VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Dávid Kováč

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ROZHRANÍ PRO ŘÍZENÍ ŘÍDICÍ JEDNOTKY PASIVNÍ OPTICKÉ SÍTĚ
INTERFACE FOR MANAGEMENT OF OPTICAL LINE TERMINATION IN PASSIVE OPTICAL NETWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Dávid Kováč**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Horváth, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Dávid Kováč ***ID:*** 214079

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Rozhraní pro řízení řídicí jednotky pasivní optické sítě

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je teoretický rozbor možností realizace grafického uživatelského rozhraní (GUI) pro ovládání řídicí jednotky. Řídicí jednotka podporuje přihlášení po SSH protokolu a GUI nástavba by měla umožnit klíčové nastavení koncových jednotek – aktivace koncové jednotky, tvorba profilů pro jednotlivé jednotky a vyčítání parametrů (sériové číslo, výkonová úroveň apod.). V praktické části bakalářské práce student vypracuje grafické rozhraní pro vyčítání parametrů veškerých koncových jednotek, kompletní aktivaci koncové jednotky a pokročilé nastavení alarmů a detekci událostí v pasivní optické síti.

DOPORUČENÁ LITERATURA:

[1] HOOD, Dave a Elmar TROJER. Gigabit-capable passive optical networks. Hoboken: Wiley, c2012. ISBN 978-0470936870.

[2] E. WILLNER, Alan, ed. Optical Fiber Telecommunications VII. Cambridge, Massachusetts: Academic Press, 2019. ISBN 978-0-12-816502-7.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Tomáš Horváth, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

VYHLÁSENIE

Vyhlasujem, že svoju bakalársku prácu na tému „GUI pro řízení řídicí jednotky pasivní optické sítě“ som vypracoval samostatne pod vedením vedúceho bakalárskej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu bakalárskej práce pánovi Ing. Tomášovi Horváthovi Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

ABSTRAKT

Táto bakalárska práca sa zaoberá s vytvorením Grafického rozhrania pre konfiguráciu riadiacej jednotky pasívnej optickej sieti. Cieľom práce bolo uľahčiť proces konfigurácie riadiacej jednotky. Vytvorené grafické rozhranie dovoľuje užívateľovi konfiguráciu koncového zariadenia, výpis pripojených koncových zariadený, pozrieť záznam udalostí a nastavenie alarmu. Teoretická časť práce sa zaoberá zo zoznámením sa s pasívnymi optickými sieťami a s problematikou práce. Ďalej sa nachádza výber programovacích prostriedkov a rozšírenia ktoré boli použité pri praktickej časti práce, vysvetlenie a zdôvodnenie spôsobu tvorby jednotlivých častí programu a to triedy a metódy ktoré boli programované, popis testovania programu a dosiahnuté výsledky v práci. Na konci je manuál pre prácu s programom, ktorý obsahuje návod ako nainštalovať a pracovať s ním.

KĽÚČOVÉ SLOVÁ

Grafické užívateľské rozhranie, konfigurácia, Java, HTML, IntelliJ IDEA, SpringWEB, Thymeleaf, GPON, pasívna optická sieť,SSH

ABSTRACT

This bachelors thesis deals with the creation of a graphic user interface for the configuration of optical line terminal for passive optical networks. The aim of the thesis was to make the configuration of optical line terminal easier and more convenient. The graphic user interface allows the user to configure ONTs, display all connected ONTs, show the event log and to configure alarm profiles. The theoretical part of the thesis deals with getting acquainted with passive optical networks and getting familiar with how the graphical interface was created. Furthermore, there are the programming tools and the extension used to create the GUI, an explanation of how and why each part of the GUI was created, including the classes and the methods which were used, how the testing went and the results. At the end of the thesis is a manual of what needs to be installed to make the GUI work and how to use it.

KEYWORDS

Graphic user Interface, configuration, Java, HTML, IntelliJ IDEA, SpringWEB, Thymeleaf, GPON, pasive optical network, SSH

KOVÁČ, Dávid. *GUI pro řízení řídicí jednotky pasivní optické sítě*. Brno, 2030, 61 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. TOMÁŠ HORVÁTH, Ph.D

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

Táto bakalárska práca sa zaoberá s vytvorením Grafického rozhrania pre konfiguráciu pasívnych optických sietí, zaznamenanie a výpis udalostí, výpis statusov koncových jednotiek a konfiguráciu alarmu. Prvá kapitola obsahuje základné informácie k pasívnym optickým sieťam. Ďalšia kapitola obsahuje informácie k typom pasívnych optických sietí ktoré sa budú konfigurovať. Tretia kapitola porovnáva konfiguráciu pomocou príkazového riadku a grafického rozhrania. Vo štvrtej kapitole je teoretický návrh pre grafické rozhranie, vybrané programy a praktická časť práce. V piatej kapitole je manuál na prácu vytvoreným grafickým rozhraním. Výstupom práce je vytvorenie grafického rozhrania na konfigurácie riadiacej jednotky ktorá by nahradila klasický spôsob konfigurácie cez príkazový riadok. Konfigurácia pomocou grafického rozhrania by odstránila nutnosť výpisu celých príkazov a možných chýb syntaxe. Pripojenie k riadiacej jednotke bude riešené pomocou protokol Secure Shell [18].

1 Pasívne optické siete

Pasívne optické siete (PON) sú telekomunikačná technológia využívajúca optické vlákna spolu s point to multipoint architektúrou. Dáta sú posielané z jedného prenosového bodu do viacerých koncových zariadení. PON dostalo svoje pomenovanie vďaka tomu, že na trase medzi riadiacou jednotkou (OTL) a koncovou jednotkou (ONU) nepoužívajú žiadne aktívne prvky, ktoré by potrebovali napájanie, ale používajú pasívne rozbočovače a väzbové členy k rozdelení a vysielaní prenosu k koncovým užívateľom. Pasívne siete sú spoľahlivejšie a lacnejšie ako aktívne optické siete[1].

1.1 Aplikácia pasívnej optickej siete

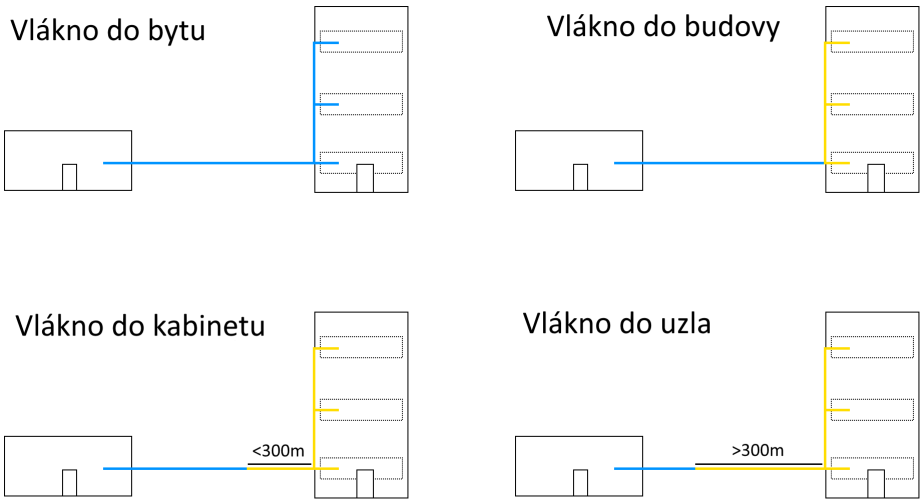
Pasívne optické siete sú označované aj ako posledná míľa medzi poskytovateľom a spotrebiteľom alebo nazývaný aj ako vlákno k x (FTTx), kde x udáva kde je ukončené vlákno. Najčastejšie zapojenie FTTx je vlákno do bytu (FTTH). Tým že pasívne optické siete neobsahujú aktívne prvky majú zmenšenú kabelážovú infraštruktúru. Flexibilný prenos mediálnych dát ho robia výborným pre internet, hlas a video aplikácie[2].

1.2 Fiber to the X

Fiber to the X (FTTx) je výraz pre sieťovú architektúru, ktorá v poslednej míle zabezpečuje celú alebo časť širokopásmové pripojenie k užívateľom. FTTx využíva pre pripojenie užívateľov optické vlákna. X predstavuje objekt alebo budovu ku ktorej sa FFTx pripája, napríklad H pre byt, R pre smerovač alebo D pre dvere. FTTx sa rozdeľuje do dvoch skupín FTTP a FTTC/N. FTTP dodáva optické vlákno po celej ceste od ústrednej kancelárie do areálu užívateľa. Pripojenie FTTP k požadovanej miestnosti môže byť čisto optická alebo môže mať posledné metre pripojené s neopticky. FTTC/N dodáva optické vlákno do kabinetu z ktorého sa k užívateľovi pripája pomocou neoptických pripojení. FTTx dovoľuje prenášať vysoké rýchlosti do vzdialenosti niekoľko desiatkoch kilometrov, kde krútené dvoj-páry by tú istú rýchlosť vedeli dodať len do 100 metrov[2, 5].

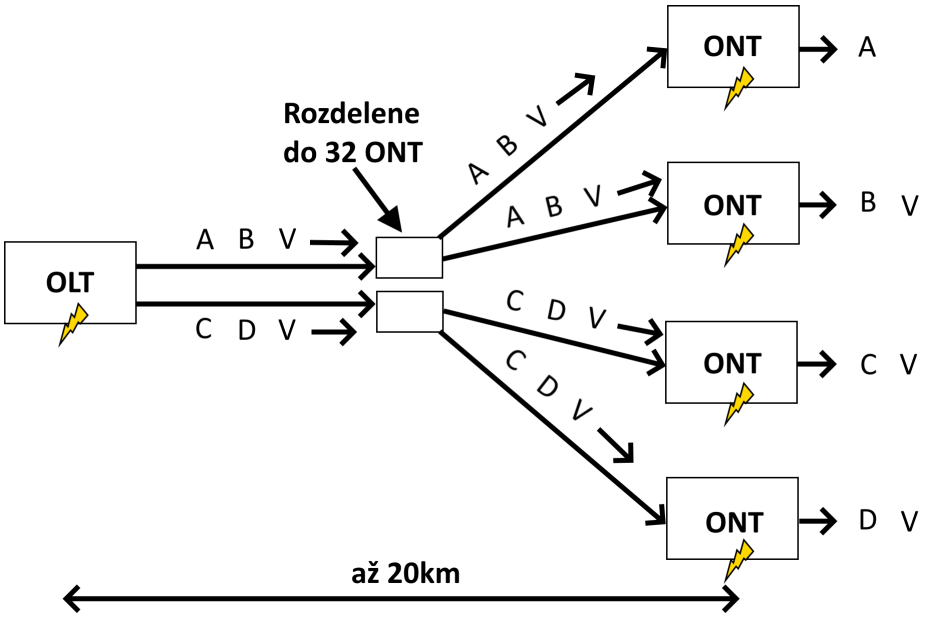
1.3 Štruktúra a komponenty PON

Pasívne optické siete distribuujú signály pomocou deliča lúčov. Deliče rozdeľujú optický signál na požadovaný počet zákazníkov. Od deliča má každý zákazník vlastné



Obr. 1.1: Porovnanie rôznych druhov FTTx [5]

optické vlákno. V jednom kabinete sa môžu nachádzať viacere deliče. Delič si nevyberá vlnové dĺžky ale delí všetky vlnové dĺžky ktoré mu prídu. Nemé medzi-pamäť a vytvára Point to Multipoint link. Toto zapojenie spôsobuje že každý odoslaný signál z riadiacej jednotky je broadcastový čo vytvára potrebu nastavenia filtrovania dát na koncových jednotkách užívateľov. Filtrácia sa dosiahne pomocou optického izolátora ktorý dovoľuje signálu prejsť len jednou stranou, alebo optickým filtrom ktorý dovolí prejsť len požadovanej vlnovej dĺžke. Ak filtrácia by nebola nastavená každý užívateľ by dostával všetky dáta posielané na sieti. Tým že signál je posielaný každému užívateľova je potrebné v ústrednej kancelárií mať výkonné zariadenie na prenos signálu nazývané riadiaca jednotka. Je potrebné aj zapojiť koncové zariadenia do multiplexového zapojenia aby sa zabránilo kolízií signálov. Používajú sa dve typy multiplexov: vlnovej dĺžky a časový. S multiplexorom vlnovej dĺžky (WDM) užívatelia posielajú signály v rozdielnych šírkach pásma. S časovým multiplexorom (TDM) užívatelia sa striedajú pri posielaní dát. Pasívne optické siete obsahujú riadiacu jednotku u dodávateľa a koncovú jednotku u užívateľa, tieto jednotky sú jedinou aktívnou časťou v pasívnej optickej sieti[4].



