

On-line IDE pre CUDA aplikácie

Bc. Martin Belanec

Diplomová práca
2015



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin Belanec**
Osobní číslo: **A13415**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **prezenční**

Téma práce: **On-line IDE pro CUDA aplikace**

Téma anglicky: **An On-line IDE for CUDA Applications**

Zásady pro vypracování:

1. Vytvořte literární rešerši na téma tvorby bohatých webových aplikací s interaktivním uživatelským rozhraním.
2. Prozkoumejte aktuální stav (state-of-the-art) v oboru tvorby on-line IDE.
3. Implementujte on-line webové IDE běžící na Linuxové serverové platformě založené na technologiích HTML/HTML5/JavaScript určené pro vývoj a testování aplikací vytvořených pomocí CUDA API na vzdáleném překladovém serveru.
4. Zvažte využití technologií AJAX, jQuery, Jango, programovacích jazyků PHP, Python nebo JAVA a dalších vhodných.
5. Vytvořte programovou a uživatelskou dokumentaci IDE.
6. Zdrojové kódy publikujte pod licencí GNU/GPL v2/3.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. SANDERS, Jason. CUDA by example: an introduction to general-purpose GPU programming. 1st print. Upper Saddle River: Addison-Wesley, 2011, 290 s. ISBN 978-0-13-138768-3.
2. CHENG, John, Max GROSSMAN a Ty MCKERCHER. Professional CUDA C Programming. Indianapolis, Indiana: John Wiley & Sons, Inc., 2014, 499 s. ISBN 978-1-118-73932-7.
3. ECKEL, Bruce. Myslíme v jazyku Java: knihovna zkušeného programátora. Praha: Grada, 2000, 472 s. ISBN 80-247-0027-1.
4. ECKEL, Bruce. Myslíme v jazyku Java: knihovna programátora. Praha: Grada, 2000, 431 s. ISBN 80-247-9010-6.
5. GOODMAN, Danny. JavaScript and DHTML cookbook. 2nd ed. Sebastopol, CA: O'Reilly, 2007. ISBN 05-965-1408-5.
6. SCHMITT, Christopher. CSS Cookbook. 2nd ed. Sebastopol, CA: O'Reilly, 2006, 516 s. ISBN 978-059-6527-419.
7. TACY, Adam, Robert HANSON, ESSINGTON a TÖKKE. GWT in Action. Second edition. Shelter Island, NY: Manning Publications Co., 2013. ISBN 978-193-5182-849.
8. VUKOTIC, Aleksa a James GOODWILL. Apache Tomcat 7. Berkeley, CA: Apress, 2011. ISBN 978-143-0237-242.

Vedoucí diplomové práce:

Ing. Michal Bližňák, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

6. února 2015

Termín odevzdání diplomové práce:

15. května 2015

Ve Zlíně dne 6. února 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



L.S.



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu


Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s tím, že vyrovnaní případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, 15.5.2015


.....
podpis diplomanta

ABSTRAKT

Cieľom diplomovej práce je vytvorenie a implementácia on-line webového IDE bežiaceho na Linuxovej serverovej platforme a určené pre vývoj a testovanie aplikácií vytvorených pomocou CUDA API na vzdialenom prekladovom serveri. Teoretická časť práce sa zaoberá základnými pojmami v oblasti tvorby bohatých webových aplikácií a vhodnými vývojevými technológiami. Následne sú rozobrané technológie a programovací jazyk využitý pri tvorbe webového IDE. Praktická časť práce rozoberá definície a požiadavky na tvorbu webového IDE, jeho implementáciu a odporúčania k ďalšiemu vývoju.

Kľúčové slová:

CUDA API, on-line IDE, web, Linux, GWT, Java, vzdialený prekladový server

ABSTRACT

The aim of this thesis is to create and implement of on-line web IDE running on Linux based server platform. It is intended for development and testing applications built using CUDA API on remote build server. In the theoretical section, there are explained basic concepts of development of rich web applications and suitable development tools. Subsequently presents the analysis of technology and programming language used for creating web IDE. In the practical part, this work deals with the definition and requirements for creating web IDE, implementation and recommendations for further development.

Keywords:

CUDA API, on-line IDE, web, Linux, GWT, Java, remote build server

Na tomto mieste by som sa chcel poďakovať vedúcemu diplomovej práce pánovi Ing. Michalovi Bližňákovi, Ph.D. za odbornú pomoc, cenné rady a v neposlednom rade za trpezlivosť, ktorú so mnou mal. Ďalej by som sa chcel poďakovať Ing. Františkovi Špačkovi, ktorý spolupracoval na vývoji programovacieho prostredia, za jeho prínosy k práci, rady a vynaložený čas. Taktiež by som sa chcel poďakovať rodine za podporu behom štúdia.

Prehlasujem, že odovzdaná verzia diplomovej práce a verzia elektronická nahraná do IS/STAG sú totožné.

OBSAH

ÚVOD	10
I TEORETICKÁ ČASŤ	11
1 WEBOVÉ APLIKÁCIE	12
1.1 KLASICKÉ WEBOVÉ APLIKÁCIE.....	13
1.2 BOHATÉ WEBOVÉ APLIKÁCIE.....	13
1.2.1 História.....	13
1.2.2 Výhody.....	13
1.2.3 Nevýhody.....	14
1.2.4 Rozhodovacie kritéria pre výber RIA technológie.....	14
1.2.5 Obmedzenia a požiadavky	15
2 WEBOVÉ TECHNOLOGIE	17
2.1 KLIENSKA VRSTVA	17
2.1.1 HTML	17
2.1.2 CSS.....	17
2.1.3 XML.....	18
2.1.4 JavaScript	18
2.1.5 AJAX.....	19
2.1.6 Flash.....	20
2.2 SERVEROVÁ VRSTVA	21
2.2.1 PHP	21
2.2.2 Java.....	21
2.2.3 Python	21
3 WEBOVÉ FRAMEWORKY	23
3.1 JQUERY	23
3.2 EXTJS	24
3.3 DJANGO.....	24
3.4 ECLIPSE REMOTE APPLICATION PLATFORM	25
3.5 GOOGLE WEB TOOLKIT.....	26
4 TECHNOLOGIA CUDA	27
4.1 ARCHITEKTÚRA.....	27
4.1.1 Použitie architektúry	28
4.1.2 Vlastnosti architektúry Kepler	29
4.1.3 Programovací model	31
4.1.4 Pamäťový model	31
4.2 PREKLAD APLIKÁCIE PRE GPU	31
4.3 CUDA C	32
II PRAKTICKÁ ČASŤ	35
5 POUŽITÉ TECHNOLOGIE	36
5.1 WEBOVÝ FRAMEWORK.....	36
5.1.1 Klientske rozhranie	37
5.1.2 GWT RPC	37
5.1.3 JSNI.....	39
5.1.4 Sencha GXT	39

