiger Informationen ändern musste. Fazit: Ich muss die Funktion "versionprueft()" verfeinern, die dem Kunden eine Warnung auf dem Bildschirm anzeigt und eine Nachricht an die IT-Abteilung sendet. Außerdem habe ich einige Sicherheitslücken bei der Datenspeicherung entdeckt, da die Anwendung Daten aus dem Inhalt von JSON- und Textdateien liest. Die Lösung für dieses Problem habe ich in Punkt 4.3 beschrieben.

# 6 Glossar

| Begriff / Funktionen | Beschreibung |
|---|---|
| API | Eine API (Application Programming Interface) ist eine Schnittstelle, die es Anwendungen ermöglicht, miteinander zu kommunizieren und Daten auszutauschen |
| Bootstrap-Clientframework | Bootstrap ist ein beliebtes Client-Framework für die Entwicklung von responsiven und benutzerfreundlichen Webanwendungen und bietet eine umfangreiche Sammlung von HTML-, CSS- und JavaScript-Vorlagen und -Komponenten. |
| Bootstrap-Klasse navbar | Die Bootstrap-Klasse "navbar" ermöglicht die Erstellung einer Navigationsleiste in einer Webanwendung. Mit dieser Klasse kann eine gutaussehende und responsive Navbar erstellt werden, die auf verschiedenen Bildschirmgrößen funktioniert. |
| cURL-Bibliothek | Die cURL-Bibliothek ist eine Softwarebibliothek, die Funktionen und APIs zur Verfügung stellt, um HTTP-, FTP-, SMTP- und andere Netzwerkprotokolle zu implementieren. |
| Drop Paket | Ein "Drop-Paket" bezieht sich normalerweise auf ein Datenpaket, das in einem Netzwerk verworfen oder abgelehnt wird. In Netzwerken werden Daten in Paketen übertragen, die von einem Gerät zum anderen übertragen werden |
| FileZila | FileZilla ist eine kostenlose und Open-Source-Software, die als FTP-Client (File Transfer Protocol) fungiert. Sie ermöglicht das Hochladen und Herunterladen von Dateien zwischen einem lokalen Computer und einem entfernten Server. |
| HTTP/ HTTPS | HTTP (Hypertext Transfer Protocol) und HTTPS (Hypertext Transfer Protocol Secure) sind zwei Protokolle, die zur Übertragung von Daten |

