VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

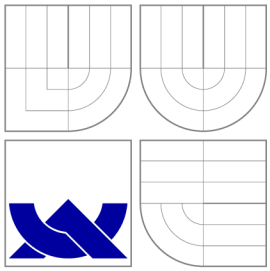
WEBOVÝ FRONTEND K SYSTÉMU PRO ANALÝZU SÍŤOVÉHO PROVOZU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

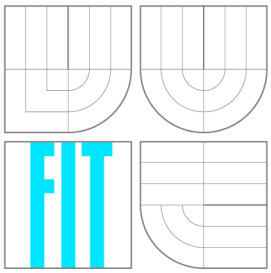
AUTOR PRÁCE
AUTHOR

JIŘÍ KOTÁSEK

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÝ FRONTEND K SYSTÉMU PRO ANALÝZU SÍŤOVÉHO PROVOZU

WEB FRONTEND FOR THE SYSTEM FOR NETWORK TRAFFIC ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ KOTÁSEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV BARTOŠ,

BRNO 2015

Abstrakt

Tato práce popisuje vývoj webového grafického rozhraní, jenž se snaží o zjednodušení práce s programem Supervisor. Ten je řídicím prvkem systému pro analýzu dat Nemea a jeho funkcionalita bude grafickým rozhraním plně zachována. K ověření správnosti a také uživatelské přívětivosti slouží testovací scénáře. Grafické rozhraní je implementováno v JavaScriptu za pomoci knihovny JointJS.

Abstract

This thesis describes development of a web graphical user interface that seeks to simplify work with the Supervisor program. Supervisor is a control element for the network traffic analysis system Nemea and his functionality will be fully maintained. Testing scenarios are used to verify correctness and check that the graphical interface is user-friendly. Graphical iterface is implemented in JavaScript with help of library JointJS

Klíčová slova

monitorování sítí, Nemea, supervisor, webové uživatelské rozhraní, testovací scénáře, JointJS.

Keywords

network monitoring, Nemea, supervisor, web user interface, testing scenarios, JointJS.

Citace

Jiří Kotásek: Webový frontend k systému pro analýzu síťového provozu, bakalářská práce, Brno, FIT VUT v Brně, 2015

Webový frontend k systému pro analýzu síťového provozu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Václava Bartoše

.....
Jiří Kotásek
18. května 2015

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu Václavu Bartošovi za poskytnuté rady. Dále Davidu Alexovi, který byl vždy ochotný a nápomocný.

© Jiří Kotásek, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době jsme obklopeni technologiemi do takové míry, že většina z nás by si jen těžko uměla představit život bez nich. Inteligentní systémy jsou zabudovány do mobilních telefonů, osobních počítačů, televizorů, či dokonce lednic. Aby tato zařízení mohla pracovat dle očekávání uživatelů, téměř vždy spolu musí komunikovat nebo být připojena k internetu. Internet je nejrozšířenější počítačovou sítí a každou hodinu přes něj projde neuvěřitelné množství dat. Na internetu ovšem existují i mnohé hrozby, před kterými je nutno se bránit. Jeden z možných způsobů ochrany před útočníky je monitorování sítí, které nám dovolí zjistit, jak je daná síť vytížená v určitých časech, kdo k síti přistupoval a jaká data na ní stahoval nebo odesílal. Díky těmto datům může správce včas učinit potřebné kroky ke zlepšení stability nebo bezpečnosti sítě.

Napadení sítě však už není pouze doménou počítačových specialistů, nýbrž díky programům, které lze snadno nalézt na internetu, dokáže vytvořit síťový útok takřka kdokoliv s průměrnou schopností ovládat počítač. Tato skutečnost pouze podporuje nutnost mít alespoň určitý stupeň ochrany na každé síti.

Jak roste potřeba ochrany před hrozbami internetu, tak roste i snaha o jednodušší ovládání systémů, jenž zajišťují bezpečnost počítačových sítí. Webové grafické rozhraní je jednou z možností, jak usnadnit ovládání složitějších systému či programů. První pokusy o implementaci grafického rozhraní sahají až do první poloviny šedesátých let minulého století. V té době se začaly používat počítače pro zpracování vědeckých dat. Výstup dat v číslicové podobě je pro člověka zřídka kdy srozumitelný, zvláště pak když je nutné porozumět většímu množství dat. Bylo zjištěno, že v některých případech je lepší reprezentovat data graficky. Tak je tomu i v případě systémů pro analýzu síťového provozu. Uživatelské webové rozhraní pro systém Nemea chce docílit jednoduššího ovládání, větší uživatelské přívětivosti a celkového usnadnění práce s celým systémem.

Kapitola 2

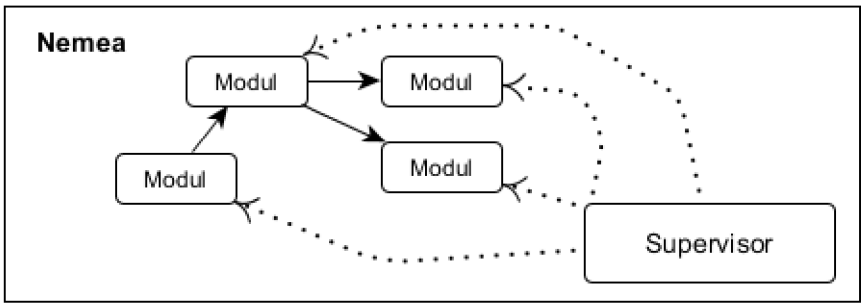
Popis cílové platformy

Tato kapitola pojednává o programu Supervisor, pro který je v rámci této práce vyvíjeno GUI [1] a který je řídícím prvek pro systém Nemea. Nemea je systém pro analýzu síťových dat. Dále je v této kapitole popisován systém Netopeer a jeho grafické rozhraní.

2.1 Nemea framework

Nemea (Network Measurements Analysis) je vyvíjena v rámci organizace CESNET skupinou Liberouter [4] a je to framework, který umožňuje poskládat z jednotlivých modulů systém, jehož účelem je automaticky analyzovat a zpracovávat síťová data.

Nemea moduly jsou reprezentovány procesy a jsou na sobě vzájemně nezávislé, díky čemuž je možné moduly odebírat a přidávat za běhu systému. Moduly komunikují pomocí knihovny libtrap [5], jež vytváří jednoduché API. Toto API dovoluje jednotlivým modulům specifikovat pouze jejich vstupní a výstupní rozhraní, o komunikaci mezi nimi se pak stará samotná knihovna libtrap. Každý modul může mít libovolný počet vstupních a výstupních rozhraní, kde každé rozhraní může být různého typu. Pomocí těchto rozhraní lze moduly navzájem spojovat a vytvářet tak systémy pro analýzu síťových dat. Ukázka systému se Supervisorem (více o Supervisorovi zde 2.3), který ovládá Nemea moduly můžeme vidět na obrázku 2.1.



Obrázek 2.1: Na obrázku lze vidět Nemea systém. Jednotlivé moduly jsou navzájem propojeny. Supervisor řídí jednotlivé moduly.

[1] GUI (Graphical User Interface) - grafické uživatelské rozhraní

2.2 Netopeer

Tento projekt se soustředí na vzdálenou konfiguraci zařízení pomocí *NETCONF* protokolu [2]. Hlavní část *NETOPEER* projektu je reprezentována NETCONF serverem se zásuvným modulem pro ovládání vzdálených zařízení. Celý projekt se potom skládá z následujících částí.

- libnetconf
- Netopeer Server
- Netopeer Client

2.2.1 libnetconf

Libnetconf je knihovna napsaná v jazyce C a slouží jako stavební kámen pro *NETCONF* klienty a servery. Poskytuje funkce, které umožňují komunikaci mezi serverem a klientem. Veškeré funkce implementované *NETCONF* klienty nebo servery jsou postaveny nad touto knihovnou.

2.2.2 Netopeer Server

Netopeer Server je nadstavbou nad *NETCONF* protokolem a snaží se o to, aby vývojáři bez znalosti tohoto protokolu byli schopni změnit nastavení síťových zařízení.

2.2.3 Netopeer klient

Cílem Netopeer klienta je vytvořit modul pro Apache [9] web server, který poskytne funkcionalitu *NETCONF* protokolu. Uživatel připraví jakékoliv změny na určitém zařízení a je schopen tyto data poslat z uživatelského rozhraní přes Apache server modul do *NETCONF* a dále do samotného zařízení.

2.3 Supervisor

Supervisor je program, který dovoluje uživateli monitorovat a konfigurovat Nemea moduly pomocí souborů XML a je tedy ovládacím prvkem pro paralelně běžící Nemea moduly [6]. Níže 2.1 lze vidět ukázku modulu DTEST_logger v konfiguračním XML souboru. Každá definice modulu obsahuje:

1. name - unikátní jméno modulu v daném xml souboru
2. enabled - příznak, který říká, zda spustit modul po startu Supervisora
3. path - cesta k binárnímu souboru modulu
4. params - parametry modulu
5. trapinterfaces - element obsahuje údaje o jednotlivých rozhraních modulu

   I note - nepovinná poznámka k rozhraní

[2] Network Configuration Protocol (NETCONF) je standardem IETF pro správu síťových zařízení. První verze byla pracovní skupinou *NETCONF* publikována v prosinci 2006 jako RFC 4741.

II type - určuje typ rozhraní, existují tři typy:

- UNIXSOCKET
- TCP
- SERVICE - umožňuje Supervisorovi získat statistiky o přijatých a odeslaných zprávách z daného modulu

III direction - určuje, jestli je rozhraní vstupní nebo výstupní

IV params - parametry rozhraní

```
<module>
  <enabled>false</enabled>
  <params></params>
  <name>flowcounter</name>
  <path>../modules/flowcounter/flowcounter</path>
  <trapinterfaces>
    <interface>
      <note></note>
      <type>TCP</type>
      <direction>IN</direction>
      <params>localhost ,8004</params>
    </interface>
  </trapinterfaces>
</module>
```

Ukázka kódu 2.1: XML example

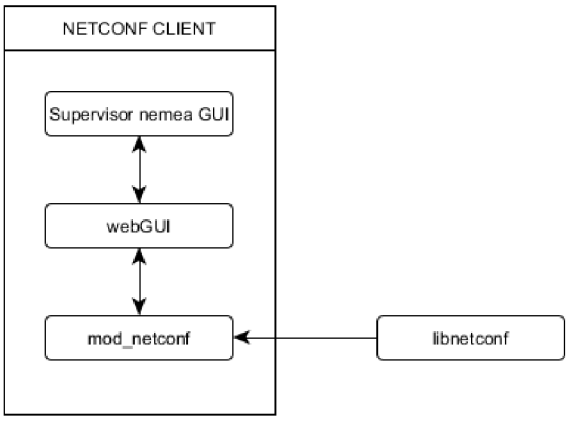
Supervisor nabízí mnoho operací, které pomocí něhož lze provádět:

1. START ALL MODULES - spustí všechny moduly z načtené konfigurace
2. STOP ALL MODULES - zastaví všechny moduly z načtené konfigurace
3. START MODULE - spustí jeden konkrétní modul z načtené konfigurace
4. STOP MODULE - zastaví jeden konkrétní modul z načtené konfigurace
5. STARTED MODULES STATUS - zobrazí status načtených modulů
6. AVAILABLE MODULES - vytiskne současnou konfiguraci
7. SHOW GRAPH - vytiskne orientovaný graf načtených modulů
8. RELOAD CONFIGURATION - uživatel může načíst konfigurace z původního nebo jiného XML souboru
9. STOP SUPERVISOR - zastaví celý proces Supervisora

Cílem této práce je tedy vytvořit grafické rozhraní, které umožní vytvářet nebo modifikovat konfigurační XML soubory. Více o tom v kapitole 5.3.