Obr. 1.2: Štruktúra pasívnej optickej sieti [1]

1.4 Riadiaca jednotka

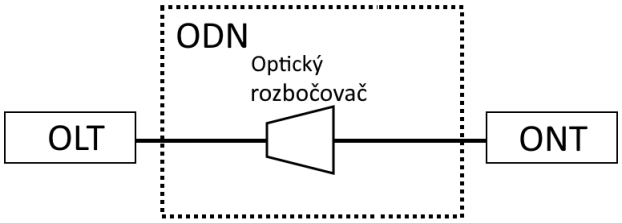
Riadiaca jednotka (OLTl) je začiatočný bod pre PON, ktorý je spojený s prepínačom pomocou ethernetových káblov. Jeho hlavnou funkciou je prevedenie, zarámovanie, vysielanie signálov a koordinovanie multiplexovania ONU pre zdieľaný upstream. Riadiaca jednotka sa všeobecne skladá zo stojana, riadiaceho a prepínacieho modulu (CSM), modul EPON link, karty PON, ochrany proti redundancií, 48V DC napájania alebo 110/220V AC napájania a ventilátora. Riadiaca jednotka pracuje v dvoch smeroch, vzostupne kde zabezpečuje distribúciu dát a hlasového prenosu od spotrebiteľov a zostupne, kde zabezpečuje posielanie dát, hlasu a video do každého koncového zariadenia v optickej distribučnej sieti[4].

1.5 Koncová jednotka

Koncová jednotka (ONU) slúži na prevedenie optického signálu doručených z optického vlákna na elektrický signál, ktorý je posielaný individuálnym zákazníkom. Koncová jednotka dokáže posielať, agregovať a upravovať dáta. Koncová jednotka podporuje pripojenia pomocou krúteného páru, koaxiálneho kábla, optického kábla alebo bezdrôtovo cez Wi-Fi. Koncová jednotka má pomenovanie ONT podľa ITU-T terminológie a ONU podľa IEEE terminológie.[4]

1.6 Optická distribučná sieť

Optická distribučná sieť (ODN) zaobstaráva optický prenos pre fyzické spojenie medzi riadiacou jednotkou a koncovou jednotkou. Obsahuje káble z optických vlákien, optické konektory, pasívne optické rozdeľovače a pomocné súčiastky, ktoré medzi sebou spolupracujú. Do optickej distribučnej sieti sa z riadiacej jednotky posiela signál, Signál v je ODN spracovaný a pomocou optického deliča je rozdelená na požadovaný počet zákazníkov. Von z ODN je signál posielaný do koncových jednotiek ktoré zabezpečujú pripojenie zákazníkov. Zapojenie optickej distribučnej siete ovplyvňuje prenos, kvalitu, spoľahlivosť a škálovateľnosť pasívnych optických sietí.[4]



Obr. 1.3: Príklad optickej distribučnej sieti [4]

1.7 Výhody PON

Najzásadnejšou výhodou je že na prevádzkovanie siete je potrebné len napájanie pre riadiacu a koncovú jednotku. Toto zjednodušuje údržbu siete a znižuje pravdepodobnosť zlyhania napájaných zariadení. Architektúra odstraňuje potrebu pre rozvodové skrinky a chladiacu infraštruktúru. Infraštruktúra dovoľuje viacerým službám existovať na jednej pasívnej optickej sieti čo umožňuje získať väčšiu šírku pásma na jednom vlákne. Optické siete nie sú náchylné na rušenie, kontroluje sa či aktívne zariadenia správne prenášajú signál a či pasívne zariadenia nemajú veľkú stratu signálu.[1, 2]

1.8 Nevýhody PON

Nevýhodou je malá vzdialenosť na ktorý signál vie posielať. Chyby v sieti sú náročnejšie na odhalenie. Kvôli jeho point to multi point architektúre pri prerušení vlákna, alebo pri chybnom OLT môže vytvoriť rozsiahle výpadky siete. Navyše, tým že šírka pásma nie je vyhradená pre jednotlivých používateľov, rýchlosť prenosu dát sa pri hromadnom využití siete môže spomaliť a spôsobovať oneskorenie[1, 2].

2 GPON a XG-PON

2.1 GPON

GPON je štandardom pre PON vydaný ITU-T. Je často používaný pri realizovaní posledného kilometra FTTP služieb. GPON udáva požiadavky na optické médium a hardvér a definuje spôsob transformácie ethernetového rámca na optický signál. Prvá verzia GPON bola schválená v roku 2003, od vtedy sa rozširovala a upravovala. Používa rôzne vlnové dĺžky pre zostupne a vzostupne. Zostupne pracuje na vlnovej dĺžke 1480-1500 nm a má rýchlosť 2,5Gbit/s. Vzostupne pracuje na vlnovej dĺžke 1290-1330 nm a má rýchlosť 1,25Gbit/s. Štruktúra rámcu GPON je GEM. Maximálna vzdialenosť prenosu je 60km a najväčšie možné rozdelenie signálu je 1 na 128. GPON upresňuje protokoly pre úpravu chýb, zašifrovanie a autentifikáciu[7].

2.2 XG-PON

XG-PON je sieťový štandart pre data link. Je nasledujúci štandart po GPON od ITU-T. XG-PON ako aj GPON používa rôzne vlnové dĺžky na prenos dát. Zostupne pracuje na vlnovej dĺžke 1575-1580 nm a má rýchlosť 10Gbit/s. Vzostupne pracuje na vlnovej dĺžke 1260-1280 nm a má rýchlosť 2,5Gbit/s. Štruktúra rámcu XG-PON je XGEM. Maximálna vzdialenosť prenosu je 100km a najväčšie možné rozdelenie signálu je 1 na 256. Tým že vlnové dĺžky GPON a XG-PON sa neprekrývajú umožňuje v jednej sieti mať naraz oba štandardy. Spracovanie dát je podobné ako u GPON. Symetrická verzia XG-PON sa nazýva XGS-PON a umožňuje 10Gbit/s vzostupne a zostupne[7] .

2.3 Princíp posielania dát na GPON a XG-PON

GPON a XG-PON obsahujú dve aktívne prenosové zariadenia a to riadiacu a koncovú jednotku. Jedno vlákno ide od riadiacej jednotky do pasívneho optického rozdeľovača, ktorý sa nachádza v areáli užívateľa. Optický rozdeľovač prijatý signál rozdelí na X ciest k užívateľom. Od optického rozdeľovača ide jedno vlákno ku každému užívateľovi. Používajú sa dve multiplexovacie mechanizmy. Od riadiacej jednotky ku koncovej jednotke sa signál posiela broadcastom a je použité zabezpečenie AES ma ochranu proti odpočúvaniu. Pri opačnom smere sa signáli od jednotlivých užívateľov posielajú postupne v časovom intervale ktorý je im priradený. Aby koncové zariadenie mohlo dáta posielať v vzostupnom smere musí riadiaca jednotka vytvoriť mapovanie medzi T-CONT a ALLOC-ID ktorú sú pri aktivácií koncovej jednotke

automaticky priradené pomocou PLOAM správy. Riadiaca jednotka tiež dynamicky prideľuje šírku pásma vzostupného smeru pre jednotlivé koncové jednotky. Tým, že prístup k sieti je zdielaná by vysielanie signálu od koncových jednotiek v náhodných časoch spôsobila kolíziu. Riadiaca jednotka odmeria oneskorenie vyšle PLOAM správu aby zjednotila oneskorenie s ohľadom ku všetkých koncových jednotiek. Riadiaca jednotka potom vyšle grants správu ktorá dá na určitú dobu povolenie koncovej jednotke na posielanie dát vzostupne smere. Riadiaca jednotka priraďuje šírku pásma každej koncovej jednotke tak aby mala dostatočný čas pre jej potreby[6].

3 Konfigurácia GPON a XG-PON systémov

Konfigurácia slúži na nastavenie riadiacich jednotiek (OLT) na zabezpečenie ich správnej funkcií. Pomocou konfigurácií sa dá vytvoriť DBA profily, nastaviť linkový profil, vytvoriť servisné profily pre koncové jednotky, skontrolovať procesor a použitú pamäť, pozrieť teplotu dosiek, pripojenie koncových jednotiek. Konfigurovať GPON a XG-PON systémy sa dá pomocou rozhrania v príkazovom riadku (CLI) alebo pomocou grafického rozhrania (GUI). S týmito rozhraniami sa na riadiacu jednotku môžeme pripojiť priamo cez kábel alebo ak daný systém to ponúka tak bezdrôtovo cez SSH prípadne cez internet pomocou IP adresy[8].

3.1 Konfigurácia pomocou CLI

Pomocou príkazov sa dá dostať na jednotlivé porty systému kde ich vieme nastavovať. Pri nastavovaní pomocou príkazového riadku je potrebné dávať pozor nie len na správnu syntax príkazov ale aj na privilégiách, profil alebo na aktuálny port. Napríklad ak by sme chceli nakonfigurovať vytvorený DBA profil by bolo prvé potrebné sa dostať do (config#) časti systému pomocou príkazov enable a config, priradiť k DBA vytvorený profil vybrať typ profilu nastaviť požadovanú hodnotu. Tým že každý príkaz treba písať vlastnoručne a je závislé kde v systéme sa píše daný príkaz je konfigurácia v CLI ťažšia pre používateľa ak nemá skúsenosť s konfiguráciou poprípade ak pracuje s CLI od rôznych vydavateľov kde sa syntax jednotlivých príkazov môže líšiť.

