

Seminarska naloga pri predmetu računalništvo

Riptide - Univerzalno orodje za konfiguracijo omrežij

Mentor: Marko Kastelic

Avtor: Jurij Fortuna, G 3. a

Ljubljana, 5. junij 2023

Povzetek

Naloga opisuje razvojni proces in delovanje programa Riptide. Riptide je odprtokodna programska oprema, razvita v jeziku Java, namenjena nastavitvi omrežnih naprav. Na začetku naloge je podan kratek pregled dveh glavnih komponent programske opreme: uporabniškega vmesnika (frontend) in programskega vmesnika aplikacij (API). V nadaljevanju dokument opisuje "Shede", vtičnike za interakcijo z omrežno opremo. Naloga se zaključi s primerom razvoja Sheda in obravnava izzive, povezane z njim.

Ključne besede: omrežja, omrežne naprave, nastavitve omrežij, API, odprtokodna programska oprema

Abstract

The purpose of this paper is to report on the development process and inner workings of Riptide. Riptide is an open-source software developed in Java for configuring network devices. It addresses the issue of using multiple software solutions from different manufacturers to configure networks. Initially, the document gives a brief overview of two main components of the software: the frontend and the Application Programming Interface (API). Following that, the document describes "Sheds," plugins for interacting with network equipment. The paper concludes by documenting the development of a custom Shed and discusses the associated challenges that can arise during the development.

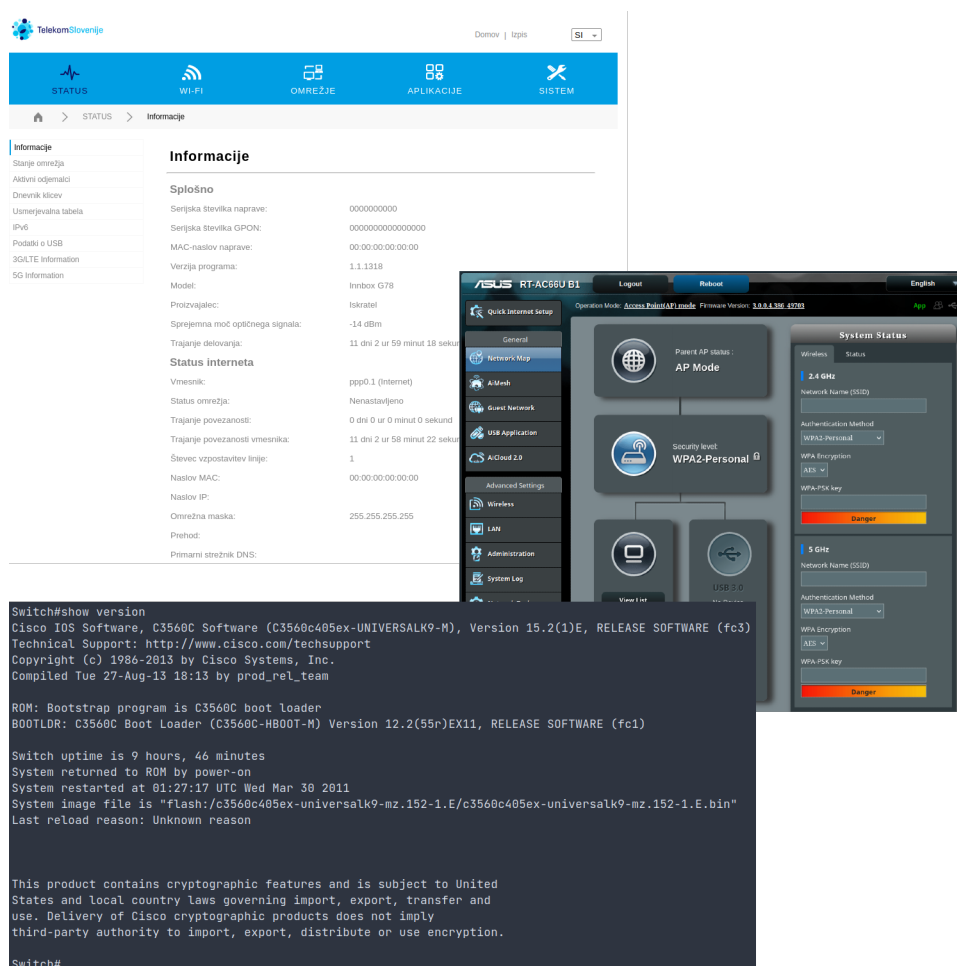
Keywords: networking, network devices, network setup, API, open-source software