2.4 Architektura platformy

Platforma je postavena na architektuře, která je vyobrazena na obrázku 2.2. Mod_netconf je modul pro server Apache, přes nějž se komunikuje z webové aplikace přes *UNIX socket*. Naše GUI pro Nemea Supervisor komunikuje přímo s *NETCONF* klientem, tedy s jeho grafickým prostředím *webGUI*[2]. Nemea Supervisr GUI se o nižší vrstvy nezajímá.



Obrázek 2.2: Komunikace *webGUI* přes *mod_netconf* s knihovnou *libnetconf*. Novým přidaným prvkem je zde Supervisor Nemea GUI, jehož vývojem se tato práce zabývá.

Co se týká vzhledové stránky věci, *WebGUI* se skládá z několika prvků:

- Menu v horní části
- Menu v levé části
- Informační panel ve spodní části
- Plocha pro umístění Nemea Supervisor GUI

Červeným čtvercem je na obrázku 2.3 zvýrazněna plocha, do které bude umístěno naše GUI. Toto ovlivňuje rozložení prvků v návrhu grafického rozhraní.

Obrázek 2.3: Prostředí cílové platformy. Červeným čtvercem je zvýrazněna plocha, do které bude umístěno grafické rozhraní pro Supervisor Nemea.

Kapitola 3

Analýza řešení

Cílem řešení navrhovaného v této práci je transformovat konfigurační XML soubory do grafické podoby a to tak, aby každý Nemea modul byl samostatná jednotka, se kterou lze manipulovat a modifikovat. To všechno musí být uživatel schopen činit v prostředí internetového prohlížeče. Je tedy nezbytné vytvořit takové grafické elementy, kde každý element bude reprezentovat právě jeden Nemea modul, a kde je tyto elementy možné dynamicky přidávat, odebírat a modifikovat jejich parametry.

3.1 Existující řešení

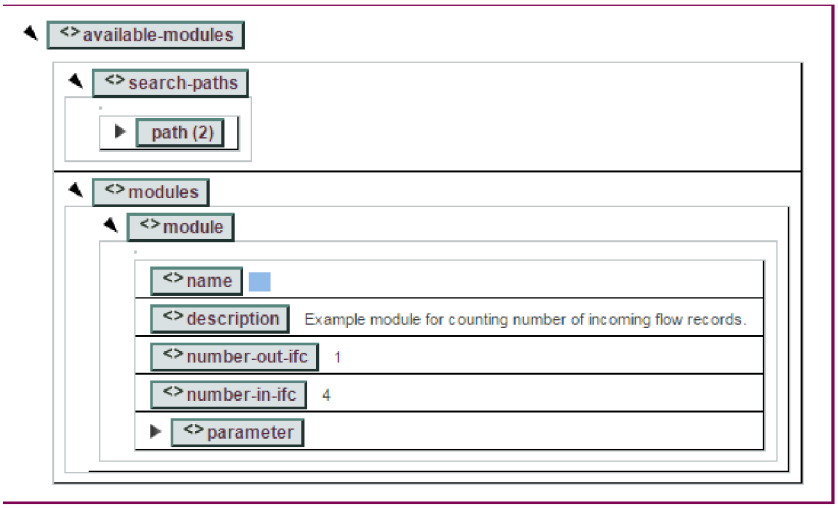
Jelikož je Supervisor Nemea GUI specifické řešení, žádné existující řešení implementující a splňující přesně definované požadavky neexistuje. Přesto se dají najít ukázky programů, které mají částečnou funkcionalitu požadovanou od našeho Supervisor Nemea GUI.

3.1.1 XmlGrid

Tento nástroj umožňuje vizualizovat XML soubory do přehledného stromového grafu, jak je možné vidět na obrázku 3.1. Uživatel může dokonce modifikovat hodnoty jednotlivých XML elementů. Tato aplikace ovšem nijak neumožňuje přidávat nové XML elementy, či je navzájem spojovat.

3.1.2 Gliffy

Gliffy je online nástroj pro vytváření diagramů, umí vytvářet grafické elementy a vzájemně je spojovat. Pro potřeby této práce však má příliš mnoho zbytečných funkcí, a naopak základní funkce jako provázání diagramu se XML soubory zde zcela schází.



Obrázek 3.1: Ukázka práce s online vizualizérem XML souborů - XmlGrid.

3.2 JavaScript

Jak již bylo zmíněno v úvodu této kapitoly, je třeba upravovat obsah webové stránky, a to nejlépe tak, aby nebylo nutné při každé změně stránku znovu načítat. K tomuto účelu se využívá JavaScript. Tento jazyk patří do rodiny skriptovacích jazyků, které není nutno před spuštěním kompilovat [13]. JavaScript je v této práci použit vzhledem ke kladným předchozím zkušenostem autora s tímto jazykem. Další otázkou, kterou bylo potřeba vyřešit, bylo, zda k ulehčení vývoje použít knihovny nebo ne.

3.3 Možnosti využití knihoven pro JavaScript

V dnešní době stále ne všechny prohlížeče interpretují JavaScript stejně [7]. Toto představuje problém pro programátora, který následně musí řešit kompatibilitu mezi jednotlivými prohlížeči. Za účelem vyhnutí se tomuto problému lze použít jednu z knihoven, která odstraňuje rozdíly v interpretaci JavaScriptu jednotlivými prohlížeči. Mezi nejpopulárnější knihovny tohoto druhu patří jQuery [1] anebo MooTools [2]. V návaznosti na předchozí výrazně pozitivní zkušenosti, dobrou dokumentaci a existenci velkého množství návodů na internetu byl zvolen jQeury.

Výhody využití jQuery:

- řeší kompatibilitu kódu v různých prohlížečích
- zjednodušuje manipulaci s DOM elementy
- dovoluje obsluhovat události jako stisk tlačítka a jiné

Nevýhody využití jQuery:

- při načtení stránky si uživatel musí stáhnout jQuery knihovnu (dá se využít i online repositářů, v takovém případě je možné považovat za nevýhodu nutnost připojení k internetu)

[1] https://jquery.com/
[2] http://mootools.net/

Po zvážení všech pro a proti nakonec jQuery bude využit při vývoji Supervisor Nemea GUI.

Jelikož existuje mnoho kvalitních knihoven, které umí vykreslit grafické objekty, stojí za zvážení, jestli některou z nich nevyužít pro popisované GUI. Po úvodním průzkumu bylo zjištěno, že tyto knihovny zvládají mnohé funkce, které jsou pro fungování našeho GUI nezbytné. Vlastní implementace těchto metod je sice možná, ale představovala by nadměrnou časovou zátěž a mohla by se stát zdrojem chyb. Na druhou stranu použití knihoven vede k částečné ztrátě kontroly nad kódem a vyžaduje spoléhat na to, že metody knihovny jsou schopny poskytnout veškerou potřebnou funkcionalitu. Dalším mínusem je, že dochází opětovnéu zatížení klienta, který je nucen stahovat soubory knihovny. Následuje analýza nabízených možností.

Podle stránky [3], která nabízí srovnání knihoven pro vykreslení diagramů, patří mezi nejlepší například tyto:

- jointJS [3]
- Mxgraph [4]
- Raphael [5]
- D3 [6]
- GoJS [7]

Nicméně ne u všech licence dovoluje jejich použití. Například interní využití GoJS pro jednoho vývojáře stojí 1 350 dolarů, Mxgraph pak dokonce 5 000 dolarů. Tím se výběr zužuje na D3, Raphael a jointJS.

3.3.1 D3

D3 je nejpopulárnější knihovna ze tří jmenovaných, o čemž svědčí 37 613 hvězdiček na GitHubu [8]. Tato knihovna využívá široce rozšířených standardů jako SVG, HTML, CSS. D3 obsahuje několik stovek funkcí, které mohou být seskupeny do několika okruhů. Mezi ty zajímavější z pohledu této práce patří:

- Matematický okruh - 2D transformace: rotace, zmenšení/zvětšení, posunutí, zkosení
- Okruh barev - podpora RGB, HSL, HCL, zesvětlení/ztmavování
- Okruh rozložení (tvz. *layout*) - strategie pro vzájemné rozložení grafických elementů.

Všechny výše zmíněné okruhy jsou vhodné pro potřeby Supervior Nemea GUI, navíc dokumentace je dobře zpracovaná, a ani licence (BSD) nepředstavuje problém.

[3] http://www.jointjs.com/
[4] https://www.jgraph.com/
[5] http://raphaeljs.com/
[6] http://d3js.org/
[7] http://gojs.net/latest/index.html
[8] https://github.com/mbostock/d3

3.3.2 Raphael

Tato knihovna není ve srovnání s předešlou zdaleka tak populární. Na GitHubu má "pouhých"7 432 hvězdiček. Obdobně ani její funkcionalita nedosahuje možností knihovny D3. Její výhodou je malá velikost, podpora všech nejpoužívanějších prohlížečů a dobrá dokumentace.

3.3.3 jointJS

JointJS je moderní HTML5 knihovna, která nabízí řadu funkcí:

- MVC architektura
- Spojování elementů pomocí spojnice
- elementy a spojnice lze libovoně připůsobovat
- Magnety - body, přes které se jednotlivé elementy spojují
- obsluha událostí - lze reagovat téměř na cokoliv
- možnost vzájemného rozložení grafických prvků

Mezi další výhody patří velmi dobrá dokumentace, široká škála tutoriálů a návodů. Neméně důležitým faktorem byly dobré reference v autorově okolí.

Na základě zjištění, jaké mají knihovny schopnosti a jaké množství kódů by bylo nutné napsat k jejich nahrazení, se nejlepší volbou ukázalo knihovny v Supervisor Nemea GUI využít. Svou funkčností, celkovým zpracováním a kvalitní dokumentací byla pro tuto práci vybrána knihovna jointJS.

3.4 Vývojové prostředí

Zadavatel poskytnul virtuální stroj, na němž již byl předinstalován Netopeer klient i s uživatelským rozhraním. Technické podmínky bohužel vývoj v tomto prostředí neumožnily, a proto byla zvolena možnost vyvíjet na soukromém webu až do momentu, kdy bylo potřeba zasadit Supervisor Nemea GUI do Netopeer klienta.