5.2	VÝVOJOVÉ PROSTREDIE	40
5.3	WEBOVÝ SERVER	40
5.4	POUŽITÉ KNIŽNICE A NÁSTROJE TRETÍCH STRÁN	41
5.4.1	Apache Commons	41
5.4.2	Apache HttpComponents	41
5.4.3	Google Gson.....	42
5.4.4	SLF4J	42
5.4.5	Zip4j	42
5.4.6	Zt-zip	42
5.4.7	JavaMail	42
5.4.8	ACE.....	42
5.4.9	Exuberant Ctags	43
5.4.10	CMake	43
6	ANALÝZA A NÁVRH RIEŠENIA	44
6.1	SÚČASNÝ STAV.....	44
6.1.1	Výhody a nevýhody súčasného riešenia	45
6.2	INOVÁCIA RIEŠENIA	45
6.3	DEFINÍCIA POŽIADAVIEK	46
6.3.1	Funkčné požiadavky.....	46
6.3.2	Nefunkčné požiadavky.....	47
6.4	MODEL PRÍPADOV UŽITIA	47
6.4.1	Aktéri systému	47
6.4.2	Diagram prípadov užitia.....	48
6.5	NÁVRH RIEŠENIA	51
7	IMPLEMENTÁCIA.....	53
7.1	TVORBA APLIKÁCIE	53
7.2	ŠTRUKTÚRA APLIKÁCIE	56
7.2.1	Klientske rozhranie	57
7.2.2	Zdieľané objekty	60
7.2.3	Serverové rozhranie	61
7.2.3.1	Správa pracovného prostredia.....	62
7.2.3.2	Správa prihlasovania používateľa.....	67
7.2.3.3	Sťahovanie a nahrávanie súborov	67
7.3	NASADENIE APLIKÁCIE.....	68
8	ZHODNOTENIE.....	69
8.1	ZHODNOTENIE CUDA ON-LINE IDE.....	69
8.2	ODPORÚČANIA K BUDÚCEMU VÝVOJU	69
8.2.1	Implementácia interakcie s CUDA aplikáciou.....	69
8.2.2	Rozšírenie o ďalšie vzdialené servery.....	70
8.2.3	Rozšírenie automatického dopĺňania kódu	70
8.2.4	Obsluha chybových oznámení	70
8.2.5	Možnosť rozšírenia na rôzne programovacie jazyky	70
	ZÁVER	71
	ZOZNAM POUŽITEJ LITERATURY.....	72
	ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....	75

ZOZNAM OBRÁZKOV	78
ZOZNAM TABULIEK	80
ZOZNAM PRÍLOH.....	81

ÚVOD

Webové aplikácie tvoria v dnešnej dobe silné odvetvie tvorby aplikácií. Poskytujú jednoduchý prístup pomocou webového prehliadača bez potreby inštalácie do systému. Preto sa čoraz viac objavujú webové aplikácie na báze desktopových a sú s obľubou využívané používateľmi.

Z toho dôvodu ma zaujala téma tvorby on-line programovacieho prostredia pre tvorbu CUDA (Compute Unified Device Architecture) aplikácií a nahradenie využívaných technológií v kurze programovania na grafických kartách. Doterajšie riešenie programovania CUDA aplikácií obsahuje komplikácie v nastavení prekladu na vzdialenom serveri. Práca teda rieši implementovanie vytvoreného programovacieho prostredia na vzdialený prekladový server s grafickou kartou Nvidia Tesla, ktoré má zjednodušiť používanie IDE tak, aby študenti mohli spustiť toto vytvorené webové IDE (Integrated Development Environment) a ihneď prísť k tvorbe aplikácií.

Vďaka neustálemu vývoju grafických kariet a vysokému výpočtovému výkonu grafických procesorových jednotiek, sú grafické karty využívané pre výpočtovo náročné paralelné výpočty. Neustály a silný dopyt po spracovaní náročných výpočtov dostáva do popredia tvorbu takýchto aplikácií, či už pri 3D renderovaní, spracovaní obrazu, zvuku, ale aj spracovaní signálov, simulácií, výpočtovej ekonomiky, biológie a ďalších. Je preto dôležité, aby boli študenti oboznámení s návrhom, tvorbou a testovaním takýchto aplikácií.

Diplomová práca sa teda bude zaoberať zjednodušením ich tvorby pomocou on-line programovacieho prostredia. V teoretickej časti budú popísané webové technológie, ktoré môžu byť použité pre tvorbu bohatých webových aplikácií a tiež budú poskytnuté základné informácie o rozšírení CUDA C pre tvorbu CUDA aplikácií. V praktickej časti budú rozobrané technológie využité pre tvorbu webového programovacieho prostredia, zadefinované požiadavky a modely prípadov užitia pre jeho implementáciu. Následná implementácia poskytne výslednú aplikáciu, kde bude popísaná jej kompletná štruktúra a spôsob nasadenia na vzdialený prekladový server.

Výstupom práce bude CUDA On-line IDE dostupné na určenej webovej adrese, ktoré pokrýje všetky zadefinované požiadavky na tento systém.

I. TEORETICKÁ ČASŤ

1 WEBOVÉ APLIKÁCIE

Webová aplikácia je softvérový program, ktorý beží na webovom serveri. Prístup k webovej aplikácii je zabezpečený pomocou webového prehliadača, čo je zmena oproti klasickej desktopovej aplikácii, ktorá je spúšťaná operačným systémom.