3.2 Konfigurácia koncovej jednotky

Konfiguráciu sa dá uskutočniť v dvoch režimoch Profile a Simplified. Simplified je zjednodušená konfigurácia ale má obmedzenia. V zjednodušenom režime je GEM port pridelený automaticky systémom. DBA, line a service profili netreba konfigurovať je automaticky priradené predvolené profily. Medzi obmedzenia tohoto režimu patria, že End to End a non End to End nemôžu koexistovať na jednej koncovej jednotke. Ak GEM je nastavený na port, port+VLAN tak sa nemôže vytvoriť End to End na koncovej jednotke. GEM port nepodporuje zašifrovanie ak end to end je vytvorený. V Profile režime je nutné nastavovať manuálne. Je potrebné nastaviť dynamic bandwith allocation (DBA) profil, line profil, service profil a alarm profil.[8]

3.2.1 DBA a linkový profil

DBA profil definuje parametre prenášania XPON a vie byť viazaný k T-CONT na dynamické priradenie šírky pásma a zlepšenia využitia nahrávania. Konfigurácia DBA profilu prebieha pomocou príkazu dba-profile add , kde po add sa píše meno DBA profilu, typ profilu a požadovanú rýchlosť užívateľom. Line profil slúži na pridanie koncovej jednotke. AKo prvé sa použije príkaz ont-lineprofile gpon na pridanie line profilu a vstupu do line profil režimu. Použije sa tcont na zviazanie T-CONT a DBA profil. Ďalej sa použije príkaz gem add na nastavenie prepojenia medzi GEM a T-CONT , ak sa toto nevykoná koncová jednotka nevie vykonávať služby. Ďalej sa nastavý mappovanie medzi GEM portom a koncovou jednotkou s príkazom gem mapping. Nakoniec sa použije commit aby aby nastavenie nadobudlo účinosť.[8]

Výpis 3.1: Príklad konfigurácie DBA a linkového profilu

```
huawei(config)#dba-profile add profile-name (meno) (typ)
max (rýchlosť)
huawei(config)#ont-lineprofile gpon profile-id
(id line profilu)
huawei(config-gpon-lineprofile-5)#tcont (id tcon)
dba-profile-id (id DBA profilu)
huawei(config-gpon-lineprofile-5)#gem add (GEM port id)
eth tcont
(id tcon)
huawei(config-gpon-lineprofile-5)#gem mapping
(GEM port id) 0 eth (id ethernetu)
huawei(config-gpon-lineprofile-5)#commit
huawei(config-gpon-lineprofile-5)#quit
```

3.2.2 Servysný a alarmový profil

Servisný profil poskytuje kanál pre konfiguráciu koncovej jednotky. Ako prvé sa použije ont-srvprofile gpon na vytvorenie servisneho profilu a na vstup do service profile režimu. ďalej sa použije ont-port eth na konfiguráciu možností portov koncovej jednotky. Nakoniec sa dá príkaz commit na ukončenie konfigurácie. Alarm profil slúži na monitorovanie výkonu aktivovanej koncovej jedntoky. Nastavuje sa pomocou príkazu gpon alarm-profile add[8].

Výpis 3.2: Príklad konfigurácie servisného profilu

```
huawei(config)#ont-srvprofile gpon profile-id (id profilu)
huawei(config-gpon-srvprofile-5)#ont-port eth adaptive
huawei(config-gpon-srvprofile-5)#commit
huawei(config-gpon-srvprofile-5)#quit
```

3.2.3 Pridanie GPON koncovej jednotky

Ako prvé sa použije príkaz interface gpon na vstup do GPON režimu. Koncová jednotka sa môže pridať automaticky, manuálne. Na manuálne pripojenie koncovej jednotke sa používa príkaz ont add spolus s číslom portu, identifikačné číslo ONU, Sn-auth spolu su seriovým číslom ONU, omci na prenos správ medzi OLT a ONU, ont-line-profil-id na priradenie linkového profilu podla mena a ont-srvprofile-id na priradenie servisneho profilu podla mena. Na automatickú detekciu sa použije port x ont-auto-find kde x udáva čislo portu. S príkazom ont ipconfig sa konfiguruje ip adressa koncovej jednotky. Out port native-vlan konfiguruje prednastavenú VLAN pre daný port[8].

Výpis 3.3: Príklad pridania koncovej jednotky

```
huawei(config)#interface gpon 0/2
huawei(config-if-gpon-0/2)#port 0 ont-auto-find enable
huawei(config-if-gpon-0/2)#quit
```

3.2.4 Vytvorenie servisného portu

Servisný port vytvára prepojenie medzi stranou uživatela a strany siete. Bez servisného portu neni poskytovaná služba. Ako prvé je potrebné vytvoriť traffic profile s príkazom traffic table ip ktorá určuje rýchlosť a počet používatelov ktorý sa môžu pripojiť. Potom treba vytvoriť servisný port s service-port ku ktorej sa pripojí traffic profile, pridá sa GEM port index a priradí sa VLAN ID[8].

Výpis 3.4: Príklad konfigurácie servisného portu

```
huawei(config)#traffic table ip index (id) cir (rýchlsoť)
priority (id služby) priority-policy
local-Setting
huawei(config)#service-port (id portu) vlan (VLAN id) gpon
(port) ont (id ont) gemport (index GEM portu) inbound
traffic-table index (id traffic profile) outbound
traffic-table index (id traffic profile)
```

3.2.5 Vytvorenie alarmového profilu

Alarmový profil slúži na nastavenie prahovej hodnoty alarmu na monitorovanie výkonu aktívnej ONT linky. AK sa prahová hodnota prekročí tak sa vyšle alarmové hlásenie.[8].

Výpis 3.5: Príklad konfigurácie alarmového profilu

```
huawei-OLT(config)#gpon alarm-profile add profile-name x
<cr>|profile-name<K>
Command:
gpon alarm-profile add profile-name (meno)
Press Q or q to quit input
>GEM port loss of packets threshold (0~100)[0]:
>GEM port misinserted packets threshold (0~100)[0]:
>GEM port impaired blocks threshold (0~100)[0]:
>Ethernet FCS errors threshold (0~100)[0]:
>Ethernet excessive collision count threshold (0~100)[0]:
>Ethernet late collision count threshold (0~100)[0]:
>Too long Ethernet frames threshold (0~100)[0]:
>Ethernet buffer (Rx) overflows threshold (0~100)[0]:
>Ethernet buffer (Tx) overflows threshold (0~100)[0]:
...
```

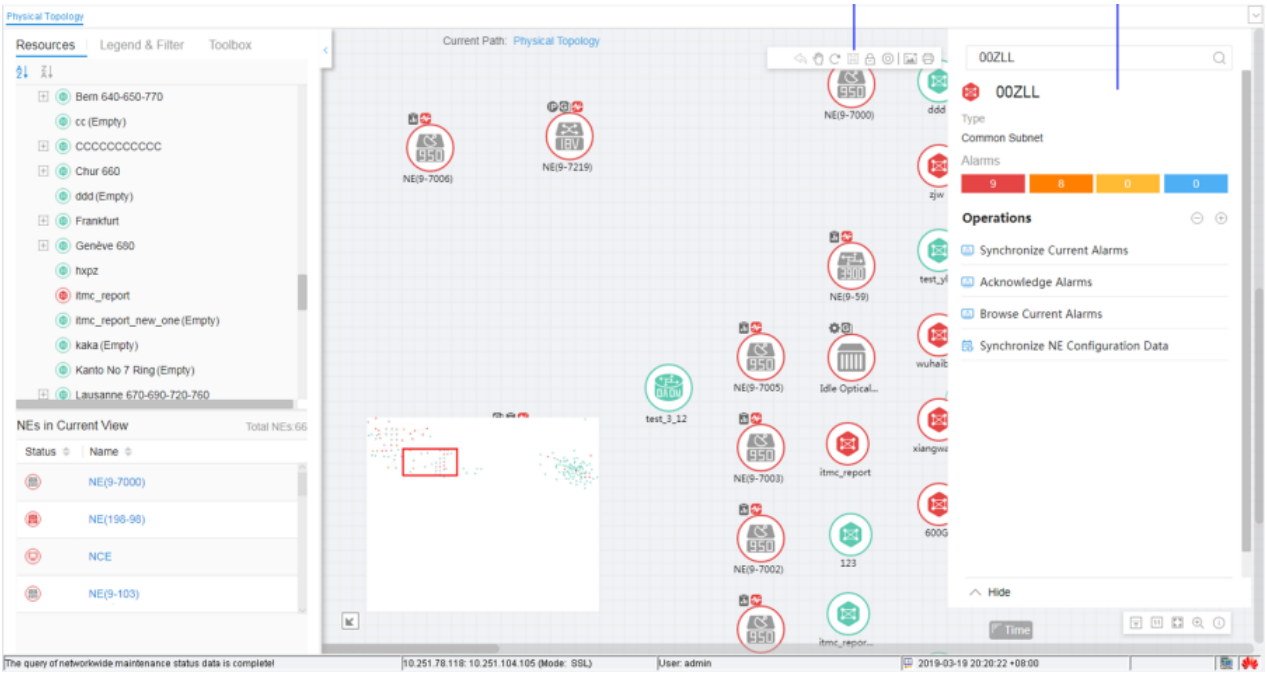
3.3 Konfigurácia pomocou GUI

Konfigurácia pomocou grafického rozhrania je zjednodušená oproti konfigurácie pomocou príkazového riadka. Namiesto potreby písania jednotlivých príkazov a dávania pozor na syntax a správnu lokáciu zadávania príkazu v systéme, grafické rozhranie to rieši v pozadí za užívateľa. Napríklad namiesto potreby vypisovania celého príkazu na konfiguráciu DBA profilu by v grafickom rozhraní by stačilo doplniť meno profilu, vybrať typ a napísať požadovanú hodnotu. Toto uľahčuje prácu s konfiguráciou systémov. Grafické rozhranie môže byť program alebo webová stránka.

3.4 Network Cloud Engine

Network Cloud Engine (NCE) je grafické rozhranie pre konfigurácie siete od spoločnosti Huawei. NCE integruje funkcie riadenie siete, kontrolu služby a analýzu siete. Je systémom pre združovanie sieťových zdrojov, automatizáciu sieťových pripojení a O&M automatizácia. NCE implementuje centralizované riadenie, kontrolu

a analýzu globálnych sietí. Umožňuje celoživotnú automatizáciu siete a inteligentnú správu uzavrenej slučky riadená analýzou údajov. NCE je považovaný za inteligentný mozog celej siete, efektívne prepája fyzickej siete s obchodným zámerom. Medzi základné funkcie NCE patria spravovanie bezpečnosti NCE pomocou bezpečnostných predpisov a zoznamu aktivít užívateľov, spravovanie sieťových prvkov po úspešnom pripojení, zobrazenie sieťových prvkov a ich prepojenia, monitorovanie výkonu siete a mnoho ďalších [9].



Obr. 3.1: Zobrazenie fyzického zapojenia v NCE

4 Programovanie funkčností GUI

4.1 Výber programovacieho prostredia

Zvolené programovacie prostredie je IntelliJ IDEA, ktoré využíva programovací jazyk Java. Jazyk Java umožňuje programu fungovať bez ohľadu na operačnom systéme na ktorom bol spustený. IDEA ponúka aj možnosť využitia závislostí Spring WEB a Thymeleaf ktoré umožnujú prácu s webovými prehliadačmi. Využitá je aj knižnica JSch ktorá programu umožní priamo pracovať s protokolom SSH čo je potrebné aby sa grafické rozhranie vedelo pripojiť na riadiacu jednotku[13] .

4.1.1 Spring WEB

Spring WEB je závislosť ktorá umožňuje programu pracovať s webovým prehliadačom, zobraziť daný program na webovej stránke a umožňuje posielanie údajov medzi kódom v Jave a hodnotami na stránke. Spring WEB bol vybraný kvôli odstránenia nutnosti vytvorenia vlastnej graficky pre program vďaka možnosti využitia HTML a zobrazenia na webovej stránke[10] .

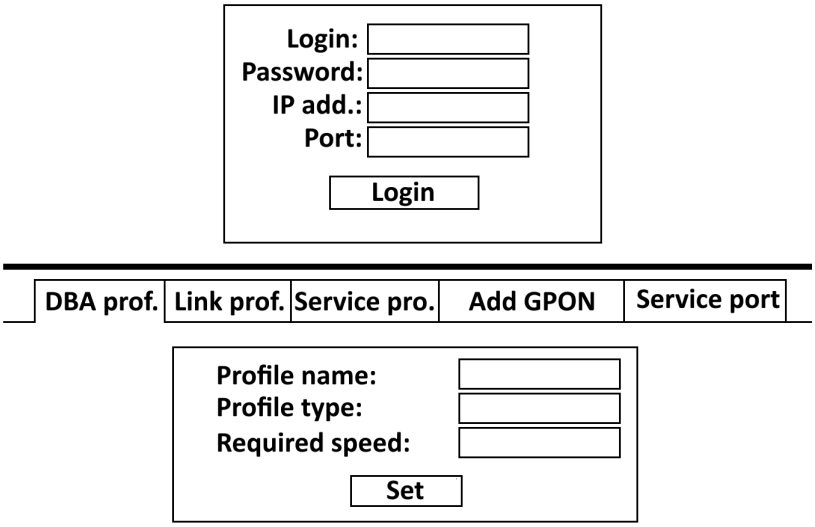
4.1.2 Thymeleaf

Thymeleaf je závislosť ktorá vylepšuje možnosti prepojenia a práce medzi Java a HTML. Umožňuje napríklad primať údaje z textových polí, umožňuje využitie podmienok if else v HTML kóde, umožňuje primať hodnoty z rádiových tlačidiel alebo zo začiarkovacích políčok. Thymeleaf bol vybraný kvôli zjednodušeniu práci s HTML a kvôli dodaniu užitočných funkcií ktoré vylepšia program[12] .