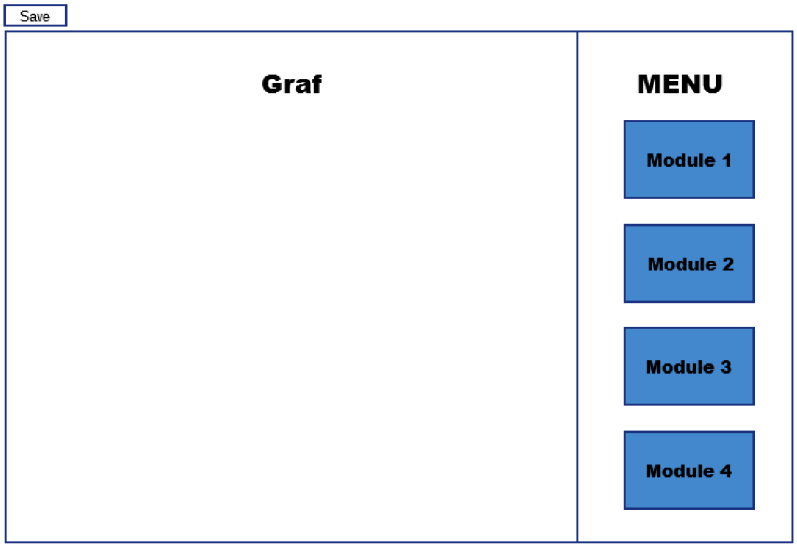
Kapitola 4

Návrh grafického rozhraní

Návrh grafického rozhraní musí splňovat požadavky na něj kladené. V tomto případě se požaduje, aby při načtení pracovní plochy byly dostupné moduly, jež lze přidat do konfigurace tzv. *available-modules*. Další komponentou, která se musí vyskytovat v rámci GUI, je plocha, jež má umožnit práci s moduly. Tato komponenta je v této práci nazývána Graf. Dále je potřeba vytvořit rozhraní pro tlačítka, která mohou sloužit k uložení právě rozpracované konfigurace nebo k jiným činnostem.

4.1 Původní návrh

Na obrázku 4.1 můžeme vidět původní návrh grafického rozhraní. Tento návrh rozděluje pracovní plochu vertikálně na dvě části. V pravé části se nachází Menu, které obsahuje dostupné moduly, jež lze přidat do hlavní části rozhraní, Grafu. Nad Grafem se nachází úzký pruh, do nějž lze umístit tlačítka potřebné pro ovládání GUI. Jedná se o hrubý návrh, který byl navržen na začátku vývoje a neřeší detaily jednotlivých částí rozhraní ani možnosti cílového prostředí.



Obrázek 4.1: Původní návrh grafikého rozhraní.

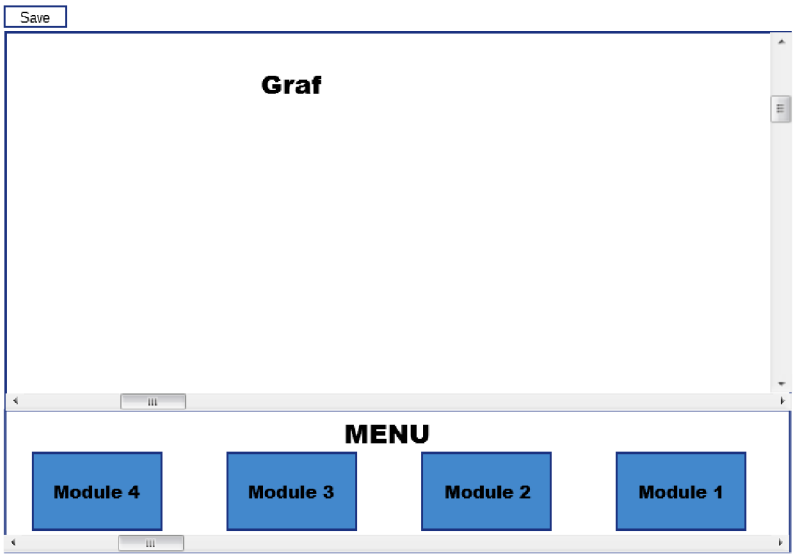
4.2 Další návrhy

Postupně, jak docházelo k využívání GUI, se objevovaly nedostatky původního návrhu. Při větším množství modulů byla velikost částí Grafu a Menu nedostatečná. Bylo tedy nutné je přizpůsobit jejich velikost podle aktuálního množství modulů. Jelikož ale pracovní plocha určená pro GUI byla na platformě již vyměřena, musel se tento problém řešit přidáním prvků typu scroll-bar jak pro Graf, tak i Menu.

Scroll-bar dovoluje zvětšit původní plochu, avšak i to, že uživatel nemůže vidět všechny moduly v jeden čas, zůstává mínusem.

Další skutečností, která se musela v návrhu GUI reflektovat, je přidávání nových modulů do Grafu. Při této akci totiž musí být nastaveny některé hodnoty definující vlastnosti modulu. Vhodným řešením tohoto problému se jeví okno, které vyskočí při přidání modulu z Menu do Grafu s možností doplnit informace nezbytné pro správné fungování modulu v konfiguraci. Toto okno je ve finální verzi vidět na obrázku 6.2.

Nejvýznamnější změnou v grafickém návrhu bylo přesunutí Menu z pravého sloupce pod Graf jak můžeme je vidět v návrhu 4.2. Tato změna byla provedena z důvodu rozšíření Grafu. I když došlo ke snížení výšky Grafu, pro práci s Nemea moduly, je šířka podstatnějším rozměrem než výška.



Obrázek 4.2: Menu bylo přesunuto pod Graf z důvodu rozšíření Grafu v horizontální rovině.

Kapitola 5

Realizace

Tato kapitola popisuje jednotlivé logické celky programu, technologie, které byly využity pro implementaci jednotlivých částí, a také funkčnost těchto celků. Celou kapitolou se prolínají zmínky o knihovně jointJS, která je stěžejní pro fungování celého programu.

5.1 Logické dělení

Program se dá rozdělit na tři logické celky:

1. Definice Grafu a Menu, o kterých je zmínka v předchozí kapitole 4.1 a dalších proměnných, které se budou využívat v celé šíři programu
2. Načtení a vykreslení Nemea modulů z XML souboru
3. Interaktivita s uživatelem - modifikace vlastního diagramu
4. Převod diagramu zpět na XML soubor

5.2 Globální promněné knihovny jointJS

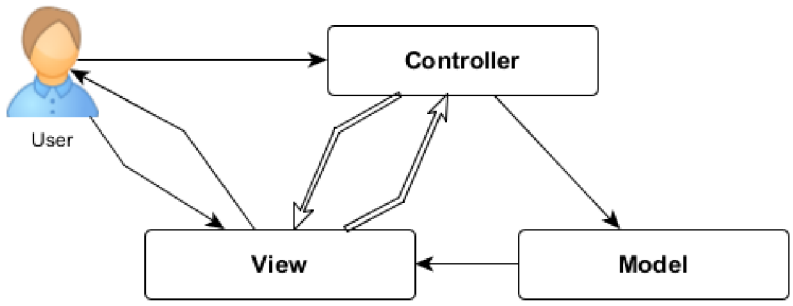
Pro definování těchto proměnných je důležité si uvědomit, že knihovna jointJS využívá MVC architektury a je postavena na knihovně Backbone.js.

5.2.1 MVC

Mnoho moderních frameworků dostupných pro JavaScript dovoluje vývojářům organizovat jejich kód. MVC rozděluje webovou aplikaci na tři části Model-View-Controller [8]:

- Model reprezentuje data a business logiku aplikace. Mezi takové data například může patřit fotka či text.
- View představuje uživatelské rozhraní aplikace. View by nemělo přímo komunikovat s Modelem.
- Controller by měl tvořit komunikační kanál mezi Model a View. Uživatel je v interakci s View (kliknutí a jiné uživatelské akce) a na jejich základě aktualizuje Model. Controller tedy obecně tvoří aplikační logiku programu.

Ne všechny knihovny JavaScriptu striktně dodržují rozdělení popsané výše. Backbone.js například přenáší část odpovědnosti z Controlleru na View. Příklad MVC architektury je uveden na obrázku 5.1



Obrázek 5.1: Ukáza komunikace mezi jednotlivými složkami MVC architektury.

5.2.2 Backbone.js

Tento framework přidává strukturu kódu na straně klienta. Kód je tak snadno rozšiřitelný a udržovatelný z dlouhodobého hlediska. Backbone.js se většinou využívá na vytváření jednostránkových aplikací (tzv. SPAs [1]). Tyto aplikace se vyznačují tím, že jsou jednou načteny a pak reagují na vstupy uživatele bez nutnosti znovunačtení celé stránky ze serveru.

5.2.3 Definice Grafu a Menu

Graf i Menu se skládá z Modelu a View. Model je v jointJS reprezentován proměnnou *joint.dia.Graph* a View proměnnou *joint.dia.Paper* [14]. Tyto globální proměnné jsou definovány následovně:

- *joint.dia.Graph* je model, tak jak je popsaný knihovnou Backbone.js, který drží všechny buňky diagramu. Kolekce všech buněk je přístupná skrze vlastnost *cells*.
- *joint.dia.Paper* je pohled (anglicky *view*), který se naváže na globální proměnnou Graph a poté se stará o automatické rendrováni grafických elementů všech buněk přidaných do *joint.dia.Graph*

V ukázce kódu 5.1 můžeme vidět definici plochy, která bude sloužit pro manipulaci Nemea modulů. Zatímco na definici *joint.dia.Graph* není nic zvláštního, u definice *joint.dia.Paper* lze vidět, že přijímá objekt *options* jako argument, který může mít mimo jiné tyto vlastnosti:

1. el - CSS selektor, jQuery objekt nebo DOM element, který bude držet promněnou *joint.dia.Paper*
2. gridSize - tato vlastnost určuje minimální počet pixelů, o které je možno posunout garifcký element na *joint.dia.Paper*
3. model - ukazuje na objekt typu *joint.dia.Graph*, s nímž má být *joint.dia.Paper* provázána

[1] SPAs - je zkratka z anglického *single-page applications*. Tady jedno stránkové aplikace.

```
graph = new joint.dia.Graph;
paper = new joint.dia.Paper({
    el: $('#graph'),
    gridSize: 1,
    model: graph,
});
```

Ukázka kódu 5.1: Ukázka definice gobální promněnné knihovny jointJS *joint.dia.Paper*

5.3 Načtení XML souborů

Když jsou plochy, na kterých se bude pracovat, připraveny, je nutné si načíst XML soubor s konfiguračními údaji pro Nemea moduly, aby mohly být na danou plochu vykresleny.

5.3.1 Struktura konfiguračního XML souboru

```
<?xml version="1.0" encoding="UTF-8"?>
<nemea-supervisor>
    <available-modules>
        <search-paths>
            <path>/usr/local/bin/</path>
            <path>...</path>
        </search-paths>
        <modules>
            <module>
            </module>
        </modules>
    </available-modules>
    <modules>
        <module>
        </module>
    </modules>
</nemea-supervisor>
```

Ukázka kódu 5.2: Příklad možného konfiguračního souboru pro Supervisor nema GUI. Elementy modul jsou popsané zvlášť, proto jsou zde pro přehlednost vynechány.