Kazalo

1	Uvod	4
2	Uporabljene tehnologije	5
2.1	Ospredni del	5
2.2	Aplikacijski programski vmesnik	5
2.3	Shedi	5
3	Ospredni del	6
3.1	Uporabniški vmesnik	6
3.2	Konfiguracija	6
3.3	Upravljanje poverilnic	7
3.4	Rokovanje z napravami	7
3.4.1	Nalaganje Shedov	7
3.4.2	Nalaganje naprav	7
4	Aplikacijski programski vmesnik	8
4.1	Dostop do API-ja	8
4.2	Struktura API-ja	8
4.3	Vgrajena orodja	8
4.4	Komunikacija z napravami	8
4.5	Čarovniki	10
5	Shedi	11
5.1	Priprava projekta	11
5.2	Struktura projekta	12
5.2.1	Metapodatki	12
5.2.2	Gonilni razredi	13
5.2.3	Konfiguracijske strani	13
5.3	Izgradnja	13
6	Eksperimentalne funkcije	14
6.1	Topološki pogled omrežja	14
6.2	Integracija z Luo	14
7	Zaključek	15
8	Viri in literatura	16

1 Uvod

Ideja za razvoj Riptide-a se je pojavila, ko sem bil med konfiguracijo domačega omrežja prisiljen uporabljati tri vrste programske opreme, različnih proizvajalcev. Med sabo so se nemalo razlikovale in so bile po večini nestabilne. Rezultat naloge je programska oprema Riptide. Deluje na principu “vtičnikov” (t.i. Shedov) za posamezne kose omrežne opreme, kar uporabnikom omogoča, da iz enega programa nastavljajo celotno svoje omrežje. Programska oprema je odprtokodna in brezplačna, kar omogoča večjo dostopnost uporabnikom.



Slika 1: Primeri uporabljene programske opreme

2 Uporabljene tehnologije

Tako Shedi, kot tudi Riptide so napisani v Javi. Java omogoča dinamično nalaaganje modulov ob zagonu navideznega stroja, tudi iz zapakiranih JAR datotek. Aplikacijo sestavljajo trije deli: Ospredni del, API in pa Shedi. Med sabo so to ločeni deli in razvoj poteka na vsakem posamezno. V nadaljevanju naloge bo vsak bolj podrobneje opisan.

2.1 Ospredni del

Ospredni del je del, s katerim uporabnik upravlja. Skrbi za prikaz podatkov in vnašanje konfiguracije. Skrbi tudi za shranjevanje konfiguracije in upravljanje z uporabnikovimi podatki.

2.2 Aplikacijski programski vmesnik

Aplikacijski programski vmesnik (v nadaljevanju API) deluje kot most med osprednim delom in Shedi. Z njim lahko preko vmesnikov ospredni del podatke pošilja Shedom.

2.3 Shedi

Shedi so vtičniki, oziroma gonilniki, ki skrbijo za komunikacijo z napravami. Praviloma jih razvija skupnost in so odprtokodni. Distribuirani in nameščeni pa na uporabnikov sistem, kot JAR datoteke.



Slika 2: Diagram delovanja

3 Ospredni del

Ospredni del je odgovoren za konfiguracijo Riptide-a ter rokovanje z uporabnikovimi poverilnicami in napravami.

3.1 Uporabniški vmesnik

Za izgradnjo uporabniškega vmesnika sem uporabil knjižnico JavaFX. Omogoča hitro izdelavo in načrtovanje, skupaj s knjižnico FXML. Za stil pa sem uporabil knjižnico AtlantaFX, ki naredi vmesnik minimalističen in vključuje štiri že narejene stilske datoteke.

Glavno okno za konfiguracijo naprav je zasnovano kot MDI (ang. multiple-document interface). Glavna prednost MDI-ja je možnost rabe večih podoken znotraj glavnega okna, kar uporabniku omogoča večopravnost.