4.2 Teoretický návrh GUI

Hlavnou častou programu je prepojenie programu s riadiacou jednotkou, toto sa dosiahne pomocou knižnice JSch pomocou ktorej sa údaje budú spracovávať v pozadí bez toho aby ich užívateľ videl. Pre úspešné prepojenie je potrebné Jave odovzdať login, heslo, adresu na ktorej sa zariadenie nachádza a číslo portu na ktorom pracuje. Tieto údaje užívateľ zadá na prvú stránku programu a následne ich odošle pomocou tlačidla. Ak sú zadané údaje nesprávne tak stánka vypíše chybu inak ukáže prvú stranu konfigurácie. Stánka konfigurácie ponuka možnosť medzi prepnutím jednotlivých častí na konfigurácie, výpis udalostí, zobrazenie statusu pripojených riadiacich jednotiek alebo odhlásenie. Na konfiguračných stránkach je potrebné zadať len premenné časti príkazu, so statickými časťami sa údaje spoja v pozadí pred poslaním

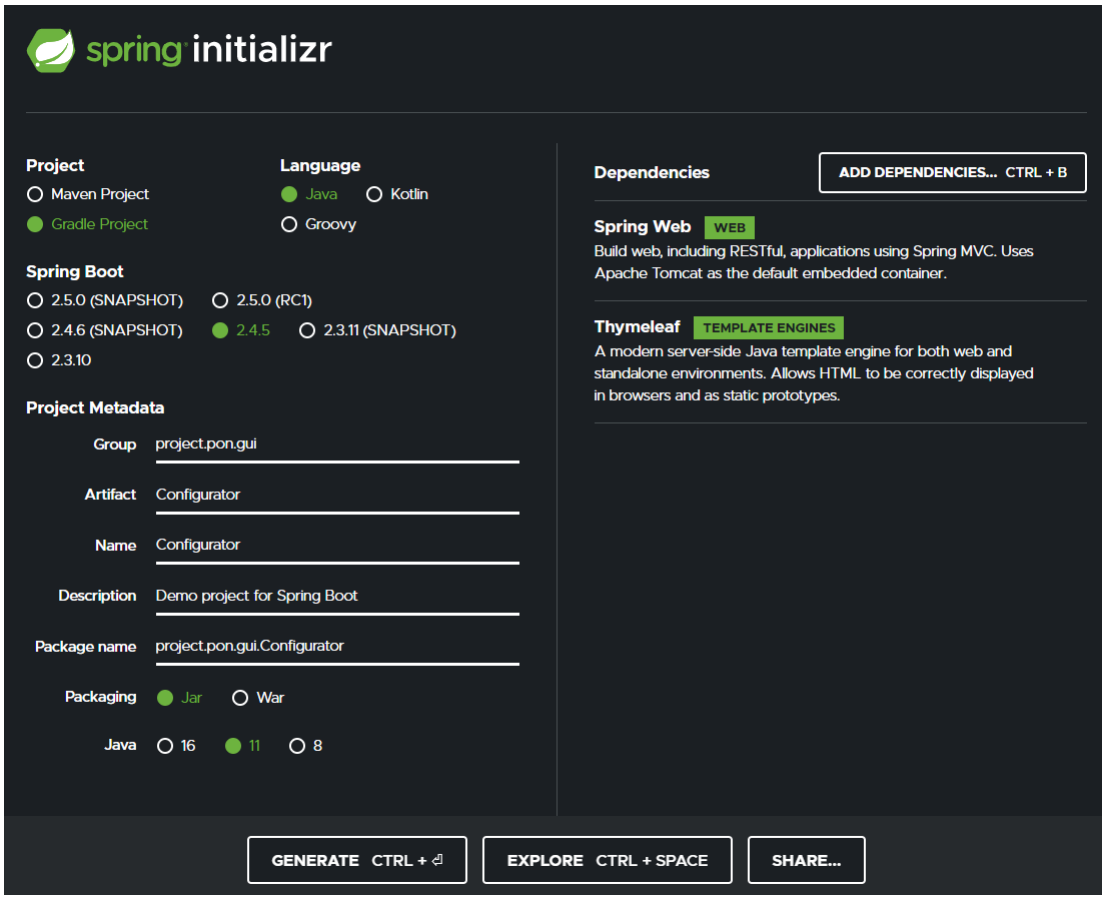
na riadiacu jednotku spoja. V prípade problému s údajmi ktoré sú poslané na konfiguráciu stránka vypíše chybové hlásenie. Každá stránka je vlastná metóda v Jave prepojená s HTML súborom ktorý ju graficky zobrazí.



Obr. 4.1: Návrh prihlasovania a konfigurácie v GUI aplikácií

4.3 Inicializácia kódu a pridanie závislostí

Pre pridanie závislostí do Javy bol použitý spring initializr os spoločnosti VMware. Vybral sa projekt Gradle, jazyk Java, Spring Boot 2.4.5, packaging sa nastavil na Jar a verzia Javy sa nastavila na 11. V poličku dependencies sa pridali Spring WEB a Thymeleaf. Vyberie sa vhodné meno skupiny a meno projektu a na koniec sa stlačí tlačidlo GENERATE[**?**].



Obr. 4.2: Nastavenie pre vygenerovanie kódu[?]

4.4 Pridanie knižnici

Pre pridanie knižnici je potrebné stiahnuť danú knižnicu JSch v formáte .jar a premiestniť ju do priečinku programu do priečinka libs. Po premiestnení je treba otvoriť súbor build.gradle a v sekcií dependenies pridať compile group: 'com.jcraft', name: 'jsch', version: '0.1.55'. Po uložený zmieň sa zobrazí ikona slona ktorá po kliknutí naň aktualizuje kód s novou knižnicou.

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    compile group: 'com.jcraft', name: 'jsch', version: '0.1.55'

}
```

Obr. 4.3: Pridanie knižnici JSch

4.5 Programovanie prihlásenia

Na vytvorenie stránky pre prihlásenie je potrebné vytvoriť metódu na prepojenie premenných použitých v kóde písanom v Jave so súborom HTML ktorý obsahuje kód stránky prihlásenia. Nato slúži metóda @GetMapping("/adresa"), ktorá danú metódu priradí na zadanú adresu. Metóda bude napočúval na adresu localhost:8080/[zadaná adresa]. Metóda je typu string ktoré je viac symbolová textová premenná z dôvodu, že metóda vracia meno HTML súboru. V tele metódy sa inicializuje globálna premenná POST ktorá má za úlohy inicializovať premenná polia v HTML a uchovávať údaje z HTML. Ďalej v tele metódy sa nachádza funkcia Model ktorá pomocou .addAttribute(meno premennej v html,premenná) inicializuje premenné. Nakoniec metóda vráti meno HTML súboru.

Výpis 4.1: Kód pre metódu pre prihlásenie

```
    @GetMapping("/index")
    public String index(Model model){
        Post post = new Post();
        firstRun = true;
        model.addAttribute("post",post);
        return "index";
    }
```

V súboru HTML sa pomocou tagu <form> vytvorili štyri textové polia na zadanie údajov a jedno tlačítko na odovzdanie zadaných údajov. V tagu <form> je treba medzi m a > pridať th:action, th:object a method. Th:action určuje na akú stránku sa po odoslaní údajov presmeruje užívateľ, th:object sa priradí premenná z Javy a method sa nastavý akým spôsobom form má pracovať.

Výpis 4.2: Príklad kódu HTML pre prihlásenie

```
<form action="#" th:action="@{/DBA}"
th:object="${post}" method="post">
    <table class="center">
    <tr>
        <th>Port:</th>
        <th class = "input"><input type="text"
        th:field="*{post}" /></th>
    </tr>
    <tr>
        <td><input type = "submit" value = "Prihlasiť">
        </td>
    </tr>
    </table>
</form>
```

Tým že po odoslaní údajov je užívateľ automaticky presmerovaný na stránku /DBA je potrebné riešiť spracovanie dát a prihlásenie v metóde pre stránku DBA. Aby sa odlíšilo v metóde DBA či sa jedná o prihlásenie alebo o konfiguráciu sa v metóde Index sa vytvorila boolová premenná nazvaná firstrun nastavená na hodnotu pravda. V metóde DBA sa skontroluje či je firstrun pravda ak áno tak sa pošlú údaje na prihlásenie a nastatý sa na nepravdu.

Výpis 4.3: Príklad kódu v metóde DBA na prihlásenie

```
if(firstRun)
{
    connection = SSH.Connection(send);
        System.out.println(connection);
        firstRun = false;
        ChangeLog.LTTF("Log␣in");
}
```

Vyvolá sa trieda SSH s metódou .Connection(textová premená). Pokus o prihlásenie je ošetrený proti chybám pomocou try catch funkciou, ktorá testuje či sa prihlásenie podarilo a podmienkou regex, ktorá testuje či údaje sú v správnom formáte. Regex je naprogramovyný tak aby kontroloval či prvé tri údaje sú oddelené čiarkami a či štvrtá hodnota je čisto číselná. Toto je dokázané s funkciou .matches(".*,.*,.*,
d+") kde .* znamená ľubovolný symbol a d+ označuje viaceré číselné hodnoty za sebou. Tým že údaje z HTML sú spojené do jedného reťazca oddelené s čiarkou je

nutné daný údaj rozdeliť, využije sa metóda .split(symbol kde rozdeliť,koľko krát rozdeliť). Rozdelené údaje sa uložia do pola textových premenných. Vytvorí sa premená Session kde sa priradí nové pripojenie pomocou new().getSession(login,adresa,port), tým že hodnota portu vo formátu textu a príkaz očakáva číselnú premennú je potrebné port previesť na číslo pomocou Integer.parseInt(port). Po vytvorený session sa pomocou .setPassword(heslo) nastatý heslo a pomocou .connect() nastane pokus o pripojenie. Ak sa pripojenie podarí tak sa vráti hodnota YES a uživatel môže ísť na konfiguráciu, ak sa pripojenie nepodarilo tak stránka vyhodí chybové hlásenie a uživatel musí znova zadať prihlasovacie údaje.

Výpis 4.4: Kód pre pokus o prihlásenie pomocou SSH