Z kódu vyplývá, že kořenovým ulzem XML je <nemea-supervisor>, který má dva potomky <available-modules> a <modules>. První z nich, tedy <available-modules>, obsahuje elementy <search-paths>, který obsahuje cesty k binárním kódům modulům a <modules>, tento potomek uchovává informace o jednotlivých Nemea modulech, které budou načteny do Menu. Stojí za povšimnutí, že XML soubor obsahuje dva typy modulů - moduly určené pro načtení do Grafu (zde zvané "moduly", které jsou popsány v kapitole 2.3) a moduly, které se načtou do Menu (tzv. "dostupné moduly"). Tyto dostupné moduly jsou v XML souboru popsány následujícími XML elementy:

- name - jméno dostupného modulu, může být i prázdné, nastavuje se při přidání dostupného modulu do Grafu.
- description - popis modulu
- number-out-ifc - počet výstupních rozhraní
- number-in-ifc - počet vstupních rozhraní
- parameter - určuje možné parametry modulu

Definici dostupného modulu si lze prohlédnout v ukázce kódu 5.3.

```
<module>
    <name>MyModule</name>
    <description>Example module for counting number
    of incoming flow records.</description>
    <number-out-ifc>1</number-out-ifc>
    <number-in-ifc>4</number-in-ifc>
    <parameter>
        <short-opt>-n</short-opt>
        <long-opt>--n</long-opt>
        <description>idk</description>
        <mandatory-argument>Argument</mandatory-argument>
        <argument-type>Type</argument-type>
    </parameter>
</module>
```

Ukázka kódu 5.3: Ukázka definice dostupného modulu v XML konfiguračním souboru.

5.3.2 Ajax

Tato technologie je v podstatě XMLHttpRequest objekt v JavaScriptu, který vysílá požadavky na server. Může poslat nebo obdržet data v mnoha formátech jako JSON [2], XML, HTML [11]. Nejdůležitější ovšem je, že to dokáže bez nutnosti znovunačíst stránku. To je důvodem, proč Supervisor Nemea GUI využívá této technologie na odeslání i získání konfiguračního XML souboru ze serveru.

5.3.3 Zpracování XML řetězce

Jako nejlepší způsob, jak uchovat informace, které XML soubor nese, v JavaScriptu se jeví uložení přímo těchto hodnot do grafické reprezentace Nemea modulů. Knihovna jointJS vytváří *joint.dia.Element* modely, které díky vlastnosti prop umožňují svázat jakékoliv data přímo s grafickým elementem.

[2] JSON (JavaScript Object Notation) je odlehčený formát pro výměnu dat. Je jednoduše čitelný i zapisovatelný člověkem a snadno analyzovatelný i generovatelný strojově.

```
element.prop('name', 'Jirka')
element.prop('name')   // 'Jirka'
```

Ukázka kódu 5.4: Ukázka práce s *joint.dia.Element* a jeho vlastností prop.

Pro převod zpracování XML souboru je využito XML DOM. DOM je rozhraní nezávislé na platformě a programovacím jazyku, které umožní aplikacím dynamicky zpřístupnit a modifikovat obsah, strukturu a styl daného dokumentu [10]. DOM umožňuje přístup k dokumentu jako ke stromové datové struktuře. Toto vyžaduje nahrání celé struktury do paměti. Uzel je objekt, který je základem ve stromové struktuře DOM. Dokument je XML třída, která podporuje metody pro práci s uzly v celém dokumentu XML. Uzel má tyto vlastnosti:

- nodeName - jméno uzlu
- nodeValue - hodnota uzlu
- parentNode - rodičovský uzel
- childNodes - potomci uzlu
- attributes - atributy uzlu

Metody uzlu jsou:

- getElementsByTagName(name) - argument této metody je jméno uzlu a vrací všechny uzly s tímto jménem
- appendChild(node) - vloží nového potomka, argument je uzel, který má být vložen
- removeChild(node) - odstraní potomka, argument je uzel, který má být odstraněn

Zpracování XML potom probíhá tak, že metodou getElementsByTagName() se program dotazuje, zda je v XML souboru přítomný element <available-modules:, >. Pokud ano, je zřejmé, že se tam musí vyskytovat právě dva elementy <modules>. První potom obsahuje všechny dostupné moduly a druhý všechny moduly, které jsou určeny do Grafu. Pak se prochází všechny elementy <module> a díky XML DOMu je snadné dostat všechny potřebné hodnoty uzlů, které se následně uloží do struktury daného grafického elementu. Tyto elementy jsou uloženy v poli, které je pomocí metody knihovny jointJS přidáno do Grafu.

5.3.4 Spojení správných modulů

Poté, co jsou moduly přidány do grafu, je potřeba je provázat mezi sebou. Uzel <params> u každého vstupního a výstupního rozhraní je klíčem k tomu určit, jaké moduly spojit. Uzel <params> se skládá z portu a adresy, které jsou navzájem odděleny čárkou. Pro spojení dvou modulů je nutné najít dvojici vstupního a výstupního rozhraní, které mají stejný port. Aby vyhledávání bylo snazší, přidala se do struktury každého rozhraní kromě hodnot jednotlivých uzlů z XML konfiguračního souboru také vlastnost compareParam. Právě v ní je uloženo číslo portu daného rozhraní.

Ještě než bude popsáno, jak se jednotlivé vstupní a výstupní rozhraní spojují, je podstatné zmínit, že každému modulu, který je přidán do Grafu, přiřadí jointJS jednoznačné

identifikační číslo. Toto číslo má formu pseudo-generovaného UUID [3]. Díky tomu je možné přistupovat k jednotlivým modulům.

Samotné vyhledávání dvojic vstupní a výstupní rozhraní se stejnou hodnotou v compareParam probíhá v několika krocích. V prvním kroku se hledají všechny výstupní rozhraní. Pokud je takové rozhraní nalezeno, je sestavena struktura, která nese informace o daném rozhraní - konkrétně jednoznačné identifikační číslo modulu (dále ID) a číslo složené z ID modulu a čísla samotného rozhraní, které je taktéž jednoznačné. Dále pak struktura obsahuje hodnotu compareParam a typ rozhraní. Ve druhém kroku se porovnávají vstupní rozhraní s nově vytvořenou strukturou. Pokud je nalezena shoda ve vlastnosti compareParam, je mezi dvěma Nemea moduly vytvořena spojnice.

5.4 Interaktivita s uživatelem

Důležitou součástí každého grafického rozhraní je jeho ovládání. Interaktivita s uživatelem by měla být intuitivní a jednoduchá. Od vytvářeného grafického uživatelského rozhraní se očekávají následující prvky:

- Přidat Nemea modul z Menu do Grafu.
- Mazat po jednom Nemea moduly z Grafu
- Mazat více Nemea modulů z Grafu zároveň
- Vytvářet spojnice mezi moduly
- Mazat spojnice mezi moduly
- Zobrazit popis modulu z Menu
- Měnit jednotlivé parametry a typy rozhraní
- Vhodně rozložit prvky na grafu
- Uložit současnou konfiguraci

5.4.1 Přidání Nemea modulu do Grafu

Pro přidání Nemea modulu z Menu do Grafu se využívá obsluhy událostí, které poskytuje knihovna jointJS. Mezi tyto události patří *cell:pointerdblclick*, která je vyvolána dvojklikem na některou buňku vyrendrovanou na *joint.dia.Paper*. Tato událost je naimplementována tak, že se naklonuje daný Nemea modul a přidá se do Grafu pomocí metody *add*. Moduly v Grafu ovšem mohou mít jiné parametry než dostupné moduly, a proto se při obsloužení události *add* vyvolá vyskakovací okno, které nabídne uživateli nastavení všech potřebných údajů Nemea modulu. Jedná se přesněji o položky uvedené v sekci 2.3 a musí splňovat podmínky, které jsou uvedeny v téže sekci. Navíc musí být vyplněny parametry u každého výstupního rozhraní. To je dáno tím, jak se spojují Nemea moduly.

[3] UUID (Universally unique identifier) - je 128 bitové číslo. V kanonické podobě je reprezentováno 32 malými hexadecimálními čísly seskupenými do pěti skupin, kde jsou jednotlivé skupiny odděleny pomlčkou.

5.4.2 Mazání Nemea modulů z Grafu

Mazání modulů je provedeno tak, aby bylo možné mazat moduly jednotlivě nebo po skupinách. Tentokrát se k tomu využívá obsluha události *cell:pointerclick*. Pokud uživatel klikne na modul, vybraný modul změní barvu. Tím je uživatel informován o tom, že výběr modulu proběhl úspěšně. Ke smazání modulu potom stačí zmáčknout klávesu *delete*. Pokud je již jeden modul vybrán a uživatel označí jiný modul, první modul bude odznačen. Přeje-li si uživatel smazat více modulů najednou, využije k tomu klávesu *ctrl*. Když je klávesa stisknuta, uživatel může kliknutím myši označit více modulů zároveň. Pak stačí zmáčknout klávesu *delete* a odstraní se všechny moduly.

Mazání modulu z Grafu má ovšem za následek, že se odstraní všechny spojení s tímto modulem.

5.4.3 Spojování nemea modulů

Nemea moduly mají vstupní a výstupní rozhraní, která umožňují spojení dvou či více modulů. Vstupní a výstupní rozhraní jsou oddělena barvou, aby je uživatel byl na první pohled schopen rozlišit. Podobně se barvou liší i různé typy výstupních rozhraní. Jinou barvou je reprezentováno výstupní TCP rozhraní a jinou UNIXSOCKET. Uživatel může propojit pouze výstupní rozhraní se vstupním, naopak to nelze. Provede se to kliknutím myši na výstupní rozhraní a tahem myši na vstupní rozhraní, které se v případě najetí myši označí červeným čtvercem.

Každá spojnice má štítek, který nese informaci o portu výstupního rozhraní. Pokud dojde ke změně hodnoty v parametru rozhraní, odstraní se všechny spojnice, které jsou na toto rozhraní připojeny. Stejně tak se odstraní všechny spojnice při změně typu rozhraní. Pokud je odstraněna spojnice, vstupnímu rozhraní, na které byla napojena, se změní hodnota parametru na prázdný řetězec.