3.2 Konfiguracija

Uporabnik lahko svoje naprave shrani, in si s tem prihrani čas za morebitno kasnejšo konfiguracijo. Poleg tega Riptide omogoča spremembo barve vmesnika po uporabnikovi želji.

Okoljske spremenljivke, naprave in poverilnice so zapisane v objektu imenovanem `Workspace`.

```
@Data
public class Workspace implements Serializable {
    private Theme theme;
    private ArrayList<Credential> credentials;
    private ArrayList<Connection> connections;

    public Workspace() {
        theme = Theme.PRIMER_DARK;
        credentials = new ArrayList<>();
        connections = new ArrayList<>();
    }

    public Workspace(Theme theme, ArrayList<Credential>
        credentials, ArrayList<Connection> connections) {
```

```

        this.theme = theme;
        this.credentials = credentials;
        this.connections = connections;
    }
}

```

Ko uporabnik nastavitve shrani je objekt serializiran in zapisan v datoteko na uporabnikov sistem. Datotek je lahko več, kar omogoča prilagoditev vmesnika za različna okolja (npr. posebej za domačo in poslovno rabo). Workspace datoteke so shranjene v JSON formatu, kar uporabnikom omogoča enostavno izmenjavo ali prenos.

3.3 Upravljanje poverilnic

Za konfiguracijo večine omrežnih naprav je potrebna avtorizacija. Riptide uporabnikom omogoča varno shranjevanje poverilnic na njihovem sistemskem keyringu. Poverilnice so ob povezavi na napravo prek API-ja podani Shedu, kar pomeni, da se razvijalci teh ne rabijo ukvarjati z varnim hranjenjem poverilnic.

3.4 Rokovanje z napravami

3.4.1 Nalaganje Shedov

Nalaganje shedov poteka v dveh korakih, lociranje Sheda in branje metapodatkov. Na uporabnikovem sistemu se nahajajo v obliki JAR datotek v dveh mapah, `~/.config/Riptide/sheds` na *NIX sistemih in `%UserProfile%\Riptide\sheds` na Windows sistemih.

3.4.2 Nalaganje naprav

Ob izbiri naprave, Riptide iz Shedovih metapodatkov najprej prebere njen t.i. model path, ki izgleda približno tako: `telekomslovenije.innboxg78.ts`. Niz je sestavljen iz ID-ja Sheda, ID-ja modela naprave in ID-ja različice modela. Vsi ID-ji so edinstveni, kar pomeni, da lahko program najde Shed, v njem poišče model naprave, njegovo različico in vzpostavi povezavo z napravo prek API-ja.

4 Aplikacijski programski vmesnik

API omogoča uporabnikom, da ustvarijo lastno podporo za omrežno opremo, ki je programska oprema še ne podpira. To pomeni, da lahko skrbniki omrežij enostavno integrirajo nove naprave v svoje omrežje, ne da bi morali čakati, da prodajalec izda uradno podporo.

4.1 Dostop do API-ja

Dostop do APIja je mogoč preko SDK knjižnice. Nahaja se v Riptide Maven repozitoriju. Ker so Javanski upravitelji paketov med seboj bolj kot ne kompatibilni, je tudi postopek dodajanja knjižnice med njimi zelo podoben. Več o strukturi projekta pa v razdelku 5.2.

4.2 Struktura API-ja

Vsi pomembni vmesniki in razredi za pisanje Shedov se nahajajo v paketu `org.riptide.sdk.sheds`. Paket je deljen na podpakeete za uporabniški vmesnik, tipe naprav, čarovnike in avtentikacijo.

4.3 Vgrajena orodja

API vsebuje uporabna orodja za izdelovanje statusnih strani in formularjev, ki se nahajajo v paketu `org.riptide.sdk.sheds.ui`.

4.4 Komunikacija z napravami

Ospredni del s Shedi komunicira preko vmesnikov. Za primer vzemimo Telekomov usmerjevalnik. Ob izbiri naprave se instancira njen gonilni razred v nov objekt, ki je odgovoren za komunikacijo z napravo (postopek je opisan v razdelku 3.4.2).