| Begriff / Funktionen | Beschreibung |
|---|---|
| | über das Internet verwendet werden. |
| IP-Adresse | Eine IP-Adresse (Internet Protocol Address) ist eine eindeutige numerische Kennung, die jedem Gerät in einem Computernetzwerk zugewiesen wird. Sie dient dazu, die Kommunikation zwischen verschiedenen Geräten in einem Netzwerk zu ermöglichen und die richtige Weiterleitung von Datenpaketen sicherzustellen. |
| jQuery-Skript | Ein jQuery-Skript ist ein Code, der die jQuery-Bibliothek verwendet, um interaktive und dynamische Funktionen in einer Webseite zu implementieren. |
| Latency | Latency (Latenz) bezieht sich auf die Verzögerung oder Zeitverzögerung, die bei der Übertragung von Daten in einem Netzwerk auftritt. Es ist die Zeitspanne, die vergeht, bevor Daten von einem Punkt zum anderen gelangen. |
| MIB Browser | Ein MIB-Browser (Management Information Base Browser) ist eine Softwareanwendung oder ein Tool, das verwendet wird, um auf MIB-Daten (Management Information Base) zuzugreifen und sie zu durchsuchen. |
| Popper.js | Popper.js ist eine JavaScript-Bibliothek, die verwendet wird, um das Positionieren und Ausrichten von HTML-Elementen zu erleichtern. |
| Priority Status | Im Zusammenhang mit Peplink bezieht sich der Priority-Status auf die Reihenfolge oder Rangfolge, die den verschiedenen WAN (Wide Area Network) -Verbindungen in einem Peplink-Router oder -Gerät zugewiesen wird. Peplink-Router unterstützen mehrere WAN-Verbindungen wie Ethernet, Wi-Fi, Mobilfunk oder Satellitenverbindungen, die für verschiedene Zwecke wie Lastausgleich, Ausfallsicherung oder Priorisierung des Datenverkehrs genutzt werden können. |
| Rx Geschwindigkeit | Rx (Receive) Geschwindigkeit bezieht sich auf die Datenübertragungsrate, mit der ein Gerät Daten empfangen kann. Es handelt sich um die Geschwindigkeit, mit der Daten von einem Sender an ein Empfängergerät übertragen werden. |
| Tx Geschwindigkeit | Tx (Transmit) Geschwindigkeit bezieht sich auf die Datenübertragungsrate, mit der ein Gerät Daten senden kann. Es handelt sich um die Geschwindigkeit, mit der Daten von einem Sender zu einem Empfängergerät übertragen werden |
| URL | URL steht für "Uniform Resource Locator" und ist eine Zeichenkette, die die Adresse einer Ressource im Internet angibt. Eine URL wird verwendet, um auf Webseiten, Dateien, Bilder, Videos und andere Inhalte im Internet zuzugreifen. |
| Use-Case Diagramm | Geht es um ein Diagrammtyp, der in der Softwareentwicklung verwendet wird, um die Funktionalität eines Systems aus der Sicht der Benutzer oder Akteure darzustellen |
| VPN-Verbindungen | (Virtual Private Network-Verbindungen) ermöglichen es Benutzern, eine sichere und verschlüsselte Verbindung über ein öffentliches Netzwerk herzustellen, wie zum Beispiel das Internet. Sie werden häufig verwendet, um eine sichere Kommunikation zwischen entfernten Standorten oder mobilen Geräten und einem zentralen Netzwerk herzustellen. |
| WANs-Verbindungen | (Wide Area Network-Verbindungen) beziehen sich auf die Verbindung von Computernetzwerken über große geografische Entfernungen hinweg. Hier sind einige wichtige Aspekte von WAN-Verbindungen: |
| XAMPP | XAMPP ist eine kostenlose, plattformübergreifende Softwarelösung, die entwickelt wurde, um die Einrichtung und Verwaltung eines lokalen Webentwicklungsumfelds zu erleichtern. Der Name XAMPP steht für "X" (für die verschiedenen Betriebssysteme), "Apache" (Webserver), "MySQL" (Datenbank), "PHP" (Skriptsprache) und "Perl" (Skriptsprache). |