5.4.4 Ostatní možnosti ovládání

Další důležité prvky ovládání jsou tlačítka umístěné nad Grafem. Mezi ně patří:

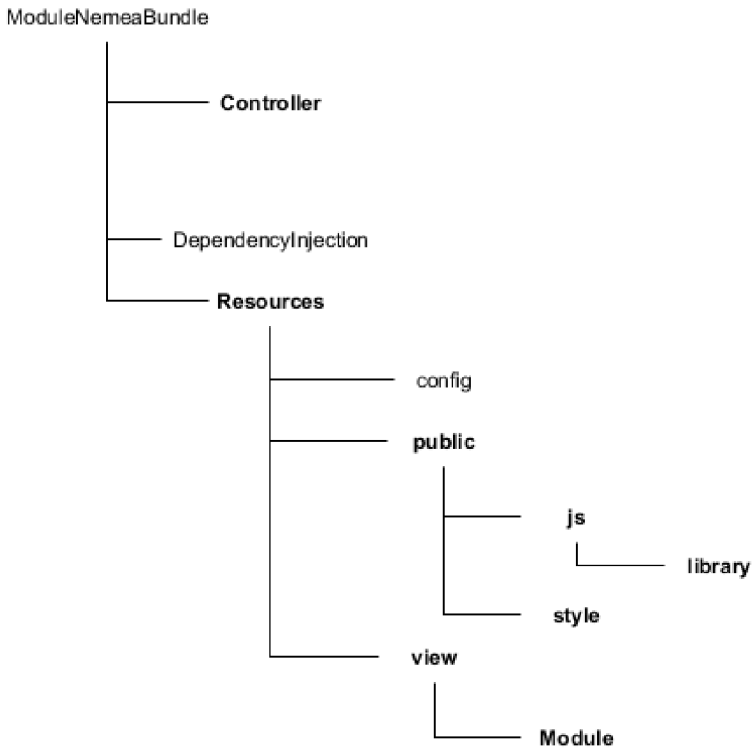
Tlačítko	**Funkce**
Save	Převede diagram na XML soubor a ten odešle na server
Remove	Odstraní všechny moduly z Grafu
Reload GUI	Smaže všechny moduly z Menu a Grafu a načte původní XML soubor
Layout	Vhodně rozloží moduly na Grafu

5.5 Převod diagramu na XML

Převod diagramu do výsledného konfiguračního XML souboru probíhá tak, že se pomocí metody *getElements*, kterou poskytuje knihovna jointJS, získají elementy z Grafu. Jelikož informace o Nemea modulech jsou přímo vázány na tyto elementy, stačí tyto informace uložit do proměnné typu string obalené správnými XML uzly. Poté se celý řetězec převede na XML objekt a odešle se na server.

5.6 Zasazení do platformy

V této části je popsána podrobněji struktura platformy, do které bude Supervisor Nemea GUI zasazeno. Celá platforma je postavena na PHP frameworku Symfony 2. V tomto frameworku jsou veškeré spolu související části shlukovány do balíků tzv. *budnles*. Pro Supervisor Nemea GUI byl připraven balík, který má následující stromovou strukturu.



Obrázek 5.2: Struktura adresáře ModuleNemeaBundle. Tučně jsou vyznačeny adresáře, které jsou významné z pohledu této práce.

Z obrázku 5.2 vyplývá, že název balíčku připraveného pro vyvíjené GUI je ModuleNemeaBundle, který obsahuje důležité podadresáře:

- Controller - obsahuje soubor ModuleController.php, který bude spojojovat Supervisor Nemea GUI s back-endem aplikace.
- public:
  - js - obsahuje soubor module-nemea.js, ve kterém je logika celého popisovaného GUI. Složka library potom slouží k umístění knihovny jointJS
  - style - zde jsou uloženy kaskádové styly, které využívá knihovna jointJS. Také se zde nachází kaskádové styly pro Supervisor Nemea GUI samotné.
- view - zde se nachází soubory pro tzv. HTML šablony. Tyto šablony jsou definovány jako dokument obsahující zápis HTML značek dle standardu W3C [12]. V adresáři Module se pak nachází HTML šablona, která popisuje základní kostru stránky určenou pro Supervisor Nemea GUI (tu můžeme vidět zde 4.2).

Kapitola 6

Testování

Testovávání je nedílnou součástí každého vývoje aplikace. Při testování se často používají testovací případy. Ty popisují konkrétní akce prováděné s určitou softwarovou komponentou a jejich očekávané výsledky [1]. Sada několika testovacích případů tvoří testovací scénáře. Testovací případy na sebe musí navazovat tak, aby vytvořili logickou posloupnost kroků, která otestuje určitou funkčnost aplikace. Správný testovací scénář by měl obsahovat:

- testovací oblast - jakou oblast aplikace bude scénář testovat
- testovací případy - seznam testovacích případů, ze kterých se testovací scénář skládá
- vyhodnocení - určení, zda scénář dopadl dobře či nikoliv

6.1 Testovací scénář

Byl vytvořen jeden testovací scénář pro ověření funkčnosti vyvíjeného GUI. Testovací oblastí bude celá šíře funkcionality popsána v této práci, konkrétně se tedy bude jednat o tyto kroky:

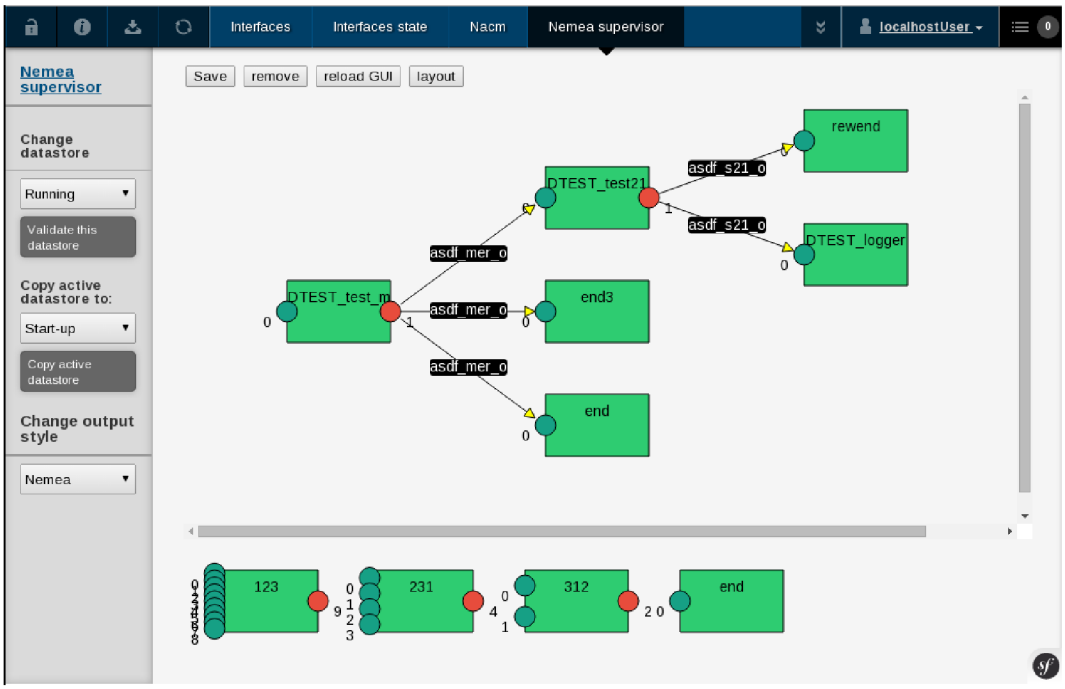
1. načtení konfigurace z daného XML souboru
2. přidání Nemea modulu z Menu do Grafu
3. mazání Nemea modulů z Grafu
4. správné rozložení Nemea modulů v Grafu vůči sobě
5. změna typu rozhraní jednoho z Nemea modulů
6. uložení současné konfigurace

XML soubor využit pro tento scénář je možné najít zde B.1 Na závěr je testovací scénář vyhodnocen.

6.1.1 Načtení konfigurace

Prvním testovacím případem logicky musí být načtení konfiguračního XML souboru po načtení stránky. Když uživatel vstoupí na stránku musí se do Grafu a Menu nahrát Nemea moduly tak, jak jsou popsány v XML souboru. Na obrázku 6.1 je zobrazen stav vyvíjeného

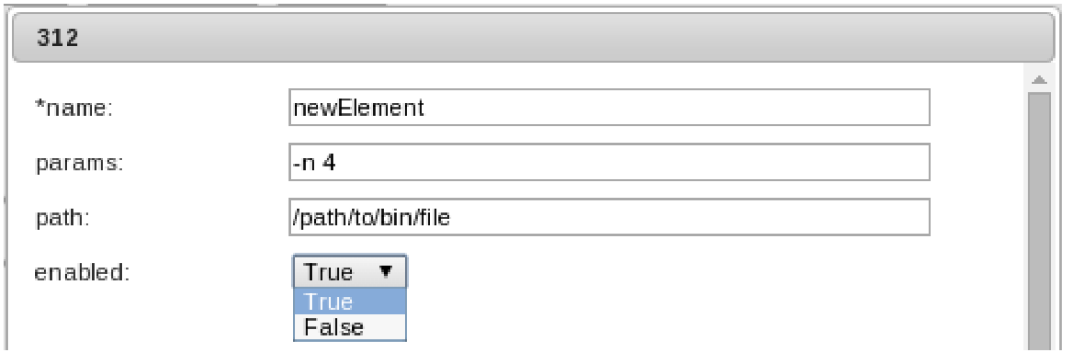
GUI hned po načtení stránky. Lze vidět, že všechny Nemea soubory definované v XML souboru jsou vyrendrovány v Grafu a Menu. Dále je potřeba zkontrolovat, zda všechny moduly mají stejné parametry jako v konfiguračním XML souboru. Po manuální kontrole bylo zjištěno, že parametry se shodují, a tím byl testovací případ vyhodnocený úspěšně.



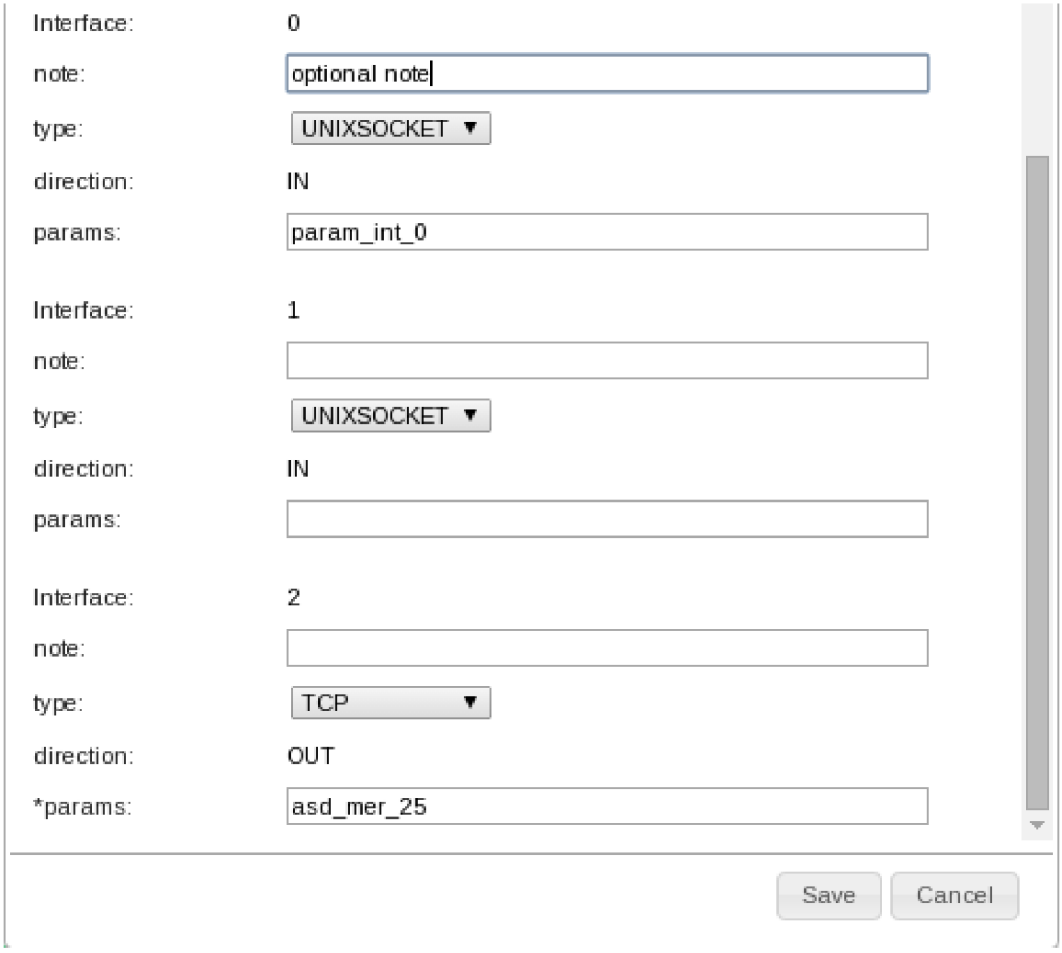
Obrázek 6.1: Stav Supervisor Nemea GUI hned po na4ten9 stránky.