Program iz metapodatkov razbere, da naprava tipa router. To pomeni, da mora njen gonilni razred implementirati vmesnik `Router` in posledično `Device`, saj ga `Router` razširja.


```

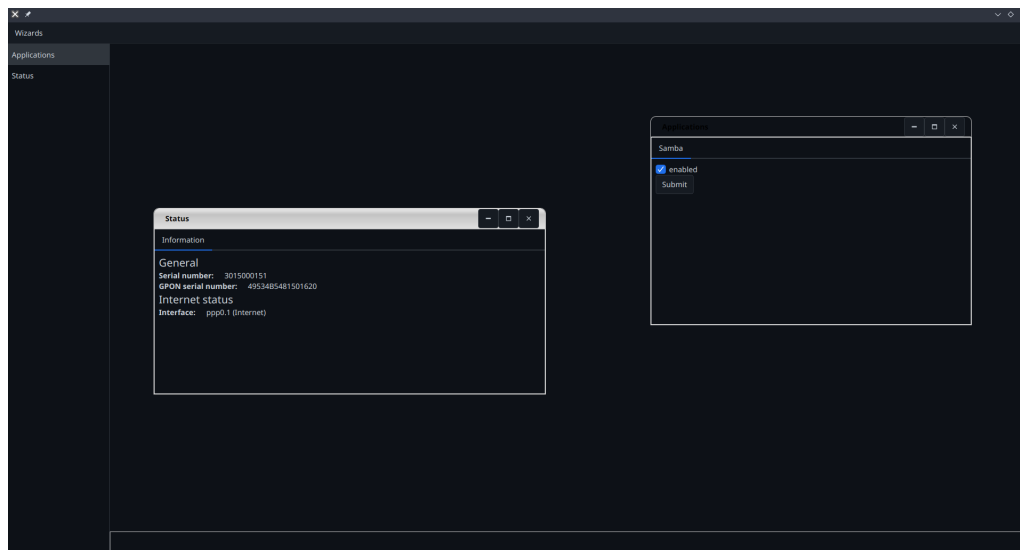
public interface Device {
    void initialize(String address, Credential credential);
    LinkedHashMap<String, Tab[]> getPages();
}

public interface Router extends Device {
    PATWizard patWizard();
}

```

Za tem se kliče metoda `initialize/2`, ki gonilnemu objektu poda omrežni naslov naprave in poverilnice.

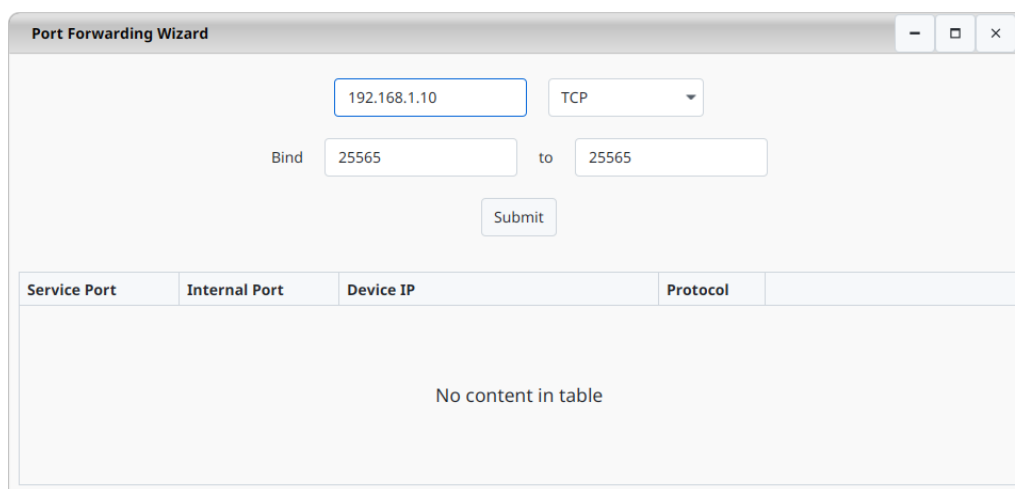
Nato program od gonilnega objekta zahteva seznam konfiguracijskih strani in za-
vihke z metodo `getPages/0` (glej 5.2.3). Ospredni del jih z orodnim razredom
`ContentRenderer` generira, tako kot primer spodaj.