Výhody webovej aplikácie v porovnaní s desktopovou sú nasledovné:

- aplikácia beží v prehliadačoch, nie je potrebné vyvíjať multiplatformové riešenie,
- odpadá nutnosť inštalácie aplikácie,
- nie sú potrebné softvérové aktualizácie, užívateľ má aktuálnu verziu ihneď po aktualizovaní aplikácie na webovom serveri,
- ponúka dôsledné užívateľské prostredie,
- dáta sú distribuované vzdialene, užívateľ má teda prístup k dátam z rôznych zariadení, odpadá nutnosť prenosu dát.

Nevýhody webovej aplikácií sú:

- bežia pomalšie,
- aplikácia má obmedzený prístup k systémovým zdrojom ako je CPU, pamäť, súborový systém,
- pri neočakávanom vypnutí prehliadača je veľká pravdepodobnosť straty neuložených dát,
- aktualizácia prehliadača môže spôsobovať nesúlad alebo neočakávané problémy.

[1]



Obr. 1: Komponenty webovej aplikácie [2]

1.1 Klasické webové aplikácie

Klasické webové aplikácie sústreďujú všetky aktivity ku klient server architektúre s tenkým klientom. To znamená, že všetky procesy sú vykonávané na serveri a na klientovi je zobrazovaný statický obsah. Nedostatkom toho je, že všetky požiadavky musia byť smerované na server, ktorý vyžaduje, aby naň boli smerované dáta, na ktoré odpovedá, čo spôsobí obnovenie stránky na klientovi. Tento synchronný spôsob komunikácie vytvára negatívny dopad na použiteľnosť. Klasické webové aplikácie sú prevažne tvorené na zobrazovanie textových a obrazových informácií. [3]

1.2 Bohaté webové aplikácie

Bohatá webová aplikácia - RIA (Rich Internet Application) je typ webovej aplikácie, ktorá má vlastnosti a funkcie klasických desktopových aplikácií. V RIA zvyčajne dochádza k prenosu potrebných dát pre užívateľské prostredie k webovému klientovi, ale ponecháva väčšinu dát na aplikačnom serveri. RIA predstavuje prídavnú klientsku vrstvu, ktorá umožňuje spustenie kódu na strane klienta pre poskytnutie lepšieho výkonu. Takáto aplikácia je interakčne založená. Vďaka interakcii vytvára wau efekt pre užívateľa. Bohatá webová aplikácia používa asynchrónnu komunikáciu na získanie dát zo servera, čo nevyžaduje znovu načítanie stránky. Aplikácia je tak rýchlejšia a viac užívateľsky prívetivá. Hlavná stránka aplikácie je načítaná len raz, obsah je dynamicky menený v nadväznosti na užívateľskú interakciu. [3]

1.2.1 História

Pojem bohatá webová aplikácia bol predstavený v Macromedii v marci 2002. Koncept RIA aplikácii však bol známy už mnoho rokov predtým pod rôznymi názvami:

- Remote Scripting,
- X Internet,
- Rich Clients,
- Rich Web Application.

1.2.2 Výhody

Hlavnými výhodami bohatých webových aplikácií v porovnaní s klasickými sú:

- využívanie klientskeho CPU,

- ponúka možnosti používateľského rozhrania v reálnom čase,
- používanie bohatších systémových funkcionalít, ako operácia ťahaj a pusti,
- výpočty na klientskej strane, bez nutnosti posielania dát na server,
- výpočtové zdroje klienta a servera sú vyrovnannejšie, takže server nemusí byť veľmi vyťažený webovou aplikáciou, čím sa mu uvoľnia zdroje pre spravovanie viacerých súbežných relácií,
- sieťová prevádzka môže byť výrazne znížená, pretože klient môže byť viacej inteligentnejší ako webový prehliadač v rozhodovaní čo zaslať na server. [4]

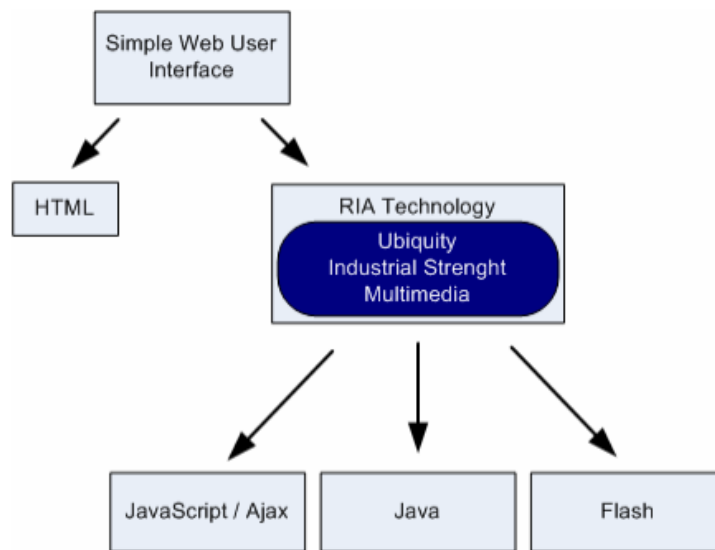
1.2.3 Nevýhody

Nedostatkami RIA technológie sú:

- aplikácia RIA má obmedzený prístup k systémovým zdrojom, ak je prístup k očakávaným zdrojom zakázaný, aplikácia RIA nemusí fungovať správne,
- často je nutné, aby bol povolený skriptovací jazyk ako JavaScript, ak je vypnutý, webová aplikácia nebude fungovať správne alebo nepôjde vôbec,
- v závislosti na programovacom jazyku na strane klienta môže byť klientsky kód vykonávaný pomalšie, čo obmedzuje funkcionalitu,
- inteligencia na strane klienta musí byť dodaná zo servera a aj keď sa väčšinou ukladá do vyrovnávacej pamäte, musí sa preniesť aspoň raz, čo môže niekedy trvať veľmi dlho. [4]