6.1.2 Přidání Nemea modulu z Menu do Grafu

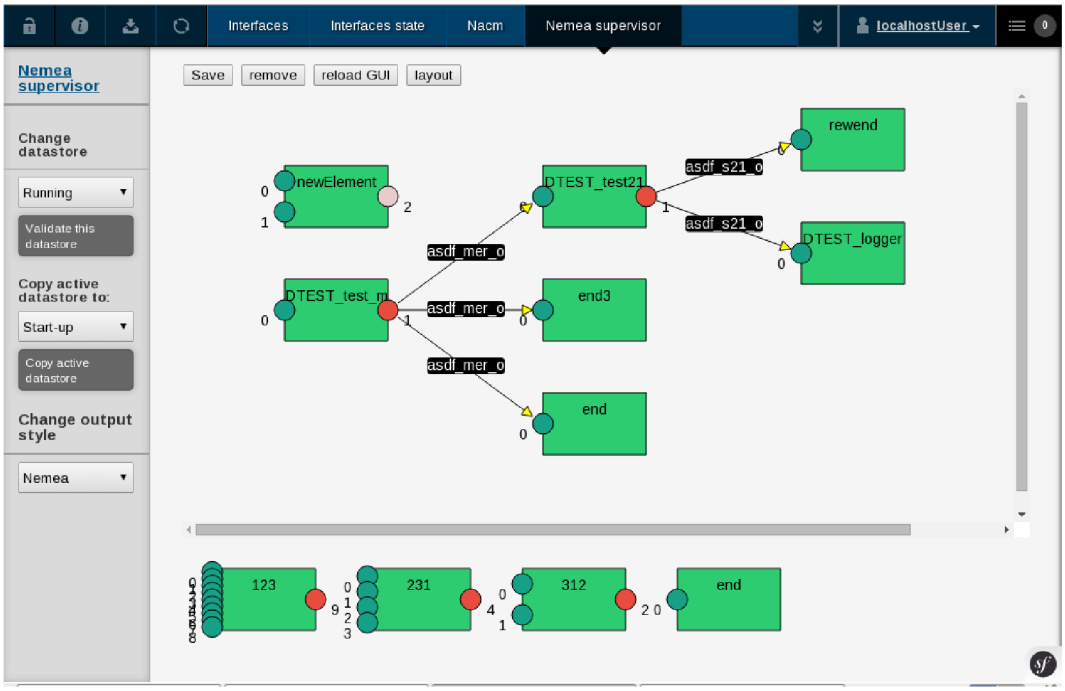
Dalším krokem, jak modifikovat současnou konfiguraci, je možnost přidat Nemea modul z Menu do Grafu. Očekávané chování aplikace v této situaci je takové, že se objeví vyskakovací okno s parametry daného modulu a všech jeho rozhraní. Toto okno můžeme vidět na obrázku 6.2 a 6.3. Povinné parametry označené znakem "*"musí být vyplněny. Pokud tak uživatel neučiní, aplikace ho upozorní. Lze si všimnout, že bylo zvoleno výstupní rozhraní typu "TCP". Tímto krokem se testuje, zda jsou rozhraní opravdu odlišena barvou podle svého typu. Po stisknutí tlačítka Save by měl být modul přidán do grafu. Výsledek můžeme vidět na obrázku 6.4. Modul byl úspěšně přidán do Grafu a výstupní rozhraní má narůžovělou barvu, která značí, že typ tohoto rozhraní je "TCP". Tento testovací případ je vyhodnocen kladně, protože modul je v Grafu a barva jeho výstupního rozhraní rozlišuje typy.



Obrázek 6.2: Vyskakovací okno, které se objeví při pokusu přidat Nemea modul do Grafu. Z důvodu velikosti bylo rozděleno do dvou obrázků.



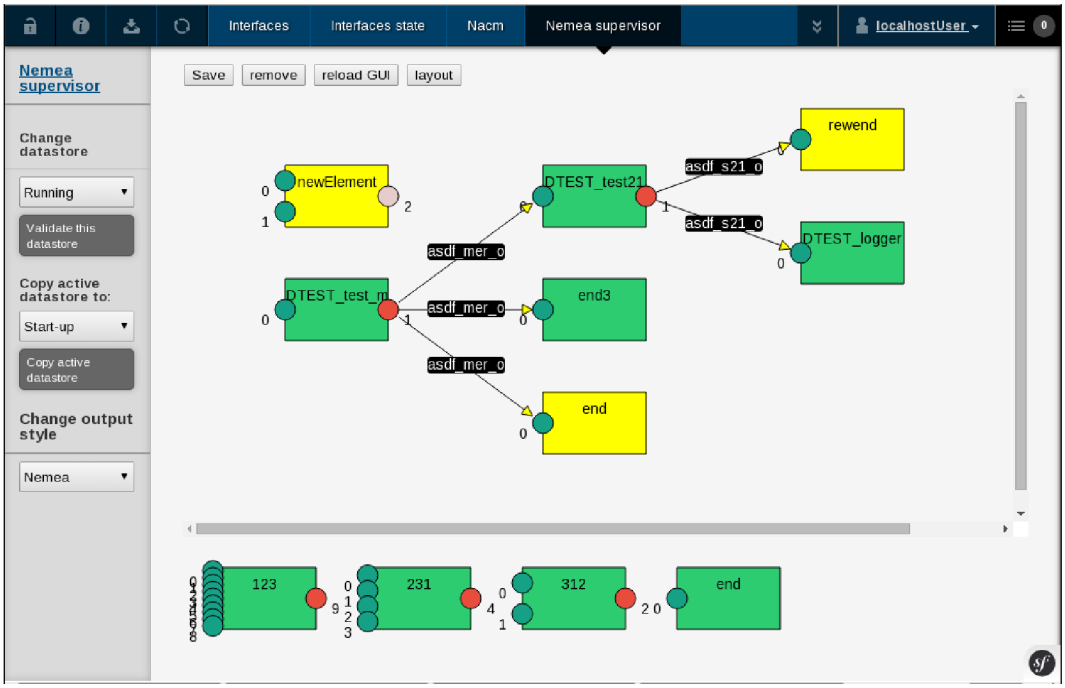
Obrázek 6.3: Vyskakovací okno, které se objeví při pokusu přidat Nemea modul do Grafu. Z důvodu velikosti bylo rozděleno do dvou obrázků.



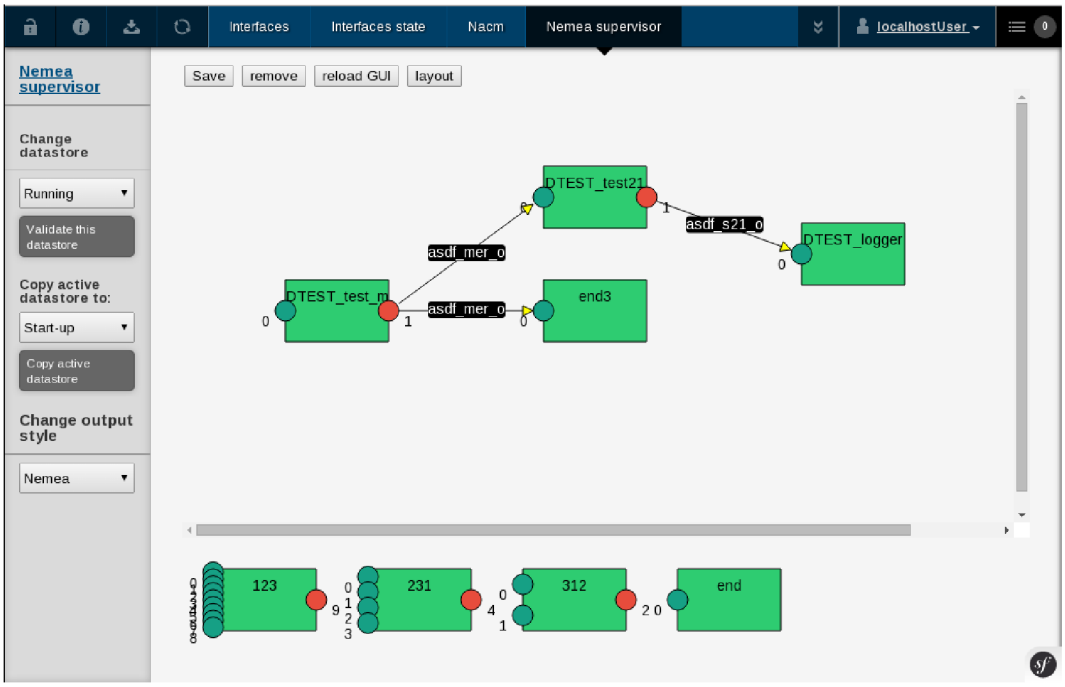
Obrázek 6.4: Na obrázku je vidět, že modul newElement byl úspěšně přidán do Grafu.

6.1.3 Mazaní elementů

Při tomto kroku je otestováno vybírání modulů a jejich následné smazání. Aplikace by se měla zachovat tak, že vybrané moduly obarví žlutou barvou 6.5. Po zmáčknutí klávesy delete by tyto moduly měly být odstraněny z Grafu a s nimi všechny spojnice, které jsou připojeny na jejich rozhraní. Z obrázku 6.6 lze poznat, že se tak opravdu stalo, a proto je tento testovací případ vyhodnocen kladně.



Obrázek 6.5: Obrázek ukazuje,že vybrané moduly Nemea jsou vybarveny žlutou barvou.