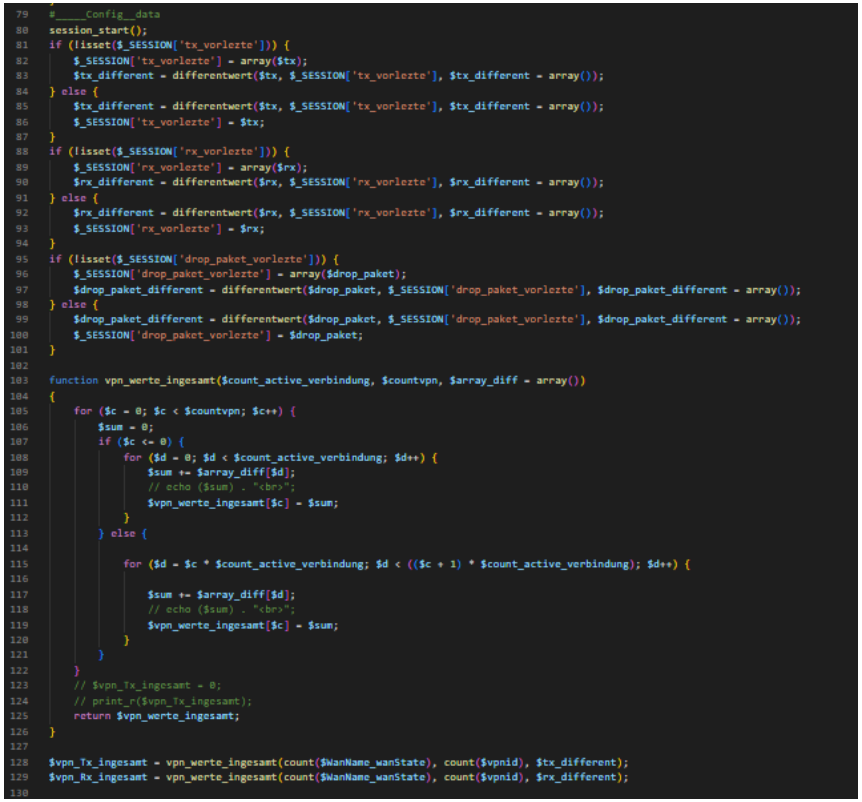
# 7 Anhang

```php
<?php
function getsnmpdaten($_count_vpn, $_count, $json_decoded, $json_decoded_vpn, $latency, $drop_paket_different, $tx_different,
                      $rx_different, $vpn_Tx_ingesamt, $vpn_Rx_ingesamt, $aktualisieren) {
    $colum_vpn_numer = 0;
    for ($b = 0; $b < $_count_vpn; $b++) {
    ?>
        <tr>
            <th><?= $colum_vpn_numer += 1; ?></th>
            <td>
                <?php echo $json_decoded_vpn['response']['profile'][$json_decoded_vpn["response"]['profile']["order"][$b]]['name']?>
            </td>
            <td>
                <?php if ($json_decoded_vpn['response']['profile'][$json_decoded_vpn["response"]['profile']["order"][$b]]["status"] === "CONNECTED") {
                ?>
                    <img src="./public/images/led-green.png" alt="led-green">
                <?php echo $json_decoded_vpn['response']['profile'][$json_decoded_vpn["response"]['profile']["order"][$b]]["status"];
                } else { echo  $json_decoded_vpn['response']['profile'][$b + 1]["status"]; }  ?>
            </td>
            <?php
            $varcount = 0;
            for ($i = 0; $i < $_count; $i++) {
                if ($json_decoded['response'][$i + 1]['enable'] === true and (str_contains($json_decoded['response'][$i + 1]['message'], "Connected") ||
                    $json_decoded['response'][$i + 1]['message'] === "Standby")) {  ?>
                    <td>
                        <?php
                        $kb = 1000;
                        $mb = 1000000;
                        if ($varcount < 1 && $b < 1) {
                            echo "Latency :  " . current($latency) . " ms " . "</br>";
                            echo "Drop Paket :  " . current($drop_paket_different) .  "</br>";
                            echo (current($tx_different) * 8 / $aktualisieren /  $kb) >= $kb ? "Tx :  " .  round(current($tx_different) * 8 / $aktualisieren / $mb, 1) . " Mbps " .
                                "</br>" : "Tx :  " .  round(current($tx_different) * 8 / $aktualisieren / $kb, 1) . " kbps " . "</br>";
                            echo (current($rx_different) * 8 / $aktualisieren / $kb)  >= $kb ? "Rx :  " .  round(current($rx_different) * 8 / $aktualisieren / $mb, 1) . " Mbps " .
                                "</br>" : "Rx :  " .  round(current($rx_different) * 8 / $aktualisieren / $kb, 1) . " kbps " . "</br>";
                            $varcount += 1;
                        } else {
                            echo "Latency :  " . next($latency) . " ms " . "</br>";
                            echo "Drop Paket :  " . next($drop_paket_different) . "</br>";
                            echo (next($tx_different) * 8 / $aktualisieren / $kb)  >= $kb ? "Tx :  " .  round(current($tx_different) * 8 / $aktualisieren / $mb, 1) . " Mbps " .
                                "</br>" : "Tx :  " .  round(current($tx_different) * 8 / $aktualisieren / $kb, 1) . " kbps " . "</br>";
                            echo (next($rx_different) * 8 / $aktualisieren / $kb)  >= $kb ? "Rx :  " .  round(current($rx_different) * 8 / $aktualisieren / $mb, 1) . " Mbps " .
                                "</br>" : "Rx :  " .  round(current($rx_different) * 8 / $aktualisieren / $kb, 1) . " kbps " . "</br>";
                        } ?>
                    </td><?php }
            }
            ?> <td>
                <?php
                echo (round($vpn_Tx_ingesamt[$b] * 8 / $aktualisieren / $kb, 1) >= $kb) ? "Tx :  " . round($vpn_Tx_ingesamt[$b] * 8 / $aktualisieren / $mb, 1) . " Mbps" .
                    "</br>" : "Tx :  " . round($vpn_Tx_ingesamt[$b] * 8 / $aktualisieren / $kb, 1) . " kbps" . "</br>";
                echo (round($vpn_Rx_ingesamt[$b] * 8 / $aktualisieren / $kb, 1) >= $kb) ? "Rx :  " . round($vpn_Rx_ingesamt[$b] * 8 / $aktualisieren / $mb, 1) . " Mbps" .
                    "</br>" : "Rx :  " . round($vpn_Rx_ingesamt[$b] * 8 / $aktualisieren / $kb, 1) . " kbps" . "</br>";
                ?>
            </td>
        </tr> <?php }
};?>
```