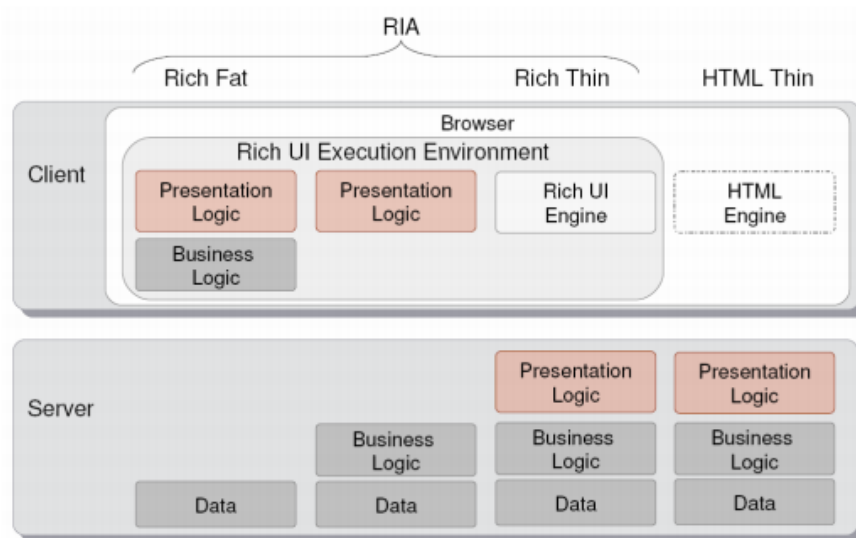
1.2.4 Rozhodovacie kritéria pre výber RIA technológie

Na Obr. 2 je znázornený typický rozhodovací strom pre výber správneho RIA prístupu na základe danej aplikácie.



Obr. 2: Rozhodovací strom pre výber RIA technológie [5]

Na základe množstva logík, ktoré sa vykonávajú na klientovi, vznikajú rôzne typy riešenia RIA technológií, a to buď tlstý alebo tenký klient. Na Obr. 3 sú znázornené typy RIA architektúr.



Obr. 3: Typy architektúr RIA technológií [5]

1.2.5 Obmedzenia a požiadavky

Bohaté webové aplikácie vyžadujú pre správne fungovanie moderné webové prehliadače. Prehliadač musí mať vo väčšine prípadov povolený javascriptový engine, teda virtuálny stroj, ktorý interpretuje a vykonáva JavaScript, používať techniky ako XMLHttpRequest

pre komunikáciu klient server, DOM skriptovanie a pokročilé CSS pre bohaté užívateľské rozhranie.

Ak chceme, aby RIA bežala vo všetkých hlavných prehliadačoch ako Internet Explorer, Chrome, Firefox a iné, rozdiely medzi prehliadačmi môžu spôsobovať problémy.

Prídavná interaktivita môže vyžadovať technické postupy, ktoré obmedzujú aplikačnú prístupnosť.

Používatelia taktiež často očakávajú od všetkých webových aplikácií rovnakú funkcionality. Preto môže nastať problém pri prijatí niektorých obvyklých funkcií. Pre príklad použite tlačidlo Späť v prehliadači, môže spôsobiť odlišné alebo nežiaduce správanie. [4]

2 WEBOVÉ TECHNOLOGIE

Existuje veľa webových technológií, kde každá technológia ma svoje špecifiká a oblasť použiteľnosti. Medzi týmito technológiami sú zahrnuté značkovacie jazyky, programovacie jazyky a ďalšie techniky dôležité k vývoju webovej aplikácie. V tejto sekcii sú popísane základné technológie, používané pri tvorbe webových aplikácií.

2.1 Klientska vrstva

2.1.1 HTML

HTML je skratka pre Hypertext Markup Language, teda hypertextový značkovací jazyk. HTML je primárne používaný značkovací jazyk pre webové aplikácie. Prostredníctvom neho je v prehliadačoch určené, kde a čo sa má na displeji zobrazit'. HTML podporuje zobrazovanie obrázkov a iných multimédií. Tento jazyk popisuje štruktúru webu a sémantický obsah webového dokumentu. Obsah webovej stránky je tvorený pomocou značiek, ktoré formujú bloky stránky. Poslednou špecifikáciou HTML je HTML5. [6, s. 14-27]

<pre><!DOCTYPE html> <html> <body> <h1>Example</h1> <p>HTML example.</p> </body> </html></pre>	Example HTML example.
--	---------------------------------

Obr. 4: Tvorba štruktúry stránky pomocou HTML

2.1.2 CSS

Kaskádové štýly alebo CSS je formátovací jazyk, ktorý poskytuje možnosť zmeniť rôzne vlastnosti textu, objektov na webových stránkach. CSS umožňuje formátovanie prvkov, ktoré sú implementované značkovacím jazykom ako HTML. Tento jazyk poskytuje dizajnérovi webovej stránky rozšírené možnosti navrhovania prezentačnej vrstvy aplikácie. Pomocou kaskádových štýlov je možné určovať pozície elementov na stránke, skrývať ich, zvýrazňovať, vytvárať grafické prechody a animácie alebo meniť základné zobrazovacie vlastnosti prehliadačov. Kaskádové štýly bývajú zvyčajne zapísané priamo v HTML súbore, uzatvorené párovou značkou *style*, pri jednotlivých značkách v atribúte *style* alebo v samostatnom súbore, ktorý je odkazovaný v HTML súbore. Zámenou kaskádových štýlov

je možné kompletne zmeniť vzhľad celej webovej stránky. Poslednou špecifikáciou CSS je CSS3. [7, s. 17-20]

```
body {  
    background-color: #d0e4fe;  
}  
  
h1 {  
    color: orange;  
    text-align: center;  
}  
  
p {  
    font-family: "Times New Roman";  
    font-size: 20px;  
}
```

Obr. 5: CSS formátovanie

2.1.3 XML

XML predstavuje Extensible Markup Language, ktorý je podobný značkovaciemu jazyku HTML. XML je na rozdiel od formátu HTML používaný na popis štruktúry dát, nie na zobrazovanie dát. Značky v XML nie sú definované tak ako u HTML, ale používateľ ich musí definovať. Tento jazyk popisuje štruktúru dát, samotné dáta, nevykonáva žiadnu činnosť a tvorí tak doplnok k HTML. XML je softvérovo a hardvérovo nezávislý nástroj na prenášanie informácií a má preto stále miesto v IT systémoch. [8]