```
try
{
    keepalive = data;
    if(data.matches(".*,.*,.*,\\d+"))
    {
        String dp [] = data.split(",",0);
        System.out.println("TEST:␣+++++++␣" + data);
        session = new JSch().getSession(dp[0], dp[2],
        Integer.parseInt(dp[3]));
        session.setPassword(dp[1]);
        session.setConfig("StrictHostKeyChecking", "no");
        session.connect();
        return "YES";
    }
    else
    {
     return "NO";
    }
}
catch (com.jcraft.jsch.JSchException e)
{
    return "NO";
}
```

4.6 Programovanie stránok pre konfiguráciu

Štruktúra kódu pre jednotlivé metódy a stránky pre konfiguráciu sú rovnaké s rozdielmi len v premennej na identifikáciu, vracanej stránky a počtu údajov ktoré sa

odosielajú. Z tohoto dôvodu sa vysvetlí len postup pri metóde pre DBA. Oproti metóde na prihlásenie je u konfiguračných metódach potreba pracovať s získanými údajmi sa použije @RequestMapping namiesto @GetMapping. V @RequestMapping(value, method) value údáva meno adresy a method udáva spôsob získania údajov. Údaje z HTML sú uložené v premenej post a získajú sa pomocou funkcie .getPost() ktorá vracia textovú hodnotu. Pred poslaným konfiguračných hodnôt sa skontroluje či údaje nie sú prázdne a či sú vo správnom formáte. Ak sú tak sa pomocou metódy .CommandSender(typ,údaj) z triedy SSH sa pošlú na riadiacu jednotku. Ak údaje boli nesprávne tak sa vypíše chyba na stránke.

Výpis 4.5: Príklad kódu pre konfiguráciu

```
String send = post.getPost();
String test = null;
Post dba = new Post();
model.addAttribute("post",dba);
    if(send != null)
    {
        test = Checker.checker("DBA",send);
        if(test == "Pass")
        {
            SSH.CommandSender("DBA",send);
            ChangeLog.LTTF("DBA change");
        }
    }

if(send != null)
{
    System.out.println(test);
    model.addAttribute("test",test);

}
model.addAttribute("con",connection);
```

4.7 Programovanie kontroly údajov

Pre kontrolu údajov sa používa trieda Checker s metódou checker. Metóda podla zadaného identifikátora funkcia switch vyberie o akú časť konfigurácie sa jedná a kontroluje údaje podľa toho. Pre kontrolu sa využíva funkcia match() ktorá kontroluje či dané údaje sú čisto číselné, toto sa dosiahne pomocou výrazu `"^[0-9]*$"`. Ak

sú údaje správne tak metóda vráti hodnotu Pass a údaje budú poslané na odoslanie opačnom prípade sa vráti hodnota Fail a na stránke sa vypíše chybové hlásenie. Ďalej sú kontrolované aj chyby zo strany riadiacej jednotky pomocou triedy ErrorFromOLT a metódy FindError(). Metóda kontroluje odpoveď od riadiacej jednotky na text "the error locates at ’^". Táto chyba nastane napríklad ak je v textových poliach zadaná medzera medzi znakmi čo riadiaca jednotka berie ako ďalšiu časť príkazu. Ak táto chyba nastane tak sa na stránke vypíše.

Výpis 4.6: Príklad kódu na kontrolu hodnôt

```
String part [] = test.split(",",0);
String result = null;
    if (part[2].matches("^[0-9]*$" ))
    {
    result = "Pass";
    }
    else
    {
    result = "Fail";
    }
```

4.8 Programovanie posielania príkazov na riadiacu jednotku

Pre posielanie príkazov na riadiacu jednotku je potrebné vytvoriť kanál na posielanie príkazov pomocou funkcie channel ktorá pomocou funkcie .setCommand(údaje) priradí príkaz a pomocou .connect() sa príkaz pošle.

Výpis 4.7: Príklad kódu na posielanie údajov z HTML

```
channel = (ChannelExec) session.openChannel("exec");
channel.setCommand("enable\nconfig\n");
channel.setCommand("enable\nconfig\n"
+CommandCompiler.CC(type,com));
channel.connect();
```

Tým že údaje sú len dynamické časti celého príkazu je potrebné celý príkaz spojiť toto je dosiahnuté s triedou CommandCompiler.CC(typ,údaj) ktorá má za úlohu dynamické a statické časti príkazu spojiť. CC využije switch ktorý podla zadaného identifikátora z metódy (typ) rozhodne o aký príkaz sa jedná. Príkaz sa vytvorí pomocou spojenie textu medzi úvodzovkami a premennými údajmi do jednej textovej premennej. Tým že je potrebné všetky príkazy posielať naraz je potrebné

každý príkaz spojiť do jednej premennej a to pomocou += ktorá pridá text na koniec premenej a \n ktorá oddelí od seba príkazy v texte aby ich riadiaca jednotka mohlo rozoznať.

Výpis 4.8: Príklad kódu na spojenie príkazu

```
switch(type)
{
    case "GPON":
    result ="interface␣gpon␣" + part[0] + "/"
    + part[1]+ "\n";
    result +="port␣" + part[2]
    + "␣ont-auto-find␣enable␣"+ "\n";
    result +="quit"+ "\n";
    break;
```

Na výpis odpovedi zo strany riadiacej jednotky je použitá funkcia .setOutputStream(responseStream) ktorá do konzoly v programovacom prostredí vypíše odpoveď, toto hlavne slúži k debbugovania a testovaniu.

Výpis 4.9: Príklad kódu na ziskanie odpovedi

```
channel.setOutputStream(responseStream);
channel.connect();
String responseString =
new String(responseStream.toByteArray());
System.out.println(responseString);
```

4.9 Programovanie odhlásenia

Metóda pre odhlásenie je tvorená podobne ako metóda pre prihlásenie tým že nie je potreba pracovať s údajmi je použitá @GetMapping(adresa). v tele metódy sa nachádza len metóda CloseConnection() z triedy SSH, ktorá má za úlohu odpojiť užívateľa.

Výpis 4.10: Kód na metódu pre odhlásanie z SSH

```
@GetMapping("/LogOff")
public String logoff() throws IOException {
    ChangeLog.LTTF("Log␣Off");
    SSH.CloseConnection();
    return "LogOff";
}
```

Metóda CloseConnection kontroluje či session a channel sú otvorené, ak sú otvorené tak sa pomocou funkcie .disconnect() sa uzatvoria.

Výpis 4.11: Kód na odhlásanie z SSH

```
if (session != null)
    {
        session.disconnect();
    }
if (channel != null)
    {
        channel.disconnect();
    }
```

4.10 Programovanie zápisu udalostí

Zápis udalostí na sieti je riešený pomocou vyvolania funkcie LTTF(text) z triedy ChangeLog, ktorá má za úlohu zapísať a vypísať udalosti. Tým že je potrebné mať zápis udalostí od najmladšej k najstaršej je potrebne alebo čítať súbor opačným smerom z dola hore alebo zapisovať do súboru opačne. Čítanie súboru opačne vyžaduje knižnicu a väčší počet kódu je jednoduchšie riešenie zapisovať udalosti opačne. Opačný zápis je dosiahnutý využitím dvoch textových súborov dočasný a trvalý. Ako prvé sa vytvorí dočasný súbor do ktorého sa nová udalosť zapíše, potom sa obsah trvalého obsahu prepíše do dočasného a trvalý sa vyprázdni a skopírujú sa údaje z dočasného doň a nakoniec sa dočasný súbor vymaže.

Výpis 4.12: Príklad kódu metódy pre zápisu udalosti

```
String date = post.getPost();
Post ch = new Post();
if(date != null)
{
    if(LogEditor.DataChecker(date))
    {
        LogEditor.LogChanger();
    }
}
ArrayList<String> logs = new ArrayList<String>();
logs.clear();
logs = new ArrayList<String>(ChangeLog.FileReader());
model.addAttribute("post",ch);
model.addAttribute("changelogs", logs);
```

Pre zapisovanie je potrebné využiť triedu File a FileWriter a PrintWriter. Trieda File vytvorí nové súbory ak neexistujú, FileWriter(x,y) slúži na písanie do súborov kde x udáva meno súboru a y udáva či sa súbor má prepísať ale nový obsah sa pridá na koniec súboru. PrintWriter slúži na vymazanie obsahu súboru bez vymazania samotného súboru. Zápis do súboru nastáva vo while cykly ktorý beží kým nenarazí na koniec súboru pomocou funkcie .hasNextLine().

Výpis 4.13: Príklad kódu zápisu do súboru

```
File myObj = new File("temp.txt");
File Obj = new File("changelog.txt");
FileWriter BW = new FileWriter("temp.txt", true);
FileWriter FW = new FileWriter("changelog.txt", true);
BW.write(formatter.format(date) + "-"
+ ipAddress + "␣:␣" + log + "\n");
try {
    Scanner myReader = new Scanner(Obj);
    while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    BW.write(data + "\n");
    }
    myReader.close();
} catch (FileNotFoundException e) {
    System.out.println("An␣error␣occurred.");
    e.printStackTrace();
}
```

Zápis udalostí nastáva vyvolaním metódy LTTF pri prihlásení, konfigurácií, odhlásení. Udalosť sa zapíše ako čas kedy nastala + IP adresa užívateľa + typ udalosti. Čas sa zapíše vo formáte deň/mesiac/rok hodiny:minuty:sekundy pomocou funkcie SimpleDateFormat("dd/MM/yyyy HH:mm:ss"). Na získanie času je potrebné inicializovať premennú typu DATE pomocou Data date = new Date() a na získanie textovej hodnoty času je použitá funkcia . format(date).

Výpis 4.14: Kód na formátovanie dátumu

```
SimpleDateFormat formatter = new SimpleDateFormat("dd/MM
/yyyy␣HH:mm:ss");
```

Pre získanie IP adresy je použitá trieda IPLocator ktorá pomocou funkcie Finder(), ktorý využíva fakt že odpoveď riadiacej jednotky obsahuje IP adresu užívateľa výpise. Finder prechádza výpis po znakoch a hľadá bodku. Ak bodku nájde tak tes-

tuje či v diaľke 9 znakov od prvej bodky sa nachádzajú dve ďalšie bodky. Ak je podmienka splnená tak sa začne testovať či znaky medzi bodkami sú čiste číselné pomocou funkcie .isDigit() ak je podmienka splnená tak sa ešte otestujú znaky pred prvou bodkou a znaky po poslednej bodke a nastatý sa kde sa nachádza začiatok a koniec IP adresy a nakoniec sa vráti Ip adresa.

Výpis 4.15: Príklad kódu z IP lokátora

```
while(i < test.length() || !found_ip)
{
    if( test.charAt(i) == '.'
    && Character.isDigit(test.charAt(i-1))
    && !found_ip) {
    A[a] = i;
    a++;
    dot_count = 0;
    if(test.length() - i < 9)
    {
        ol = test.length() - i;
    }
    for (int x = 1; x < ol; x++) {
    if (test.charAt(i + x) == '.')
    {
        A[a] = i + x;
        a++;
        dot_count += 1;
        System.out.println(dot_count);
    }
}
```

4.11 Programovanie výpisu udalostí

Výpis slúži funkcia FileReader z triedy ChangeLog ktorý prečíta súbor zapísaných udalostí a vracia ho ako pole textových údajov. Súbor sa číta po riadkoch vo while cykly a pomocou .add sa pridáva do pola typu ArrayList. Vrátené pole sa odobzdá do HTML kde pomocou cyklu th:each sa postupne celé pole vypíše.

Výpis 4.16: Kód na výpis udalostí v HTML

```
<table class="center">
    <tr th:each="changelog␣:␣${changelogs}">
        <td th:text="${changelog}">
    </tr>
    <br/>
</table>
```

4.12 Programovanie mazania starých udalostí

Stránka výpisu udalostí ponúka aj možnosť mazania starých udalostí. Predvolené nastavenie pre mazanie je jeden mesiac. Ak užívateľ chce zmeniť dobu mazania tak musí zadať číselnú hodnotu a písmeno d(deň),m(mesiac) alebo y(rok). Ak užívateľ zadá hodnotu tak sa pomocou metody DataChecker() z triedy LogEditor kontroluje či počet je väčší ako nula a menší a menší ako maximálny počet dní/mesiacov v roku a či zadané písmeno je d,m alebo y.

Výpis 4.17: Príklad kódu na kontrolu dátumu