*Abbildung 24: Funktion "getsnmdaten()".*

```php
<?php
snmp_set_quick_print(1);
require_once('encrypt.php');
require __DIR__ . '/../../config/config.php';
$fopen = json_decode(file_get_contents($config['login_file']), TRUE);
$host = textdecryption($fopen["ip"], $method, $key_schluessel, $iv);
$community = 'public';
## pepVpnStatusTable(VPN Verbindung)
snmp_set_oid_output_format(SNMP_OID_OUTPUT_NUMERIC);

$deviceFirmwareVersion_neue = snmprealwalk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.1.1.1.3.0"); // deviceFirmwareVersion
$_SESSION['deviceFirmwareVersion'] = "8.3.0 build 5200";
session_write_close();

$vpnid = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.2.1.2"); //name VPN  pepVpnStatusProfileName

$vpn_status_connection = array(0, 1, 2, 3); //Werte die passt nicht
$pepVpnStatusId = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.2.1.3"); //status vpn "pepVpnStatusConnectionState"

### pepVpnStatusWanTable
$pepVpnStatusWanId = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.1"); //id pepVpnStatusWanId
$pepVpnStatusWanName = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.2"); //name "pepVpnStatusWanName"

$tx = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.3"); //"pepVpnStatusWanTxBytes"
$rx = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.4"); //pepVpnStatusWanRxBytes"

$drop_paket = snmp2_real_walk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.5"); //"pepVpnStatusWanDropPackets"
$latency = snmprealwalk("$host", "$community", ".1.3.6.1.4.1.23695.200.1.10.1.1.3.1.6"); //"pepVpnStatusWanLatency"

$count_pepVpnStatusId = count($pepVpnStatusId);
$WanName = snmprealwalk("$host", "$community", ".1.3.6.1.4.1.23695.2.1.2.1.2"); //WAN Name

snmp_set_valueretrieval(SNMP_VALUE_PLAIN);

$wanState = snmprealwalk("$host", "$community", ".1.3.6.1.4.1.23695.2.1.2.1.3"); //WAN Status

###_combine arrayWANname und Wan Status und raumen________
$WanName_wanState = array_combine($WanName, $wanState); //new array key=Wanname; werte=> Wanstate
$count_verbindung = count($WanName_wanState); // wie viele Wan vebindung
$wanState_Wert = array('0', '1', '2', '4', '5', '6', '7'); //Werte die nicht passen
$WanName_wanState = array_diff($WanName_wanState, $wanState_Wert); //Wan vebindung die  aktiv sind

###____verkurz OID
function verkurzoid($array_verkurz)
{
    foreach ($array_verkurz as $key => $value) {
        unset($array_verkurz[$key]);
        $array_verkurz[substr($key, -5)] = $value;
    }
    return $array_verkurz;
}
$latency = verkurzoid($latency); //latency
$pepVpnStatusWanName = verkurzoid($pepVpnStatusWanName); //VpnWanName
$tx = verkurzoid($tx); //Tx
$rx = verkurzoid($rx); //Rx
$drop_paket = verkurzoid($drop_paket); //DropPaket

###___ vergleich Werte mit key und raumen
foreach ($pepVpnStatusWanName  as  $val) {
    if (!array_key_exists($val, $WanName_wanState)) {
        $pepVpnStatusWanName = array_diff($pepVpnStatusWanName, ["$val"]);
    }
}

$latency = array_intersect_key($latency, $pepVpnStatusWanName);
$tx = array_intersect_key($tx, $pepVpnStatusWanName);
$rx = array_intersect_key($rx, $pepVpnStatusWanName);
$drop_paket = array_intersect_key($drop_paket, $pepVpnStatusWanName);
//____differen wert
function differentwert($array, $array_vorlezte, $newarray)
{
    foreach ($array  as $key => $val) {
        if (array_key_exists($key, $array_vorlezte)) {
            $newarray[] = $val - $array_vorlezte[$key];
        }
    }
    return $newarray;
}
#_____Config__data
```