2.1.4 JavaScript

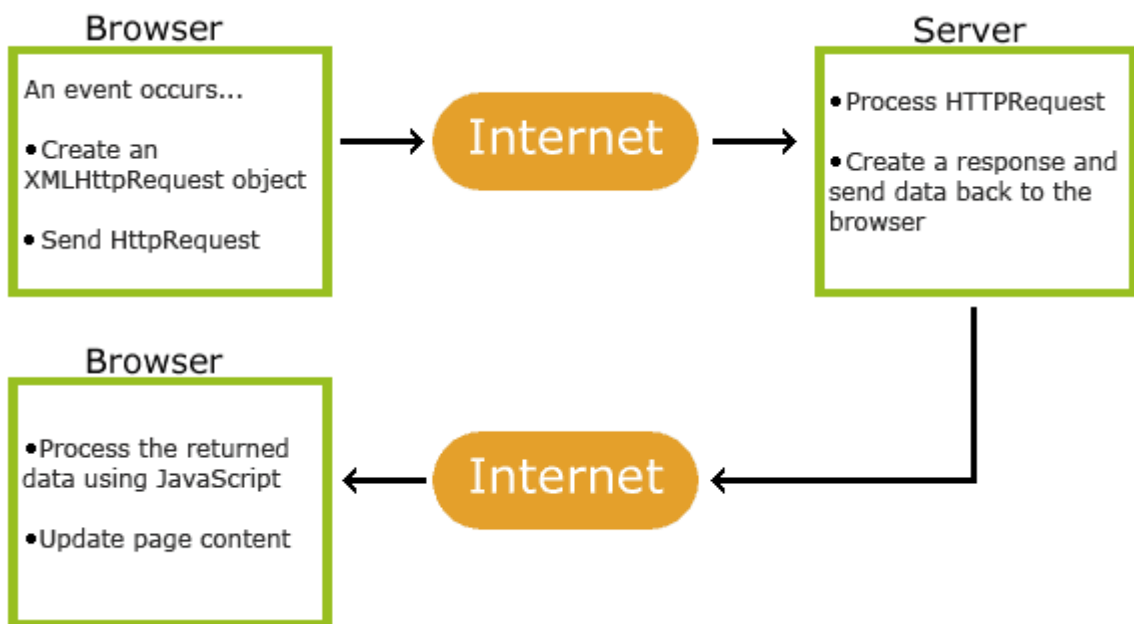
JavaScript je interpretovaný skriptovací jazyk používaný pre vytváranie webových stránok, ktoré sú v interakcii s užívateľom a odpovedajú na jeho akcie. V kombinácii s HTML a CSS vytvárajú dynamické HTML stránky. JavaScript je prevažne používaný na klient-skej strane, kde je súčasťou HTML kódu. Pomocou JavaScriptu je možné manipulovať s DOM modelom stránky, čo je využívané pri tvorbe interaktívneho rozhrania. Vykonávané sú vizuálne animácie, validácia, dynamická zmena vzhľadu, uľahčenie administrácie. JavaScript môže meniť obsah HTML, jeho atribúty a taktiež formátovanie. Umiestnenie JavaScriptu v HTML súbore je v značke *script* a toto umiestnenie nie je striktné definované. JavaScript obsahuje štandardné knižnice objektov ako pole, dátum, štruktúry a iné. Z názvu programovacieho jazyku je možné dôjsť k názoru, že JavaScript je odvodený od programovacieho jazyku Java. Nie je to tak. Medzi Javou a JavaScriptom sú významné rozdiely, ktoré sú popísané v Tab. 1. [9]

JavaScript	Java
Objektovo orientovaný.	Objektovo orientovaný.
Nie sú rozdiely medzi typmi objektov. Dedičnosť je vykonávaná prostredníctvom mechanizmu prototypovania.	Založený na triedach. Objekty sú rozdelené do tried a inštancií s dedičnosťou prostredníctvom hierarchie tried.
Vlastnosti a metódy môžu byť pridávané k objektu dynamicky.	Triedy a inštancie nemôžu mať vlastnosti a metódy pridávané dynamicky.
Premenné dátové typy nie sú deklarované (dynamické vytváranie typov).	Premenné dátové typy sú deklarované (statické vytváranie typov).
Nemožnosť automaticky zapisovať na pevný disk.	Nemožnosť automaticky zapisovať na pevný disk.

Tab. 1: Porovnanie programovacích jazykov JavaScript a Java [9]

2.1.5 AJAX

AJAX (Asynchronous JavaScript and XML) je webová technika pre tvorbu rýchlej, interaktívnej webovej aplikácie. Cieľom tejto techniky je, aby stránky vymieňali v pozadí meno dát so serverom a nemuseli tak byť vždy načítané, ak užívateľ vykoná požiadavku na zmenu. Týmto je zvýšená interaktivita webovej stránky, jej rýchlosť, funkcionálna a použiteľnosť. AJAX technika upravuje stránku asynchrónne na pozadí a namiesto obnovenia celej len jej časť. [10]



Obr. 6: Asynchronná komunikácia zo serverom [10]

AJAX technika je založená na internetových štandardoch a ich kombináciách. Využíva *XMLHttpRequest* objekt na asynchrónnu výmenu dát so serverom cez HTTP a po doručení odpovede upravuje DOM a zobrazí výsledok. Tento proces je znázornený na Obr. 6. AJAX je multiplatformová technika použiteľná na rôznych operačných systémoch, architektúrach a webových prehliadačoch. [10]

2.1.6 Flash

Flash je softvérová technológia pre vytváranie a správu interaktívnych webových aplikácií a animácií. Môže byť použitá v rôznych častiach návrhu. Flash webové stránky nie sú závislé od webových prehliadačov, na všetkých vyzerajú rovnako. Pre zobrazenie takejto aplikácie musí byť v prehliadači nainštalovaný zásuvný modul, ktorý túto aplikáciu zobrazuje. Funkcionalita technológie je založená na programovacom jazyku *ActionScript*, ktorý je jej súčasťou. Nevýhodou týchto aplikácií je, že čerpajú veľa systémových prostriedkov a ich beh je rôzne optimalizovaný na rôznych platformách. Ďalšou nevýhodou je, že aplikácie neumožňujú prístup k zdrojovému kódu. [11]