```
switch(time)
{
    case "d":
        if(numb > 0 && numb < 32)
        {
            saveData(data);
            return true;
        }
        else
        {
            return false;
        }
}
```

Ak údaje sú správne tak pomocou metódy .LogChanger() začne mazanie. Pre mazanie je potrebné vypočítať najstarší dátum ktorý sa nemaže pre toto slúži metóda dataCalculator(). DataCalculator() vytvorý aktuálny dátum a pomocou switchu ktorý pozerá zadané písmeno vyberie správnu výpočtový vzorec. Napríklad ak bolo zadané d pre deň, tak sa testuje či aktuálny deň v mesiaci je väčší ako zadané číslo. Ak nie je tak sa od aktuálneho dňa odpočíta zadaná hodnota. Ak je väčšia

tak sa od posledného dňa odpočíta zadaný deň a pripočíta deň v mesiaci a mesiac sa zníži o jednotku. Po kalkulácií dátumu sa daný dátum vráti do LogChanger().

Výpis 4.18: Príklad kódu výpočtu dátumu

```
switch(wd[1])
{
     case "d":
     if(m == 2)
     {
          if(d < md)
          {
               d = 28 - (md - d);
               m = m - 1;
          }
               else
          {
               d = d - md;
          }
     }
}
```

LogChanger() prečíta súbor po riadkoch vo while cykluse a vyčíta si z nich prvých 10 znakov čo je dátum bez hodín, minút a sekúnd. Daný dátum s prevedie na Date premennú s funkciou form.parse(). Ak je dátum zo súboru mladší ako dátum mazania tak sa riadok zapíše do pola. ak sa nájde starší dátum tak sa cyklus ukončí a nové pole sa uloží do súboru. Tým že ArrayList nedokáže mazať v cykloch je jediná možnosť správne údaje ukladať do nového pola.

Výpis 4.19: Príklad kódu na mazanie udalostí

```
tDate = form.parse(data);
if(tDate.after(dDate))
{
     newList.add(data);
}
else
{
     last = true;
}
```

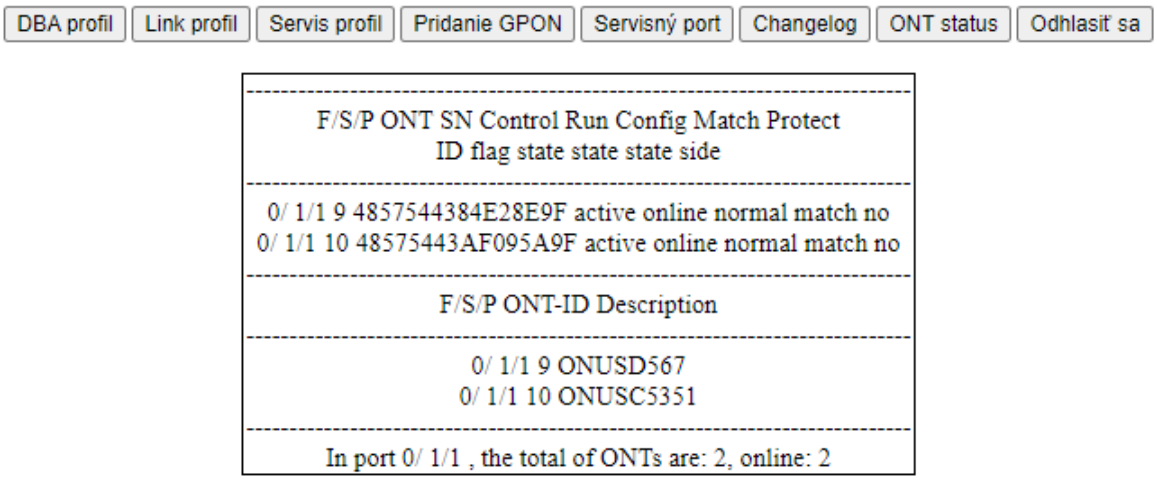
4.13 Programovanie výpisu statusu ONT

Výpis statusu sa nachádza na adrese /Display a tým že nepracuje s údajmi stačí použiť @GetMapping(). Pre získanie statusu sa pošle príkaz display ont info 0 all a odpoveď od riadiacej jednotky sa uloží do súboru a v cykly sa prevedie do pola. Tento krok je nutné spraviť pre lepšie formátovania na stránke, ak by sa odovzdala odpoveď do HTML tak text bude rozhodený po celej stránke.

Výpis 4.20: Kód metódy pre zobrazenie statusu koncových zariadený

```
@GetMapping("/Display")
public String Display(Model model) throws Exception {
    SSH.CommandSender("Display","");
    model.addAttribute("posts",
    ONT_Display.ONT_Status());
    return "Display";
}
```

Tým že odpoveď od riadiacej jednotky obsahuje nielen výpis statusu ale celý príkazový riadok je potrebné vyhľadať len potrebné časti. Toto je dosiahnuté pomocou cyklu ktorý číta súbor výpisu a hľadá riadok ktorý obsahuje display ont info a ukončí sa ak nájde riadok ktorý obsahuje (config)#. Či riadok obsahuje tieto časti je získané pomocou funkcie .contains(). Riadky medzi display ont info a (config)# sa zapíše do poľa a pošle sa do HTML a ako u ChangeLog sa pomocou th:each postupne vypíše výstup.



Obr. 4.4: Výpis statusu koncových zariadený

4.13.1 Programovanie konfigurácie Alarmu

Pre konfiguráciu alarmu je využitá metóda SHELLSENDER v triede SSH. Metóda CommandSender ktorá je využitá pri predošlých konfiguráciách preskočí požiadavky od riadiacej jednotky na zadanie hodnoty pre nastavenie jednotlivé časti alarmu a pri výpise odpovedi sa program dostane do nekonečného cyklu čakania na koniec odpovedi. Metóda SHELLSENDER toto rieši zmenou .openChannel(exec) na shell a čítaním výstupu po častiach v cyklu.

Výpis 4.21: Príklad kódu pre konfiguráciu alarmu