Slika 3: Primer generirane konfiguracijske strani

4.5 Čarovniki

Program tudi manj naprednim uporabnikom omogoča konfiguracijo njihove omrežne opreme. Na konfiguracijski strani za katerokoli omrežno napravo se v menijski vrstici nahaja gumb "Wizards". Pod njem se glede na tip naprave pojavijo različne podmožnosti. Za usmerjevalnike je to zaenkrat *Port forwarding wizard*, ki omogoča uporabniku konfiguracijo NAT-a, za stikala pa *VLAN wizard*, ki omogoča konfiguracijo VLAN-ov.



Service Port	Internal Port	Device IP	Protocol
No content in table			

Slika 4: Čarovnik za konfiguracijo NAT-a

5 Shedi

Shedi so vtičniki, ki skrbijo za komunikacijo z napravami. Sami po sebi niso izvršljivi programi, vendar so le zbirka razredov. Zapakirani so v JAR datoteke, skupaj z datoteko `shed.yml`.

V prihodnosti načrtujem omogočiti razširitev tudi drugih funkcionalnosti. Zaradi tega sem se za poimenovanje vtičnikov naprav odločil uporabiti izraz "shedi" namesto običajnega izraza "plugin".

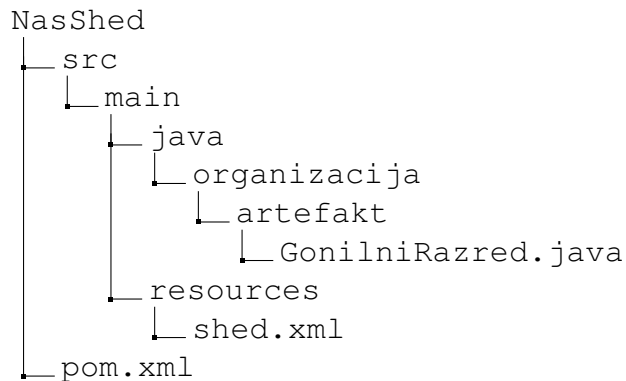
5.1 Priprava projekta

Pri pripravi projekta je bil uporabljen standardni javanski upravitelj paketov *Apache Maven*. Za začetek je potrebno dodati repozitorij in knjižnico. To naredimo v datoteki `pom.xml`.

```
<repositories>
  <repository>
    <id>repsy</id>
    <url>https://repo.repsy.io/mvn/riptide/maven</url>
  </repository>
</repositories>

<dependencies>
  <dependency>
    <groupId>org.riptide</groupId>
    <artifactId>sdk</artifactId>
    <version>VERZIJA</version>
  </dependency>
</dependencies>
```

5.2 Struktura projekta



V privzeti paket (oziroma podpakete) dodamo gonilne razrede za naše naprave. V `resources` mapo pa datoteko `shed.yml`, ki bo vsebovala metapodatke o Shedu.

5.2.1 Metapodatki

Metapodatki vsebujejo pomembne informacije o Shedu. Vključujejo unikatne identifikatorje, ki so nujni za delovanje. Metapodatki so shranjeni v datoteki `shed.yml`.

```
name: "Telekom Slovenije Shed"
id: "telekomslovenije"
version: "1.0"
api_level: "1.0"
authors:
  - "chocoeearly44"
routers:
  - name: "InnboxG78"
    id: "innboxg78"
    flavours:
      - name: "Telekom Slovenije"
        id: "ts"
        handler: "org.riptide.ts.Main"
switches:
```

Zaglavje vsebuje ime Sheda, njegov ID, verzijo, avtorje in podprto verzijo API-ja, za katero je bil zgrajen. Pod zaglavjem sledijo razdelki s tipi naprav. V njih so

shranjene podprte naprave, ki vsebujejo imena, ID-je in vrste. Vrste vsebujejo pot do gonilnih razredov, saj ima naprava lahko različne funkcionalnosti, glede na nameščeno programsko opremo. Tako zagotovimo da lahko isti Shed podpira isto napravo, ne glede na distributerja (za primer InnboxG78 usmerjevalnik, katerega distribuirata tako Telekom Slovenije, kot tudi T2).