2.2 Serverová vrstva

2.2.1 PHP

PHP je serverový skriptovací jazyk pre tvorbu dynamických a interaktívnych webových aplikácií. PHP skripty sú vykonávané na strane servera. Táto technológia je široko používaná, open-source, teda voľná pre stiahnutie. PHP súbory obsahujú HTML, CSS, JavaScript a PHP kód. Technológia je ľahká na použitie a je používaná prevažne na Linuxových a Unixových platformách, Apache serveroch s kombináciou MySQL alebo PostgreSQL databáz. Môže tiež bežať aj na Windows platforme s nainštalovaným interpretom. PHP technológia umožňuje generovať dynamický obsah stránky, vytvárať, editovať, mazať súbory na serveri, spravovať cookies, databáze, užívateľský prístup a iné. [12]

2.2.2 Java

Java je objektovo orientovaný jazyk navrhnutý pre používanie na internete. Bol predstavený firmou Sun Microsystems v roku 1995. Patrí medzi najrozšírenejšie programovacie jazyky vôbec. Java predstavila spôsob ako webový vývojári mohli zahrnúť animácie a dynamické elementy do webových stránok. Java kód je určený pre serverové spracovanie, môže byť ale spustený na klientovi pomocou JVM, teda Java virtual machine. Java platforma zabezpečuje kompatibilitu medzi operačnými systémami.

Java EE (Enterprise Edition) je štandard pre vývoj prenosných, škálovateľných, bezpečných aplikácií na strane servera. Java EE poskytuje okrem základných Java SE (Standard Edition) funkcií taktiež webové služby, model komponentov, riadenie a API pre komunikáciu a pre implementovanie podnikových webových aplikácií a informačných systémov.

JSP (JavaServer Pages) technológia umožňuje vývojárom a dizajnérom rýchlo vyvíjať a ľahko udržiavať dynamické, informačne bohaté webové stránky využívajúce súčasné podnikové systémy. JSP sú platformovo nezávislé webové aplikácie, ktoré oddelujú používateľské rozhranie od generácie obsahu a umožňujú tak ľahko meniť rozvrhnutie stránky.

Java platforma poskytuje tiež servlety, ktoré rozširujú a obohacujú webové servery. Servlety poskytujú platformovo nezávislé metódy pre vytváranie webových aplikácií. [13]

2.2.3 Python

Python je interpretovaný, objektovo orientovaný programovací jazyk s dynamickou sémantikou. Tento programovací jazyk využíva dynamické vytváranie typov s dynamickými

väzbami, ktoré sú určené pre rýchly vývoj aplikácií. Python je veľmi jednoduchý na naučenie syntaxe, ľahko čitateľný, a tak urýchľuje vývoj a redukuje náklady. Tento programovací jazyk bol navrhnutý pre tvorbu rozsiahlych a plnohodnotných aplikácií. Python je open-source projekt a je dostupný pre väčšinu používaných operačných systémov. [14]

3 WEBOVÉ FRAMEWORKY

Webový aplikačný framework je kolekcia balíčkov a modulov pre tvorbu webových aplikácií a služieb, bez potreby obsluhy nízko úrovňových detailov ako protokoly, sockety alebo správa procesov. Frameworky poskytujú podporu veľkému množstvu aktivít ako správa session, uchovávanie dát, interpretovanie požiadaviek a odpovedí (requests and responses) a iných. Taktiež uľahčujú prácu vývojárom, pretože poskytujú preddefinované objekty pre rýchlejšiu tvorbu aplikácií. V nasledujúcej časti bude rozobraných zopár používaných webových frameworkov, ich architektúra a možnosti.

3.1 jQuery

jQuery UI je javascriptový framework postavený na základoch jQuery javascriptovej knižnice. Tento framework je zameraný na tvorbu používateľského rozhrania, efektov, widgetov a vďaka nemu je možné zostaviť vysoko interaktívne webové aplikácie. jQuery UI pomáha zrýchliť programovanie používateľského rozhrania a zjednodušiť jeho tvorbu. Framework je podporovaný všetkými modernými prehliadačmi a tak vývojár nemusí riešiť problémy s kompatibilitou a rozličným vzhľadom aplikácie v rámci prehliadačov. Javascriptová knižnica jQuery zjednodušuje použitie JavaScriptu na stránkach. Výhodou je, že viacriadkový kód napísaný v JavaScripte je možné v jQuery volať jednou metódou, teda jedným riadkom kódu. Pomocou tejto knižnice je ľahšia použiteľnosť komplikovaných vecí z JavaScriptu, napríklad AJAX-ové volania, práca s DOM objektmi, CSS formátovaním, efektmi. jQuery frameworky sú veľmi populárne a používané rôznymi veľkými spoločnosťami ako je Google, Microsoft a iné. jQuery UI je vydávané pod MIT licenciou. Prvé publikovanie bolo v roku 2007. [15]



Obr. 7: Logo javascriptového frameworku jQuery UI

3.2 ExtJS

Sencha ExtJS je MVC/MVVM javascriptový framework pre tvorbu bohatých webových aplikácií. Framework je multiplatformový a využíva techniky ako AJAX, DHTML a DOM skriptovanie. Zahŕňa podporu HTML5 technológie v moderných prehliadačoch. Poskytuje veľké množstvo zostavených widgetov, ktoré sú jednoduché pre použitie. Medzi ne patria okná, panely, toolbary, dialógové okná a i., a môžu byť užívateľsky ľahko rozšíriteľné. Framework obsahuje robustné dátové balíčky, ktoré spracovávajú dáta z rôznych back-endových zdrojov. ExtJS poskytuje vlastné verzie rôznych javascriptových objektov ako pole, dátum, objekt a i., ktoré sú použiteľné napriek všetkými prehliadačmi. Poskytuje tiež manažment session a minimalizuje tak komunikáciu so serverom. [16, s. 3-4]