```
Channel channel=session.openChannel("shell");
while(x < 10)
{
    while(in.available()>0)
    {
        int i=in.read(tmp, 0, 1024);
        if(i<0)break;
        data.add(new String(tmp,0,i));
    }
    if(channel.isClosed())
    {
        System.out.println("exit-status:"
        +channel.getExitStatus());
        break;
    }
    try{Thread.sleep(100);}catch(Exception ee){}
    x++;
}
```

4.14 Test funkčnosti grafického rozhrania

Testovanie funkčnosti grafického rozhrania sa prebiehalo cez internet pomocou VPN. Ako prvé sa testovalo pripojenie na riadiacu jednotku, kde po zadaní prihlasovacích údajov testovala spätná väzba z riadiacej jednotky. Po prihlásení sa testovali konfiguračné stránky a ich príkazy, sledoval sa výpis z riadiacej jednotky či nastala chyba a ak nie či sa dané zmeny nastali napríklad či po odoslaní príkazu enable config sa užívateľ prepol z EA5800-X2> na EA5800-X2(config)#. Pri testovaní výpisu udalostí sa po každej konfigurácií kontroloval súbor changelog.txt či nastali zmeny a či zmeny sú správnom poradí. Opačný zápis a mazanie udalostí sa testoval v samostatnom

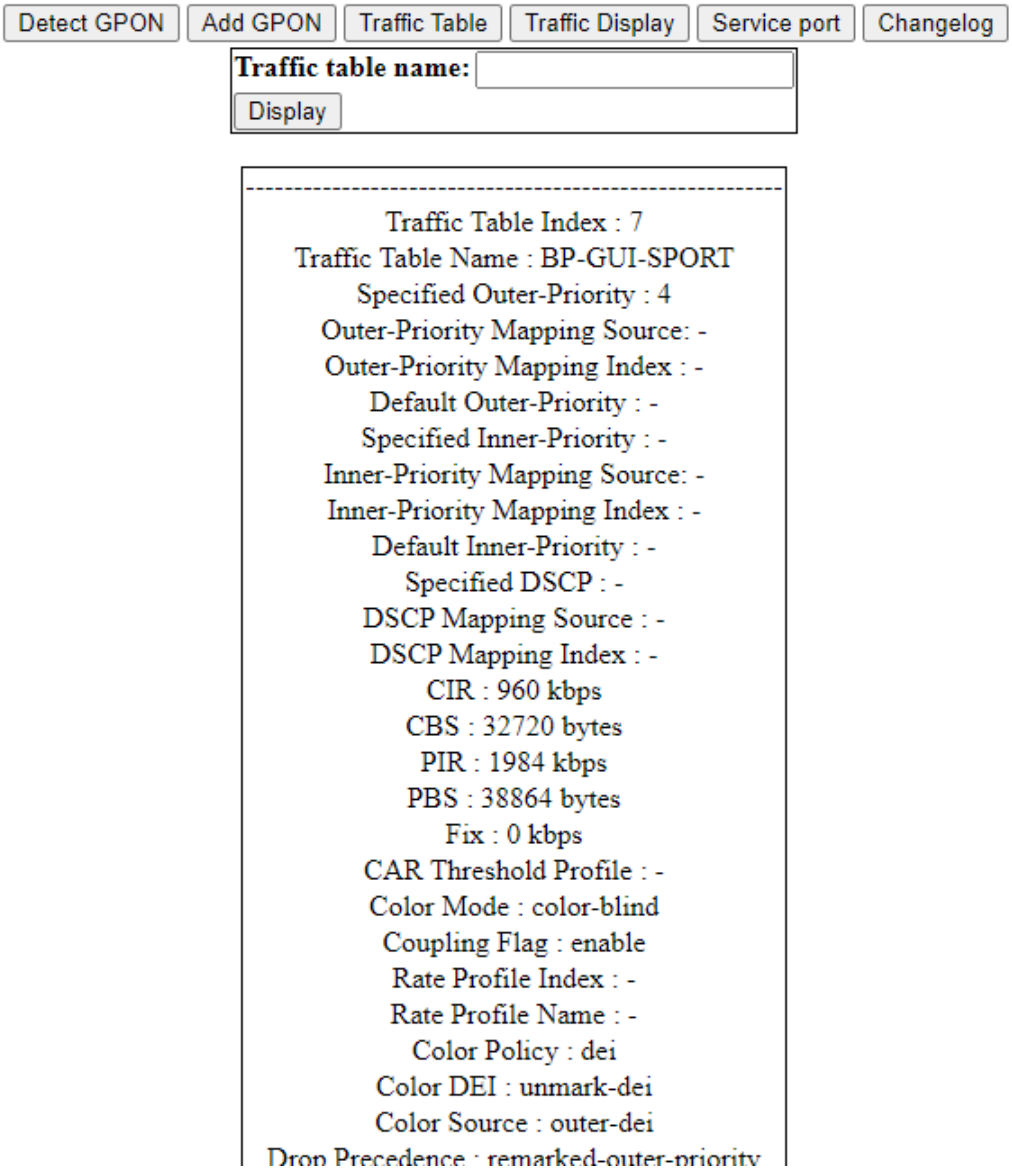
programe ktorý po overení správnosti bol pridaný do grafického rozhrania. Konfigurácia alarmu a výpis zadaného alarmu bol testovaný pomocou výpisu odpovedi od riadiacej jednotke a kontroloval sa pomocou programu Putty ktorá sa pripojila na riadiacu jednotku a kontrolovala či nastavený alarm v GUI existuje pomocou Putty a opačne. Testovanie CommandCompilera a Checker bolo rovnaké sledoval sa výpis z konzoly na chybové hlásenia.

4.14.1 Vytvorenie Traffic table, Alarm profilu a výpis udalostí

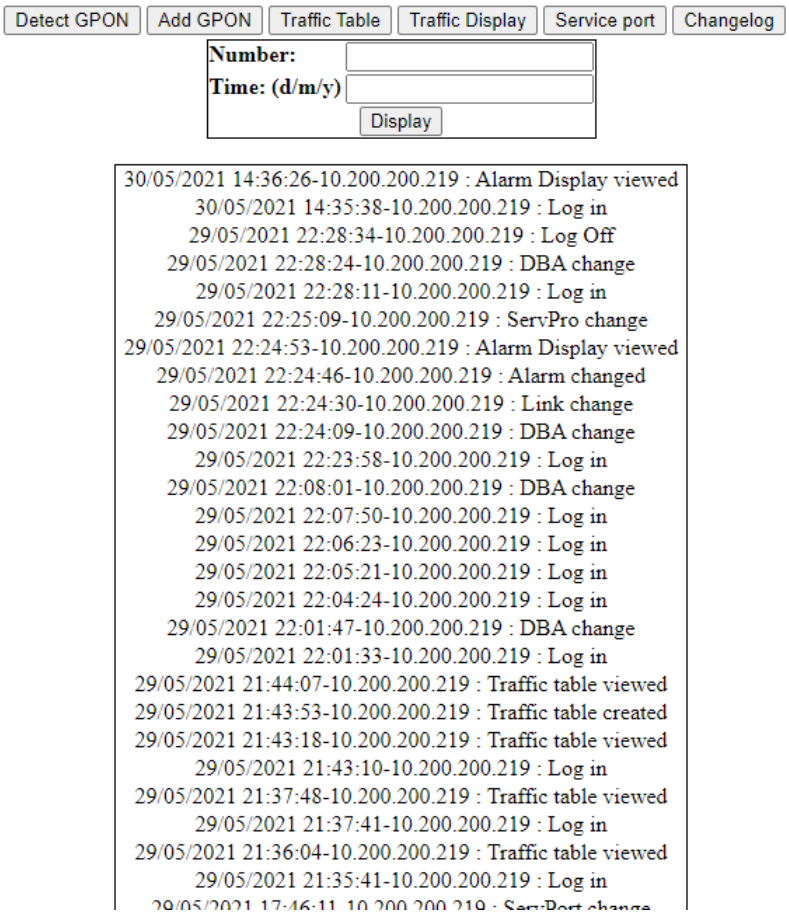
Po úspešnom naprogramovaní jednotlivých častí programu sa vytvorila traffic tabula a Alarm profile.Po vyplnení údajov sa vyskúšal či sa správne vytvorili a skontrolovali sa.

4.14.2 Pridanie ONT pomocou grafického rozhrania

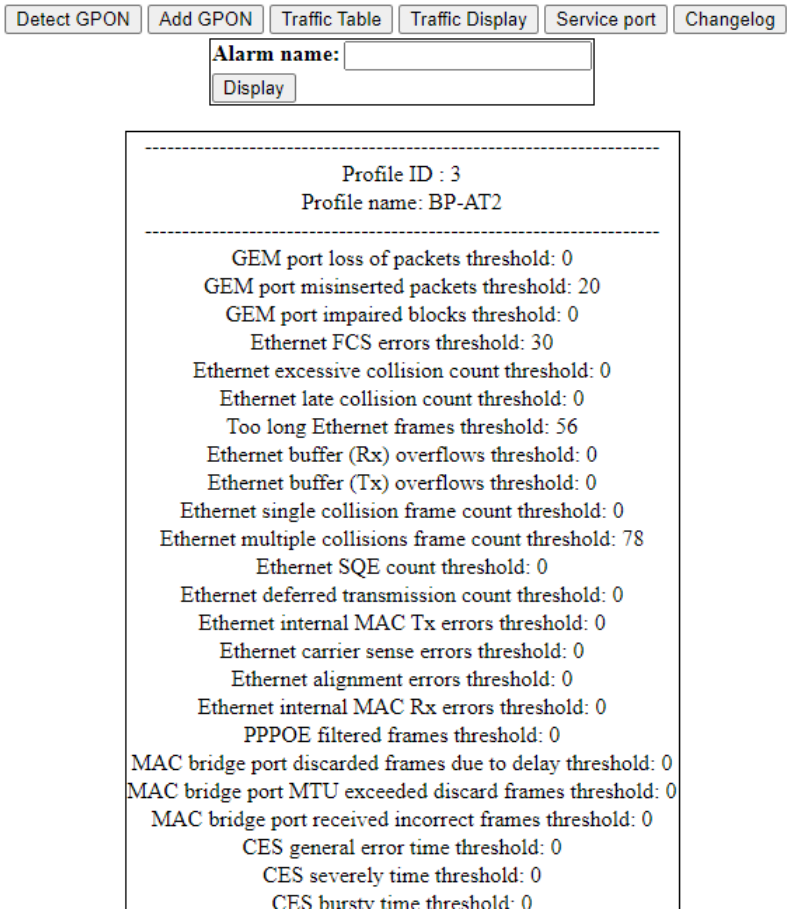
Po overení správnosti konfigurácie a výpisu programu, sa pridalo koncové zariadenie a skontrolovalo sa či sa zobrazí na stránke statusu koncových zariadení. Zariadeniu bola pridaná poznámka BP-GUI-xkovac52 aby sa dalo ľahko nájsť a overiť či sa pridalo.



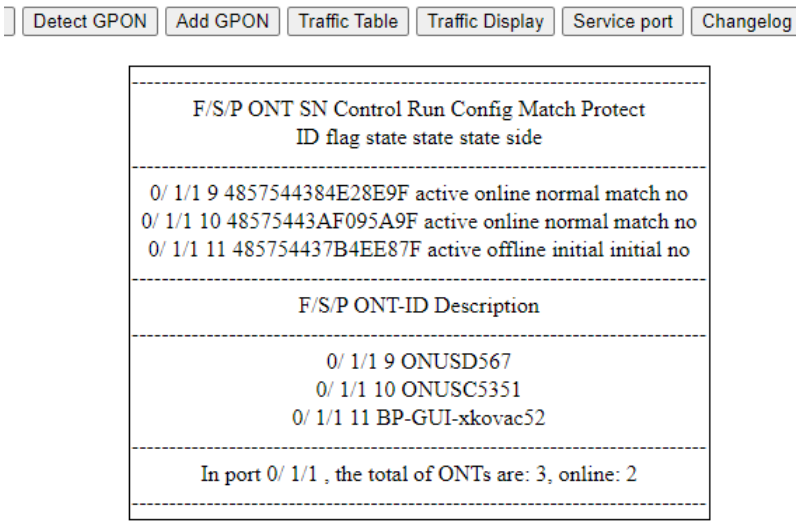
Obr. 4.5: Úspešné vytvorenie traffic table a jeho výpis na webovej stránke



Obr. 4.6: Výpis udalostí na webovej stránke



Obr. 4.7: Úspešné vytvorenie alarm profilu a jeho výpis na webovej stránke



Obr. 4.8: Úspešné pridanie ONT a výpis pripojených ONT

5 Manuál pre Grafické Rozhranie

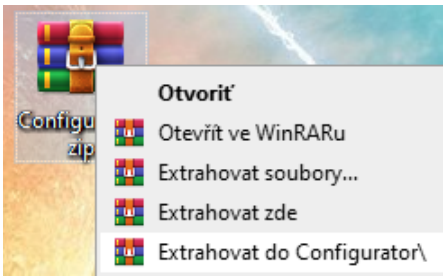
5.1 Potrebné programy

Pre spustenie Grafického rozhrania sú potrebné nasledovné programy:

- IntelliJ IDEA Community Edition, dostupný zo stránky jetbrains.com/idea/
- Ľubovolný webový prehliadač, napríklad Mozila Firefox
- Zip súbor Configurator.zip

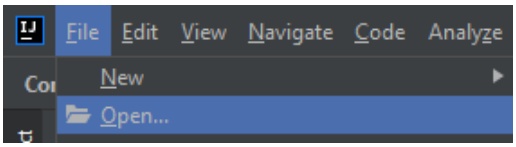
5.2 Pridanie Configuratora do IDEA

Ako prvé je potrebné na Configurátor.zip kliknúť s pravím tlačidlom myši a zip súbor rozbaliť na ľahko nájditelné miesto napríklad pracovnú plochu.



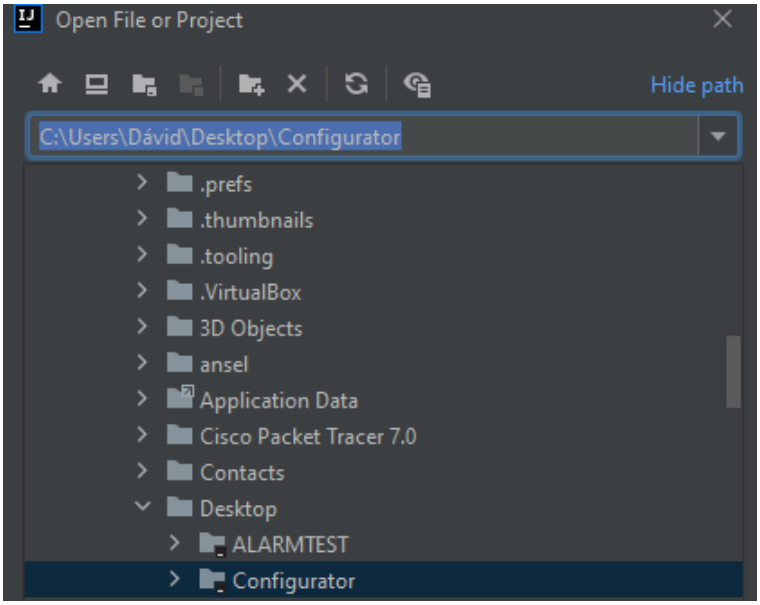
Obr. 5.1: Extrahovanie programu

Ďalej je potreba v IDEA kliknúť na tlačidlo File v ľavom hornom rohu a kliknúť na Open.



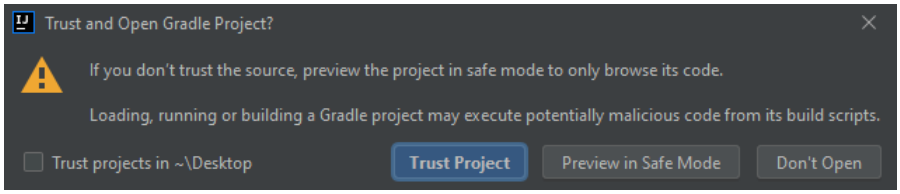
Obr. 5.2: Pridanie programu do IDEA

Po kliknutí vyskočí okienko a je potrebné zadať umiestnenie priečinka Configurator a dať tlačidlo OK.



Obr. 5.3: Hľadanie umiestenia

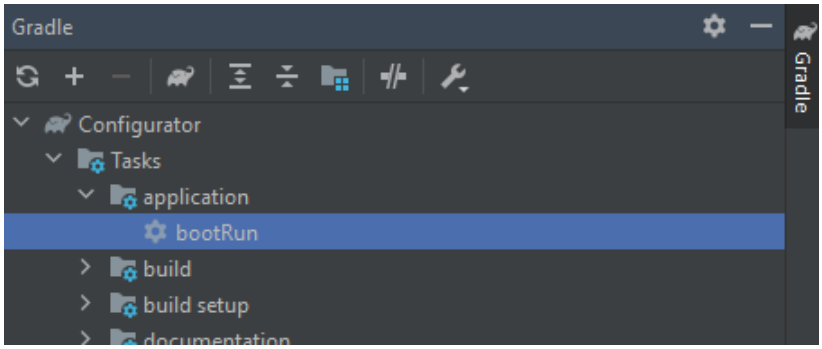
Po kliknutí OK vyskočí okienko že projekt je z neznámeho zdroja a je potrebné kliknúť na tlačidlo Trust Project.



Obr. 5.4: Hlásanie neznámeho zdroja

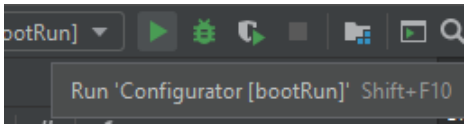
5.3 Spustenie

Pri prvom spustení je treba kliknúť na tlačidlo Gradle potom application a spustenie bootRun.



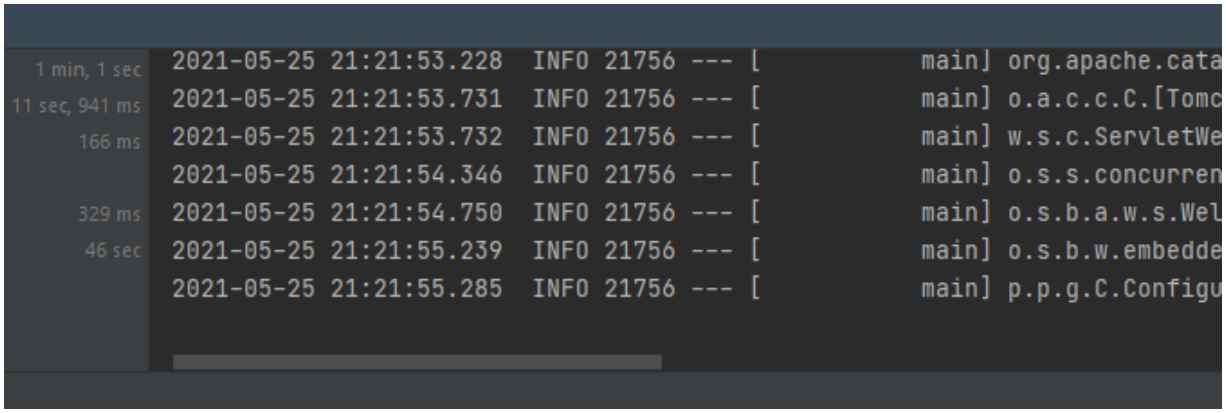
Obr. 5.5: Prvé spustenie

Pri nasledujúcich spusteniach stačí kliknúť na zelenú šípku v pravom hornom rohu.



Obr. 5.6: Spustenie programu

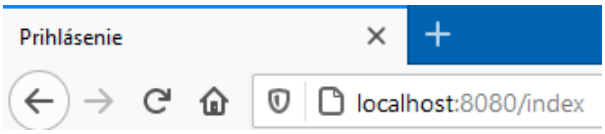
Po spustení je treba počkať kým konzola dokončí výpis INFO.



Obr. 5.7: Výpis po spustení predmetu

5.4 Pripojenie na riadiacu jednotku

Ako prvé je treba zapnúť webový prehliadač a do adresovej lišty zadať localhost:8080/index.



Obr. 5.8: Zadanie adresy stránky

5.5 Návod pre konfigurované hodnoty

5.5.1 Prihlásenie

Login a Password sú údaje potrebné na pripojenie k OLT. IP address je adresa na ktorej OLT pracuje, uznáva IP adresy aj mená domény. Port je port na ktorom OLT pracuje zvyčajne 22 alebo 2222, port uznáva len číselné hodnoty.

5.5.2 DBA profil

Profile name je meno požadovaného DBA profilu ktorý sa vytvára. Profile type je číselná hodnota od 1 do 5. Required speed je čiste číselná hodnota od 128 do 10000000 ktorá určuje požadovanú rýchlosť.

5.5.3 Link profil

Line profile name je meno požadovaného Link profilu ktorý sa vytvára. ID TCONT je číselná hodnota od 0 do 127. DBA profile name je vytvoreného DBA profilu. ID GEM port je číselná hodnota od 0 do 1023. Mapping ID je číselná hodnota od 0 do 7. Ethernet ID je číslo od 1 do 24.

5.5.4 Servis profil

Profile name je meno požadovaného profilu ktorý sa vytvára.

5.5.5 GPON

GPON interface je rozhranie na ktorom sa ONT jednotka nachádza a je zadávaná vo formáte x/y. Port ID je číselná hodnota od 0 do 15. ONT ID je číselná hodnota od 0 do 255. Sn-auth je hodnota dĺžky 12 až 16 znakov. ONT line profile je meno ONT profilu. ONT service profile je meno nakonfigurovaného servis profilu. Description je voliteľná a nie je nutné ju vyplňovať. Static IP address, Network Mask, Gateway, Primary DNS sú vo formáte a.b.c.d. Vlan je číselná hodnota od 0 do 4096. ONT port ID je číselná hodnota od 1 do 24.

5.5.6 Traffic table