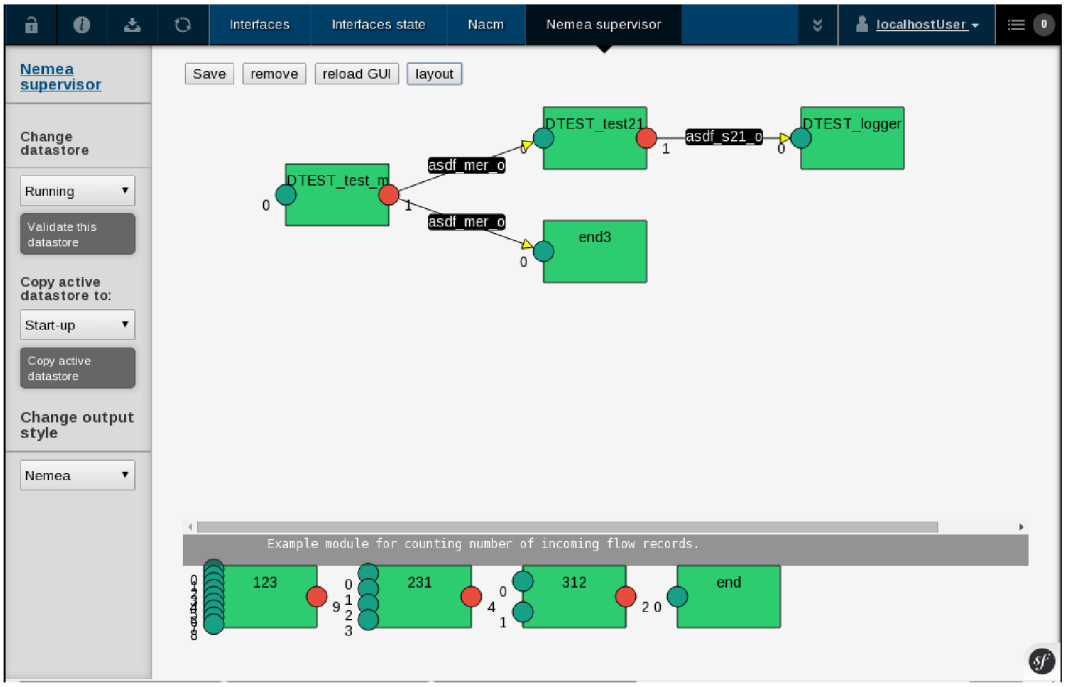


Obrázek 6.6: Všechny vybrané moduly Nemea byly po stisknutí klávesy delete odstraněny z Grafu i se všemi spojnicemi na ně navázanými.

6.1.4 Rozložení prvků vůči sobě a informační panel

Tento testovací případ testuje funkčnost tlačítka layout. V uvedeném příkladě nejsou schopnosti této funkce demonstrovány v plné míře, ale kvůli zachování posloupnosti testovacího scénáře byl tento příklad ponechán. Po stisku tlačítka byl Nemea modul DTEST_logger posunut mírně nahoru tak, aby spojnice mezi ním a modulem DTEST_test21 byla rovnoběžná s pomyslnou osou X 6.7.

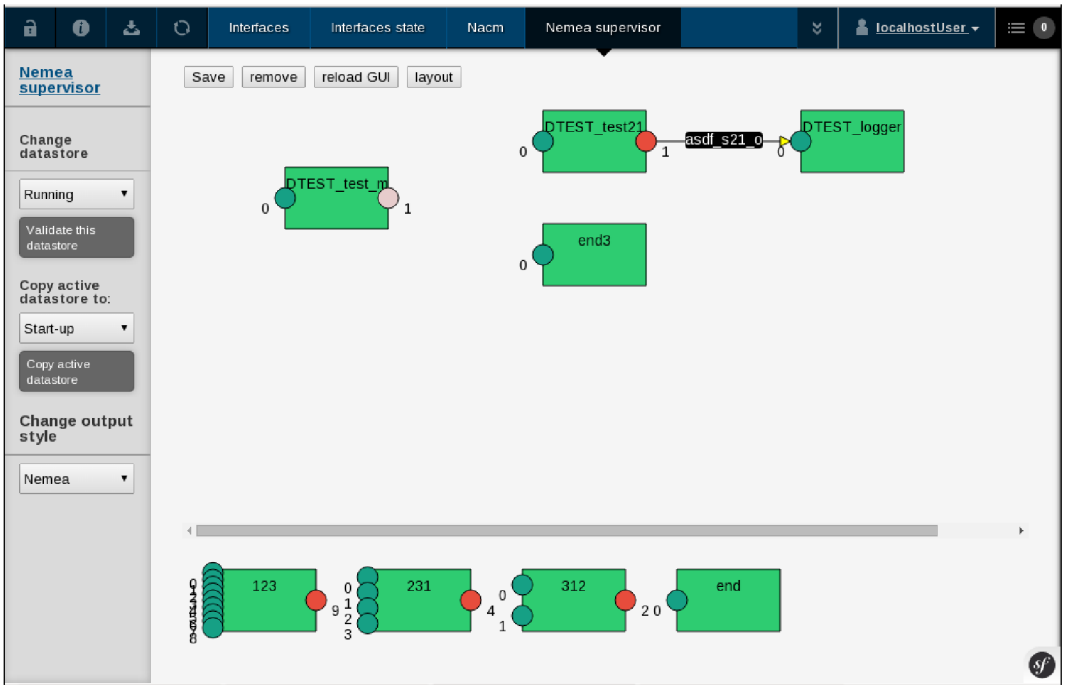
Dalším testovacím případem pak je schopnost aplikace zobrazit informační panel s popisem modulu po najetí myši na daný modul v Menu 6.7.



Obrázek 6.7: Na obrázku je vidět informační panel, který nese informace o modulu z Menu, na němž je umístěna myš uživatele. Další změnou v tomto obrázku je posun modulu DTEST_logger mírně nahoru zásluhou funkce layout.

6.1.5 Změna typu rozhraní Nemea modulu a odstranění spojnic

Při změně typu rozhraní by aplikace měla přebarvit dané rozhraní na barvu reprezentující zvolený typ a také odstranit všechny spojnice spojené s tímto rozhraním, aby nedocházelo k tomu, že jsou propojena dvě rozhraní různých typů 6.8. Při změně typu výstupního rozhraní je důležité, aby byla změněna hodnota parametrů vstupních rozhraní, která byla s tímto výstupním rozhraním spojena, na prázdný řetězec. Zda k tomu opravdu došlo, lze ověřit až z výsledného XML konfiguračního souboru, který je vygenerován po stisku tlačítka Save.



Obrázek 6.8: U modul DTEST_test_m došlo ke změně typu rozhraní, což změnilo barvu daného rozhraní a odstranilo všechny linky napojené na toto rozhraní.

6.1.6 Uložení konfigurace

Tlačítko Save slouží k uložení současné konfigurace a její odeslání na server. Test této funkcionality probíhá tak, že se do konzole prohlížeče vytiskne konfigurační XML a dochází k manuálnímu porovnání s konfigurací v diagramu. Tento testovací případ dopadl stejně jako předchozí (změna typu rozhraní Nemea modulu) pozitivně.

6.1.7 Vyhodnocení testovacího scénáře

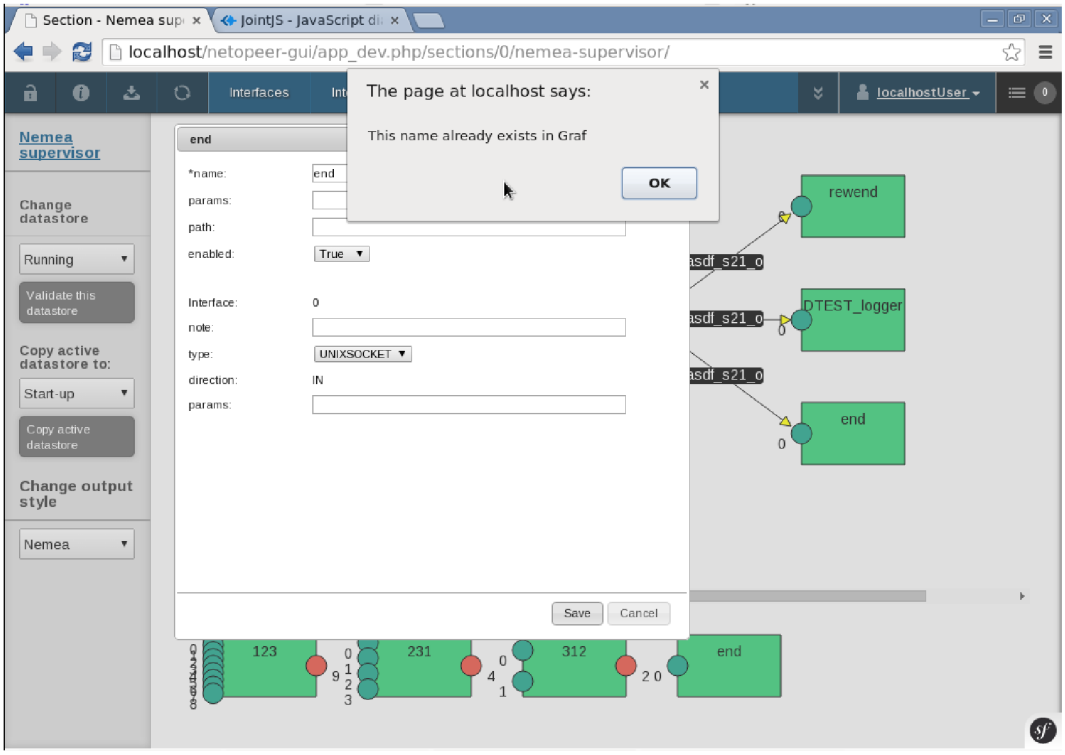
Scénář prokázal, že uživatel je schopen s vyvíjeným GUI pracovat tak, aby bylo možné vytvářet libovolné konfigurace z dostupných modulů z Menu nebo i z načtených modulů v Grafu. Uživatel může moduly mazat po jednom nebo hromadně, při přidání modulu se objeví vyskakovací okno se všemi atributy modulu, které jsou nastavitelné podle potřeb uživatele. Testovací scénář dále prokázal, že GUI rozlišuje barvy podle typu rozhraní a správně odstraní všechny spojnice na toto rozhraní připojené. Základní požadavky na GUI, kterými se tato práce zabývá, jsou tedy splněny.