5.2.2 Gonilni razredi

Gonilni razredi komunicirajo z napravami. Za pravilno delovanje, morajo implementirati enega izmed v naprej narejenih vmesnikov (npr. Router ali Switch), zaradi dedovanja pa posledično implementirajo tudi `Device` vmesnik. V njih so definirane metode za inicializacijo povezave, ipd. Postopek vzpostavitve povezave pa je opisan v razdelku 4.4.

5.2.3 Konfiguracijske strani

Konfiguracijske strani so v Shedu predstavljene z razredi, ki implementirajo `Tab` vmesnik.

```
public interface Tab {  
    String getTitle();  
    UIComponent rootComponent();  
}
```

Metoda `getTitle()` vrača naslov zavihka, metoda `rootComponent()` pa komponento uporabniškega vmesnika, ki se bo pokazala v zavihku (npr. tabela, kontejner, formular, ...).

Objekti teh zavihkov so nato zbrani v tabelo, katera se preslika v niz s pomočjo razreda `LinkedHashMap`. Taka preslikava bo predstavljala konfiguracijsko stran, ki jo lahko odpremo iz stranske vrstice. Te preslikave nato zberemo in jih vrnemo v metodi `getPages()`.

5.3 Izgradnja

Zadnji korak razvoja Sheda je izgradnja. Ko projekt gradimo, moramo paziti, da v končno JAR datoteko vključimo morebitne knjižnice, ki smo jih uporabili (npr. za komunikacijo preko SSH-ja). Izgrajeno JAR datoteko nato uporabniki naložijo v mapo kjer imajo shranjene svoje Shede (glej 3.4.1).

6 Eksperimentalne funkcije

Riptide je opremljen z nekaj novimi eksperimentalnimi funkcijami, ki pa jih zaradi pomanjkanja časa nisem uspel povsem dokončati in stestirati. Te omogočajo naprednim uporabnikom nove možnosti avtomacije in pregleda nad omrežjem.

6.1 Topološki pogled omrežja

Ta funkcionalnost omogoča naprednim uporabnikom pogled logične topologije omrežij, podoben, kot pri Ciscovem Packet Tracerju. Prikazane naprave lahko uporabniki povežejo med sabo oziroma jih dvokliknejo za hitero konfiguracijo.

6.2 Integracija z Luo

Riptide ima vgrajeno Lua izvajalno okolje. Lua je preprost programski jezik, s katerim lahko napredni uporabniki pišejo skripte. Te skripte omogočajo avtomatizacijo konfiguracije omrežja in s tem prihranijo čas uporabnikom.

7 Zaključek

Dobljeni rezultat predstavlja enostavno in stabilno rešitev za nastavitve omrežij. Z rabo "vtičnikov" (Shedov) omogoča uporabnikom konfiguracijo različne omrežne opreme na enostaven način.

Z uporabo Riptide-a lahko skrbniki omrežij enostavno integrirajo nove naprave v svoje omrežje, ne da bi morali čakati na uradno podporo proizvajalca. S svojim aplikacijskim programskim vmesnikom (API) omogoča razvijalcem ustvarjanje lastne podpore za omrežno opremo, ki je programska oprema še ne podpira.

8 Viri in literatura

lua. *Lua: about*, 2023, spletni naslov: <https://www.lua.org/about.html> (26. 5. 2023)

mkpaz. *AtlantaFX Overview*, 2023, spletni naslov: <https://mkpaz.github.io/atlantafx> (31. 12. 2022)

OpenJFX. *Getting Started with JavaFX*, spletni naslov: <https://openjfx.io/openjfx-docs> (31. 12. 2022)

Oracle. *URLClassLoader (Java SE 20 & JDK 20)*, 1998, spletni naslov: <https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/net/URLClassLoader.html> (3. 3. 2023)

Priloge

Priloga 1

Koda za ospredni del: <https://github.com/riptideconfig/Riptide>

Priloga 2

Koda za SDK: <https://github.com/riptideconfig/SDK>

Priloga 3

Koda za Shed Telekom Slovenije: <https://github.com/riptideconfig/TelekomShed>

Izjava o avtorstvu

Izjavljam, da je seminarska naloga v celoti moje avtorsko delo, ki sem ga izdelal samostojno s pomočjo navedene literature in pod vodstvom mentorja.

5. junij 2023

Jurij Fortuna