Obr. 8: Logo javascriptového frameworku ExtJS

3.3 Django

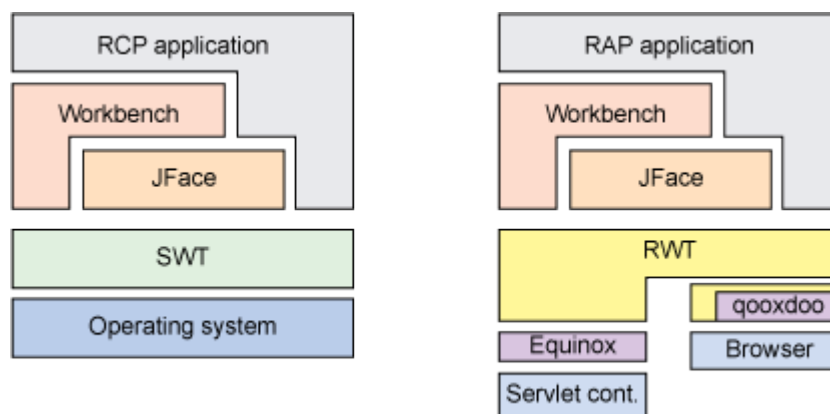
Django je open-source MVC webový aplikačný framework postavený na programovacom jazyku Python, publikovaný pod BSD licenciou v roku 2005. Django poskytuje tvorbu komplexných, databázovo zameraných bohatých webových aplikácií. Poskytuje rýchly vývoj aplikácie vďaka veľkému množstvu dostupných zásuvných modulov a dopredu pripravených objektov. Framework poskytuje administrátorské rozhranie pre jednoduchú správu CRUD operácií, používateľskú autentifikáciu, RSS prúdy, používateľské komentáre a i. Pomáha vývojárom zabezpečiť bezpečnostné diery ako SQL injection, cross-site scripting a ďalšie. Je vhodný na tvorbu systémov pre správu obsahu (CMS). Django framework je odporúčaný pre stabilitu projektov, výkon a veľkú používateľskú komunitu. Dokumentácia a návody sú jedny z najlepších v oblasti tvorby webových aplikácií. [17]



Obr. 9: Logo Python frameworku Django

3.4 Eclipse Remote Application Platform

RAP (Eclipse Remote Application Platform) je open-source Java framework pre tvorbu bohatých webových aplikácií s použitím Eclipse vývojového modelu, čo znamená, že dovoľuje vytvárať webové AJAX-ové aplikácie pomocou Java knihozien a Eclipse API. RAP pre webové aplikácie je ako RCP (Rich Client Application) pre desktopové. Rozdiel medzi RCP a RAP platformou je v použití RWT (RAP Widget toolkit) namiesto SWT (Standard Widget Toolkit), čo je vidieť v Obr. 10. Výhodou tohto je, že kód medzi RCP a RAP aplikáciami môže byť zdieľaný, a tak sa zníži záťaž na tvorbu webovej a desktopovej aplikácie. RWT sa skladá z Java objektov na strane servera a javascriptových objektov na strane klienta. Aplikčný kód pracuje s Java inštanciami a používateľ webovej aplikácie s javascriptovými inštanciami. RWT poskytuje model udalostí, ktoré obsluhujú používateľské akcie. [18]



Obr. 10: RAP architektúra [18]

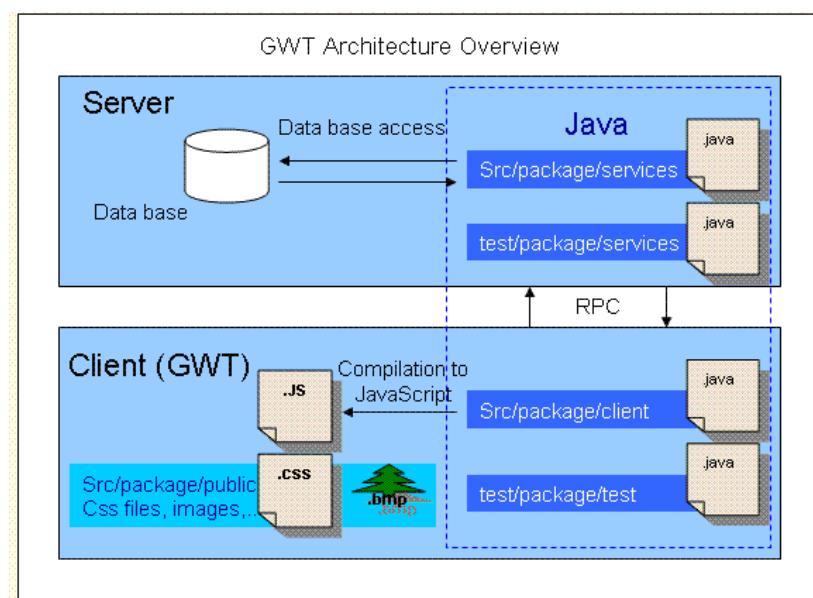
Architektúra RAP je pomenovaná ako server-centric AJAX framework, čo znamená, že všetky procesy a akcie sú vykonávané na serverovej strane. Klientska strana slúži len na prezentovanie obsahu, čo zjednodušuje prácu pre prehliadače. Tie však čakajú na vykonanie inštrukcií zo servera.



Obr. 11: Logo frameworku Remote Application Platform

3.5 Google Web Toolkit

GWT (Google Web Toolkit) je open-source vývojáři nástroj pro tvorbu a optimalizování bohatých webových aplikací. Tento nástroj využívá pro tvorbu aplikace programovací jazyk Java, tak jako na straně serveru, tak i na straně klienta. Odpadá tak zdělavý vývoj v programovacím jazyku JavaScript, technologiích XMLHttpRequest, HTML a i. Nástroj vykonává konverzi jazyka Java do JavaScriptu pomocí kompilátorů, které vytvoří optimalizovaný javascriptový kód, běžící ve všech webových prohlížečích. GWT SDK poskytuje jádro Java API a různé widgety, což umožňuje rychle vytváření uživatelského rozhraní. GWT umožňuje i na straně klienta použít značkovací jazyk HTML, formátovací jazyk CSS a nativní javascriptový kód. [19]



Obr. 12: GWT architektúra [20]

Na rozdíl od Eclipse RAP, architektúra GWT je client-centric AJAX framework, což znamená, že GWT staticky zhromažďuje všechny widgety vopred, což snižuje komunikáciu so serverom.