*Abbildung 25: Datei snmp.php-Ausschnitt 1 von 2*



```php
#_____Config__data
session_start();
if (!isset($_SESSION['tx_vorlezte'])) {
    $_SESSION['tx_vorlezte'] = array($tx);
    $tx_different = differentwert($tx, $_SESSION['tx_vorlezte'], $tx_different = array());
} else {
    $tx_different = differentwert($tx, $_SESSION['tx_vorlezte'], $tx_different = array());
    $_SESSION['tx_vorlezte'] = $tx;
}
if (!isset($_SESSION['rx_vorlezte'])) {
    $_SESSION['rx_vorlezte'] = array($rx);
    $rx_different = differentwert($rx, $_SESSION['rx_vorlezte'], $rx_different = array());
} else {
    $rx_different = differentwert($rx, $_SESSION['rx_vorlezte'], $rx_different = array());
    $_SESSION['rx_vorlezte'] = $rx;
}
if (!isset($_SESSION['drop_paket_vorlezte'])) {
    $_SESSION['drop_paket_vorlezte'] = array($drop_paket);
    $drop_paket_different = differentwert($drop_paket, $_SESSION['drop_paket_vorlezte'], $drop_paket_different = array());
} else {
    $drop_paket_different = differentwert($drop_paket, $_SESSION['drop_paket_vorlezte'], $drop_paket_different = array());
    $_SESSION['drop_paket_vorlezte'] = $drop_paket;
}

function vpn_werte_ingesamt($count_active_verbindung, $countvpn, $array_diff = array())
{
    for ($c = 0; $c < $countvpn; $c++) {
        $sum = 0;
        if ($c <= 0) {
            for ($d = 0; $d < $count_active_verbindung; $d++) {
                $sum += $array_diff[$d];
                // echo ($sum) . "<br>";
                $vpn_werte_ingesamt[$c] = $sum;
            }
        } else {

            for ($d = $c * $count_active_verbindung; $d < (($c + 1) * $count_active_verbindung); $d++) {

                $sum += $array_diff[$d];
                // echo ($sum) . "<br>";
                $vpn_werte_ingesamt[$c] = $sum;
            }
        }
    }
    // $vpn_Tx_ingesamt = 0;
    // print_r($vpn_Tx_ingesamt);
    return $vpn_werte_ingesamt;
}

$vpn_Tx_ingesamt = vpn_werte_ingesamt(count($WanName_wanState), count($vpnid), $tx_different);
$vpn_Rx_ingesamt = vpn_werte_ingesamt(count($WanName_wanState), count($vpnid), $rx_different);
```