6.2 Zvláštní testovací případy

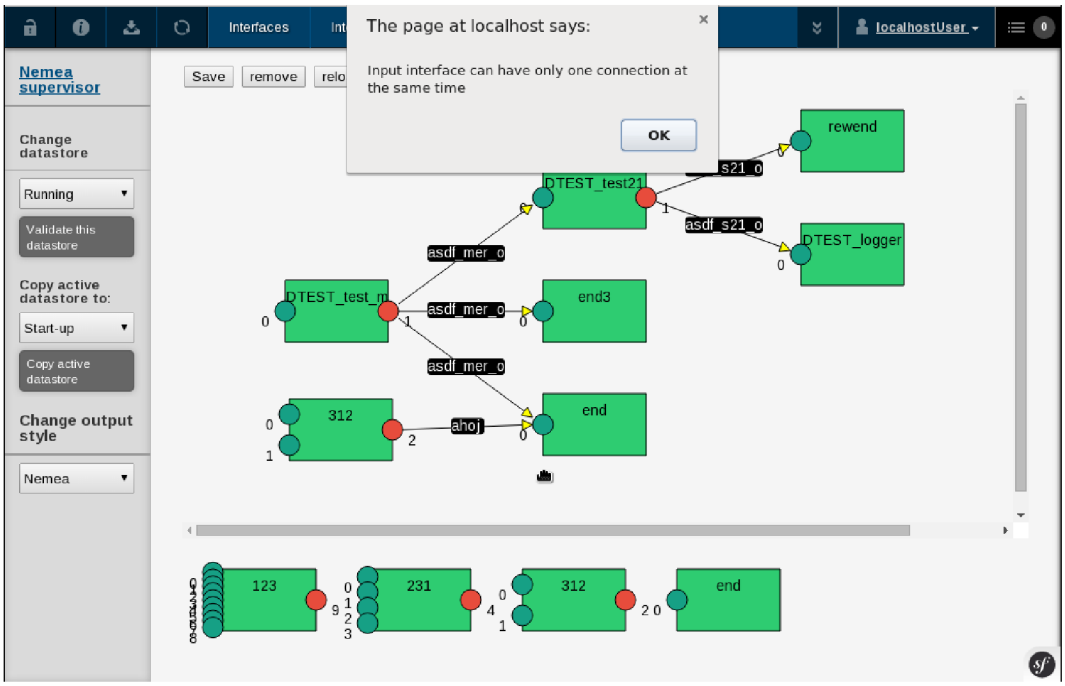
V této sekci jsou otestovány dvě situace, které nesmění nastat.. První z nich je situace, kdy se v Grafu nachází dvě stejná jména modulů. Pro ověření tohoto testovacího případu je vyzkoušeno, zda aplikace dovolí, aby byl do Grafu přidán modul se jménem, které již v Grafu je. Na obrázku 6.9 lze vidět, že aplikace reaguje chybovou hláškou a upozorní uživatele na vzniklou situaci.

Dalším testovacím případem je pokus o připojení více než jedné spojnice na vstupní

rozhraní. Očekávané chovaní testovaného GUI je takové, že upozorní uživatele při pokusu připojit k rozhraní druhou spojnici a tu následně odstraní. Výsledek lze vidět zde 6.10.



Obrázek 6.9: Na obrázku je znázorněna situace, kdy se uživatel snaží přidat modul se jménem end do Grafu. Takový modul už se v Grafu nachází, proto aplikace uživatele upozorní.



Obrázek 6.10: K rozhraní číslo 0 patřícímu modulu end se uživatel pokusil připojit druhou spojnici. GUI reaguje tak, že uživatele upozorní a následně spojnici smaže.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo implementovat grafické rozhraní pro program Supervisor, který řídí systém pro analýzu síťových dat, Nemea. V úvodu byl popsán systém Nemea a jeho moduly. Čtenářům byl přiblížen projekt Netopeer a jeho jednotlivé části. Dále byl popsán program Supervisor jakožto ovládací prvek pro systém Nemea. Na závěr druhé kapitoly byla shrnuta architektura cílové platformy.

Před samotným vývojem bylo nezbytné provést analýzu řešení s cílem zjistit, zda existují podobná řešení a jaká je jejich funkcionalita. Následoval popis JavaScriptu, což je jazyk, ve kterém je GUI napsáno. Poté byla zvážena možnost využití knihoven, které odstraňují rozdíly v interpretaci JavaScriptu jednotlivými prohlížeči. Na základě analýzy byla zvolena knihovna jQuery. Následoval výběr knihovny pro kreslení diagramů. Ze tří možností, kde každá možnost byla detailně analyzována, byla vybrána knihovna jointJS.

Následně bylo navrženo grafické rozhraní pro GUI. Po zvážení různých variant řešení bylo grafické rozhraní navrženo tak, aby vhodně zapadalo do cílové platformy a maximálně využilo plochu.

Práce dále po jednotlivých krocích popisuje samotnou implementaci vyvíjeného GUI. Implementace je rozdělena na logické celky, a to tak, aby čtenář snadno pochopil průběh vývoje GUI pro systém Nemea. Každá technologie, která byla během implementace využita, je popsána a zasazena do kontextu samotné aplikace.

Nakonec bylo GUI testováno. Za účelem testování byl využit testovací scénář navržený tak, aby pokryl co nejširší škálu funkcí, kterými GUI disponuje. Testovací scénář dokázal, že GUI lze reálně využít ke konfiguraci Nemea modulů.

Literatura

[1] Alan Page, K. J., Bj Rollison: *Jak testuje software Microsoft.* Microsoft Press A Division of Microsoft Corporation One Microsoft Way Redmond, Washington 98052-6399, 2009.

[2] Alexa, D.: *Webové uživatelské rozhraní NETCONF klienta s využitím modelu YANG.* bakalářská práce, České vysoké učení technické v Praze, 2013.

[3] Ed-Douibi, H.: 10 JavaScript libraries to draw your own diagrams, [online]. [cit. 2015-05-12]. http://modeling-languages.com/javascript-drawing-libraries-diagrams/.

[4] liberouter: Nemea, [online]. [cit. 2015-05-12]. https://www.liberouter.org/technologies/nemea.

[5] liberouter: Netopeer, [online]. [cit. 2015-05-27]. https://www.liberouter.org/technologies/netconf.

[6] liberouter: README, [online]. [cit. 2015-05-12]. https://homeproj.cesnet.cz/projects/traffic-analysis/repository/revisions/master/entry/nemea/supervisor/README.

[7] McFarland, D. S.: *Javascript & jQuery-The Missing Manual Second Edition.* O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2011.

[8] Osmani, A.: *Developing Backbone.js Applications Building Better JavaScript Applications.* O'Reilly Media Final Release Date: May 2013.

[9] WWW stránky: Apache HTTP Server, [online]. [cit. 2015-05-12]. http://cs.wikipedia.org/wiki/Apache_HTTP_Server.

[10] WWW stránky: Document Object Model (DOM), [online]. [cit. 2015-05-12]. http://www.w3.org/DOM/.

[11] WWW stránky: Getting Started, [online]. [cit. 2015-05-12]. https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started.

[12] WWW stránky: HTML: The Markup Language (an HTML language reference), [online]. [cit. 2015-05-13]. http://www.w3.org/TR/html-markup/Overview.html.

[13] WWW stránky: JavaScript Web APIs, [online]. [cit. 2015-05-12]. http://www.w3.org/standards/webdesign/script.html.

[14] WWW stránky: JointJS API, [online]. [cit. 2015-05-13]. http://jointjs.com/api.

Příloha A

Obsah CD

- Technická zpráva ve formátu PDF.
- zdrojové Latex soubory technické zprávy
- Celý adresář ModuleNemeaBundle, který obsahuje i zdrojové soubory této práce:
  - ModuleNemeaBundle/Controller - ModuleController.php
  - ModuleNemeaBundle/Resources/public/js - module-nemea.js
  - ModuleNemeaBundle/Resources/public/style - style.css
  - ModuleNemeaBundle/Resources/public/view/Module - section.html.twig

Příloha B

XML konfigurační soubory pro testování

```
<?xml version="1.0" encoding="UTF-8"?>
<nemea-supervisor>
   <available-modules>
      <search-paths>
         <path>/ava/bin/</path>
      </search-paths>
      <modules>
         <module>
            <name>123</name>
            <description>Example module for counting number
            of incoming flow records.</description>
            <number-out-ifc>1</number-out-ifc>
            <number-in-ifc>9</number-in-ifc>
            <parameter />
         </module>
         <module>
            <name>231</name>
            <description>Example module for counting
            of incoming flow records.</description>
            <number-out-ifc>1</number-out-ifc>
            <number-in-ifc>4</number-in-ifc>
            <parameter />
         </module>
         <module>
            <name>312</name>
            <description>Example module for counting number
            of incoming flow records.</description>
            <number-out-ifc>1</number-out-ifc>
            <number-in-ifc>2</number-in-ifc>
            <parameter />
         </module>
```

```
        <module>
            <name>end</name>
            <description>konec</description>
            <number-out-ifc>0</number-out-ifc>
            <number-in-ifc>1</number-in-ifc>
            <parameter />
        </module>
    </modules>
</available-modules>
<modules>
    <module>
        <name>DTEST_test21</name>
        <enabled>True</enabled>
        <path>/dns/bin</path>
        <params>localhost,asdf_s21</params>
        <trapinterfaces>
            <interface>
                <note />
                <type>UNIXSOCKET</type>
                <direction>IN</direction>
                <params>asdf_mer_o</params>
            </interface>
            <interface>
                <note />
                <type>UNIXSOCKET</type>
                <direction>OUT</direction>
                <params>asdf_s21_o</params>
            </interface>
            <interface>
                <note />
                <type>SERVICE</type>
                <direction />
                <params>service_test21</params>
            </interface>
        </trapinterfaces>
    </module>
    <module>
        <name>rewend</name>
        <enabled>True</enabled>
        <path>/home/bin/end</path>
        <params>erwer</params>
        <trapinterfaces>
            <interface>
                <note />
                <type>UNIXSOCKET</type>
                <direction>IN</direction>
                <params>asdf_s21_o</params>
            </interface>
```

```
    </trapinterfaces>
</module>
<module>
    <name>end3</name>
    <enabled>True</enabled>
    <path>/home/bin/end</path>
    <params>fwed</params>
    <trapinterfaces>
        <interface>
            <note />
            <type>UNIXSOCKET</type>
            <direction>IN</direction>
            <params>asdf_mer_o</params>
        </interface>
    </trapinterfaces>
</module>
<module>
    <name>end</name>
    <enabled>True</enabled>
    <path>/home/bin/end</path>
    <params>fdsf</params>
    <trapinterfaces>
        <interface>
            <note />
            <type>UNIXSOCKET</type>
            <direction>IN</direction>
            <params>asdf_mer_o</params>
        </interface>
    </trapinterfaces>
</module>
<module>
    <name>DTEST_test_mer</name>
    <enabled>True</enabled>
    <path>/cesta/bin</path>
    <params>localhost ,asdf_mer</params>
    <trapinterfaces>
        <interface>
            <note />
            <type>UNIXSOCKET</type>
            <direction>IN</direction>
            <params>localhost ,asdf_mer</params>
        </interface>
        <interface>
            <note />
            <type>UNIXSOCKET</type>
            <direction>OUT</direction>
            <params>asdf_mer_o</params>
        </interface>
```

```
            <interface>
                <note />
                <type>SERVICE</type>
                <direction />
                <params>service_test_mer</params>
            </interface>
        </trapinterfaces>
    </module>
    <module>
        <name>DTEST_logger</name>
        <enabled>True</enabled>
        <path>/dns/bin</path>
        <params>-t -T -w /data/xkrobo01/dns_amp_test/detected.log
        &lt;AMPLIFICATION_ALERT&gt;</params>
        <trapinterfaces>
            <interface>
                <note />
                <type>UNIXSOCKET</type>
                <direction>IN</direction>
                <params>asdf_s21_o</params>
            </interface>
            <interface>
                <note />
                <type>SERVICE</type>
                <direction />
                <params>service_dns_logger</params>
            </interface>
        </trapinterfaces>
    </module>
  </modules>
</nemea-supervisor>
```

Ukázka kódu B.1: Konfigurační XML soubor použit v testovacím scénáři.