Traffic table name je meno pre požadovanú tabuľku ktorá sa vytvára. CIR je číselná hodnota od 0 do 10240000. PIR je číselná hodnota od 64 do 10240000. Priority je číselná hodnota od 0 do 7 .

5.5.7 Service port

Service port ID je číselná hodnota od 0 do 32767. Table ID je ID vytvorenej traffic table a je to číslo od 0 do 1024. Vlan ID je číselná hodnota od 1 do 4093. GPON interface je rozhranie na ktorom sa ONT nachádza, zadávaná vo formáte x/y/z. ONT ID je číselná hodnota od 0 do 255. Gemport je číselná hodnota od 0 do 1023. Užívateľské Vlan ID je číselná hodnota od 1 do 4095.

5.5.8 Changelog

Number je číselná hodnota od 1 do 31. Time: (d/m/y) je nastavenie modifikátora kde d je deň, m je mesiac a y je rok.

5.5.9 Alarm

Požadované hodnoty pre konfiguráciu alarmu sa nachádza na stránke konfigurácie.

Závěr

Táto práca sa zaoberala s vytvorením grafického prostredia pre konfiguráciu, zápis udalostí, výpis statusu koncových jednotiek a nastavenie alarmu riadiacej jednotky v pasívnych optických sietiach. Grafické rozhranie bolo naprogramované v jazyku Java so závislosťami Spring Web, Thymeleaf a knižnicou JSch. Ku grafickému rozhraniu sa pripája pomocou webového prehliadača zadaním adresy localhost:8080/index kde sa s požadovanými údajmi dá prihlásiť na SSH server. Pri testovaní grafického rozhrania sa podarilo pripojiť na SSH server pomocou zadaných údajov. Konfigurácií jednotlivých častí boli plne funkčné a vytvorenie profilov bolo funkčné. Výpis udalostí, statusov koncových jednotiek, alarmov bol úspešný a zobrazovali sa na jednotlivých stránkach. Konfigurácia alarmu bola overená pomocou programu Putty a konfigurácia bola úspešná. Na koniec sa pomocou grafického rozhrania sa nakonfigurovalo koncové zariadenie s DBA profilom BP-GUI-DBA, linkovým profilom BP-GUI-LINK, servis profilom BP-GUI-SPRO, servis portom BP-GUI-SPORT a poznámka na ONT bola nastavená ako BP-GUI-xkovac52.

Literatúra

[1] *Fiber Optic Solutions: Introduction to Passive Optical Network (PON) [online]. [cit. 2020-12-04].* Dostupné z: <http://www.fiber-optic-solutions.com/intro-optical-network-pon.html>

[2] *Passive Optical Network (PON) [online]. [cit. 2020-12-04].* Dostupné z: <https://www.viavisolutions.com/en-us/passive-optical-network-pon>

[3] *ROUSE, Margaret. Passive optical network (PON) [online]. [cit. 2020-12-04].* Dostupné z: <https://searchnetworking.techtarget.com/definition/passive-optical-network-PON>

[4] *PON Accessories: ABC of PON: Understanding OLT, ONU, ONT and ODN [online]. [cit. 2020-12-04].* Dostupné z: <https://community.fs.com/blog/abc-of-pon-understanding-olt-onu-ont-and-odn.html>

[5] *FTTx – Fiber to the x: Explained [online]. [cit. 2020-12-04].* Dostupné z: <https://www.carritech.com/news/fttx-fiber-to-the-x-explained/>

[6] *GPON Fundamentals [online]. [cit. 2020-12-04]* Dostupné z: <https://sites.google.com/site/amitsciscozone/home/gpon/gpon-fundamentals>

[7] *What are the differences between GPON, XG-PON and XGS-PON?: Specification Differences Between 10G GPON and GPON [online]. [cit. 2020-12-04].* Dostupné z: <https://medium.com/@ivyhtfuture/what-are-the-differences-between-gpon-xg-pon-and-xgs-pon-eddbd6576b7d>

[8] *Configuring the GPON Internet Access Service (Profile Mode): Configuring a GPON ONT Profile [online]. [cit. 2020-12-04].* Dostupné z: <https://support.huawei.com/view/contentview!getFileStream.action?mid=SUPE_DOC&viewNid=EDOC0100587985&nid=EDOC0100587985&partNo=j009&type=htm#cfg_cli_2609>

[9] *Huawei NCE: Network Cloud Engine [online]. [cit. 2020-12-04].* Dostupné z: <https://support.huawei.com/enterprise/en/network-management-and-analysis-software/network-cloud-engine-pid-22752497>

[10] *Spring.io: Spring WEB [online]. [cit. 2020-12-04].* Dostupné z: <https://spring.io/guides/gs/serving-web-content/>

[11] *ITU: Gigabit-capable passive optical networks (GPON) [online]. [cit. 2020-12-04].* Dostupné z: <https://www.itu.int/rec/T-REC-G.984.7-201007-I/en>

[12] *Thymeleaf [online]. [cit. 2020-12-04].* Dostupné z: <https://www.thymeleaf.org/>

[13] *JETBRAINS. : IntelliJ IDEA [online]. [cit. 2020-12-04].* Dostupné z: <https://www.jetbrains.com/idea/>

[14] *ITU: 10-Gigabit-capable passive optical networks (XG-PON): Transmission convergence (TC) layer specification [online]. [cit. 2020-12-04].* Dostupné z: <https://www.itu.int/rec/T-REC-G.987.3-201401-I/en>

[15] *Cale, A. Salihovic and M. Ivekovic, "Gigabit Passive Optical Network - GPON,"2007 29th International Conference on Information Technology Interfaces, Cavtat, 2007, pp. 679-684, doi: 10.1109/ITI.2007.4283853. [online]. [cit. 2020-12-04].* Dostupné z: <https://ieeexplore.ieee.org/document/4283853>

[16] *M. Kumari, R. Sharma and A. Sheetal, "Passive Optical Network Evolution to Next Generation Passive Optical Network: A Review,"2018 6th Edition of International Conference on Wireless Networks & Embedded Systems (WECON), Rajpura (near Chandigarh), India, 2018, pp. 102-107, doi: 10.1109/WECON.2018.8782066. [online]. [cit. 2020-12-04].* Dostupné z: <https://ieeexplore.ieee.org/document/8782066>

[17] *Java [online]. [cit. 2020-12-04].* Dostupné z: <https://www.java.com/en/>

[18] *KOVÁČ, Dávid. GRAPHIC USER INTERFACE FOR PASIVE OPTICAL NETWORK CONFIGURATION. In: EEICT. EEICT_2021_sbornik. 2021, s. 3. ISBN 978-80-214-5942-7.*

Zoznam symbolov, veličín a skratiek

PON	Pasive Optical Network
OLT	Optical Line Terminal
ONU	Optical Network Unit
ODN	Optical distribution network
FTTx	Fiber to the x
FTTP	Fiber to the Premises
FTTH	Fiber to the Home
FTTC/N	Fiber to the Cabinet/Node
WDM	Wavelenght Division Multiplex
TDM	TimeDivision Multiplexor)
GPON	Gigabit-capable Passive Optical Network
XG-PON	Označované aj ako 10G-PON
CLI	Command Line Interface
GUI	Graphic User Inter-face
SSH	Secure Shell
HTML	Hypertext Markup Language