*Abbildung 26: Datei snmp.php-Ausschnitt 2 von 2*

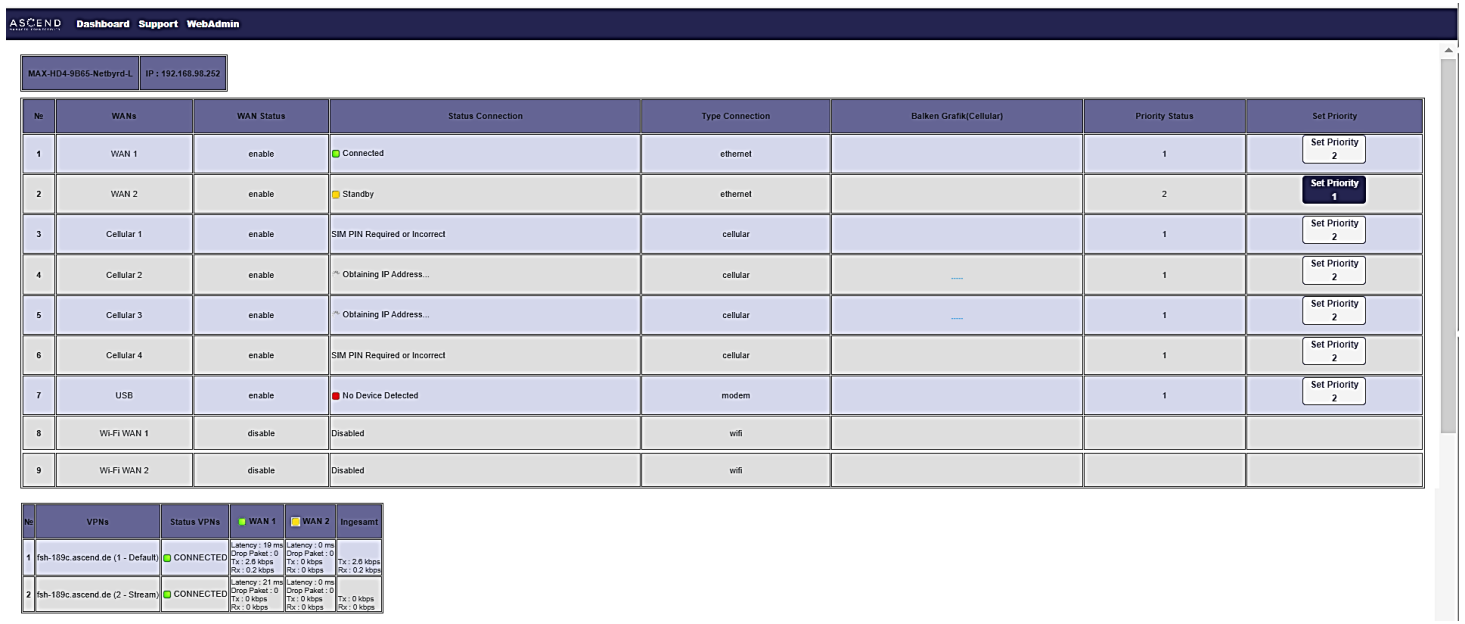



*Abbildung 35: Überprüfen des Empfangs, der Aktualisierung von Daten auf "Dashboard" Seite.*

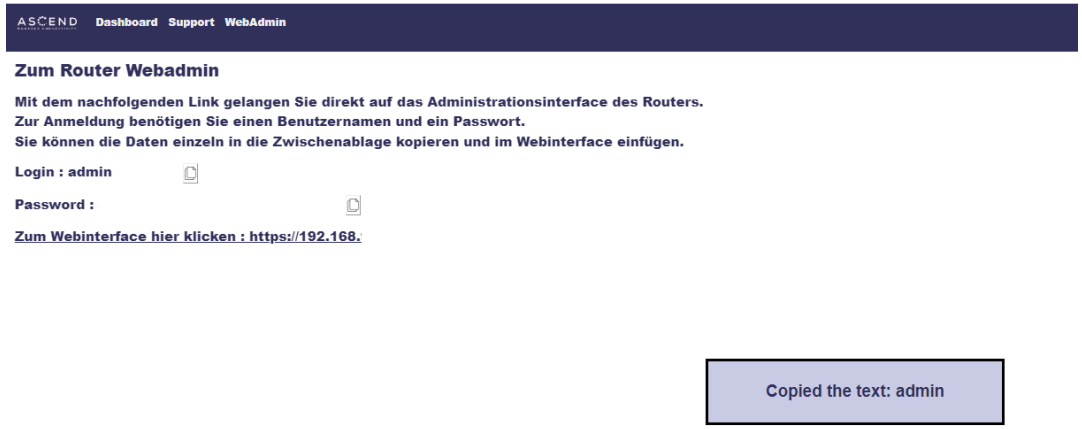



*Abbildung 36: "WebAdmin" Seite- Bestätigen des kopierten Elements*

```
index.php
├──config
│   config.php
├──public
│   checkpass_support.php
│   support_login_view.php
│   web_admin.php
│   ├──css
│   │   index.css
│   │   login_view.css
│   │   network.css
│   │   web_admin.css
│   ├──images
│   │   Ascend_LOGO-08_cropped.png
│   │   ascend_logo-1.png
│   │   Ascend_LOGO-11_Cropped.png
│   │   copy_83918.png
│   │   dialog-throbber.gif
│   │   led-flash.gif
│   │   led-green.png
│   │   led-orange.png
│   │   led-red.png
│   │   signal5-0.gif
│   │   signal5-1.gif
│   │   signal5-2.gif
│   │   signal5-3.gif
│   │   signal5-4.gif
│   │   signal5-5.gif
│   └──mibs
└──src
    ├──api
    │   api_peplink_connect.php
    │   network.php
    │   └──data
    │       api_cookie.txt
    │       login.json
    │       wantest.json
    │       pepvpn.json
    ├──inc
    │   navbar.php
    └──libs
        app.php
        checkfiles.php
        dynamishetabelle.php
        encrypt.php
        getsnmp.php
        priority.php
        script_dynamischetabelle.js
        script_priority.js
        script_tabelle_vpn.js
        snmp.php
        snmpname.php
        tabellebauen.php
        tabelle_vpn.php
        versioprueft.php
        view_dynamischetabelle.php
        view_tabelle_vpn.php
        └──data
```

*Abbildung 38: Projektstruktur*