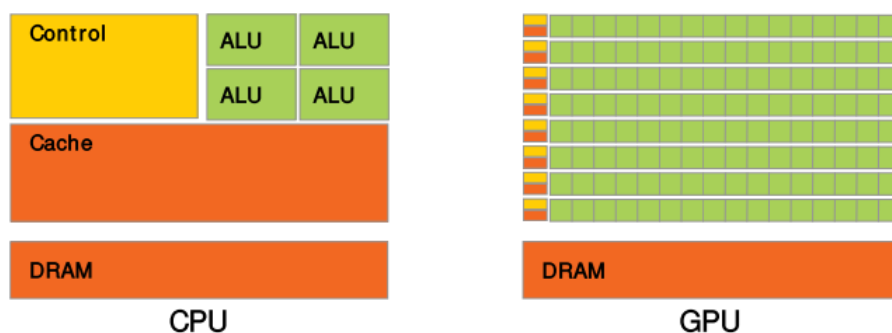
Obr. 13: Logo nástroja Google Web Toolkit

4 TECHNOLOGIA CUDA

Technológia CUDA je paralelná výpočtová platforma a programovací model vytvorený spoločnosťou Nvidia. Technológia pomáha zvýšiť výpočtový výkon využitím grafických výpočtových jednotiek GPU pre výpočet obecných algoritmov GPGPU. Využíva veľkú paralelnú výpočtovú silu Nvidia grafických kariet a umožňuje na GPU spúšťať kódy napísané v jazykoch C, C++ alebo FORTRAN. [21]

4.1 Architektúra

CUDA architektúra odpovedá modelu SIMT (Single-Instruction Multiple-Thread). Grafické karty s touto architektúrou nahradili vertexové a pixelové shadery univerzálnym procesorom, ktorý dovoľuje každej aritmetickej logickej jednotke ALU na čipe, aby bola zaradená programom na vykonávanie výpočtov. Nvidia predstavila radu grafických kariet s touto architektúrou pre použitie s GPGPU a bola postavená v súlade s požiadavkami IEEE pre výpočty s plávajúcou desatinou čiarkou a navrhnutá na mieru pre inštrukčnú sadu pre všeobecné výpočty, nie pre graficky špecifické. Taktiež bolo povolené ľubovoľné čítanie a zápis do pamäte a prístup k softvérovo spravovanej vyrovnávacej pamäti. Všetky tieto vlastnosti boli pridané s cieľom vytvoriť GPU, ktorá bude vynikajúca vo výpočtoch a tiež tradičných grafických úlohách. [22]

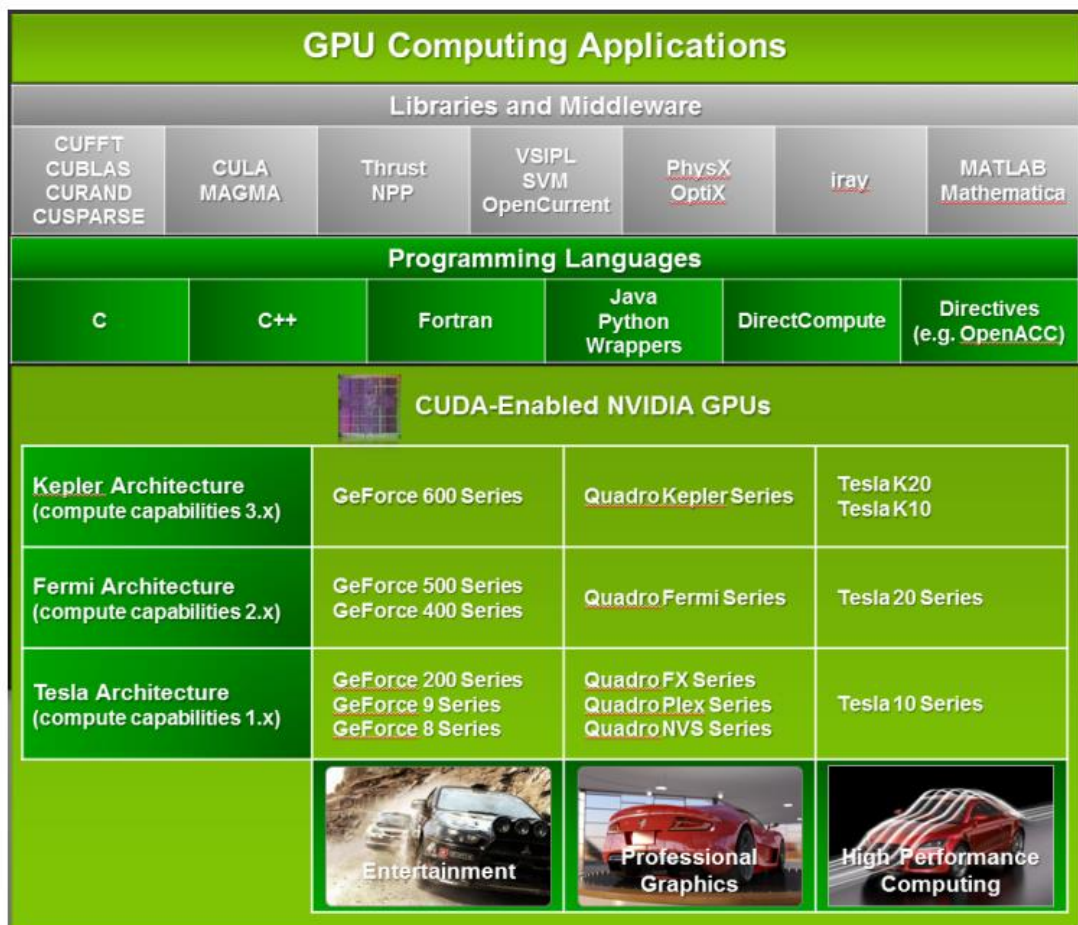


Obr. 14: CPU vs. GPU architektúra [23]

Na rozdiel od CPU, ktoré je optimalizované pre rýchle riadenie behu programu a rýchle prevedenie jednej inštrukcie, GPGPU je optimalizované pre viacnásobné jednoduché aritmetické operácie. Hardvérové a softvérové vlastnosti GPGPU sú veľmi úzko previazané.

4.1.1 Použitie architektúry

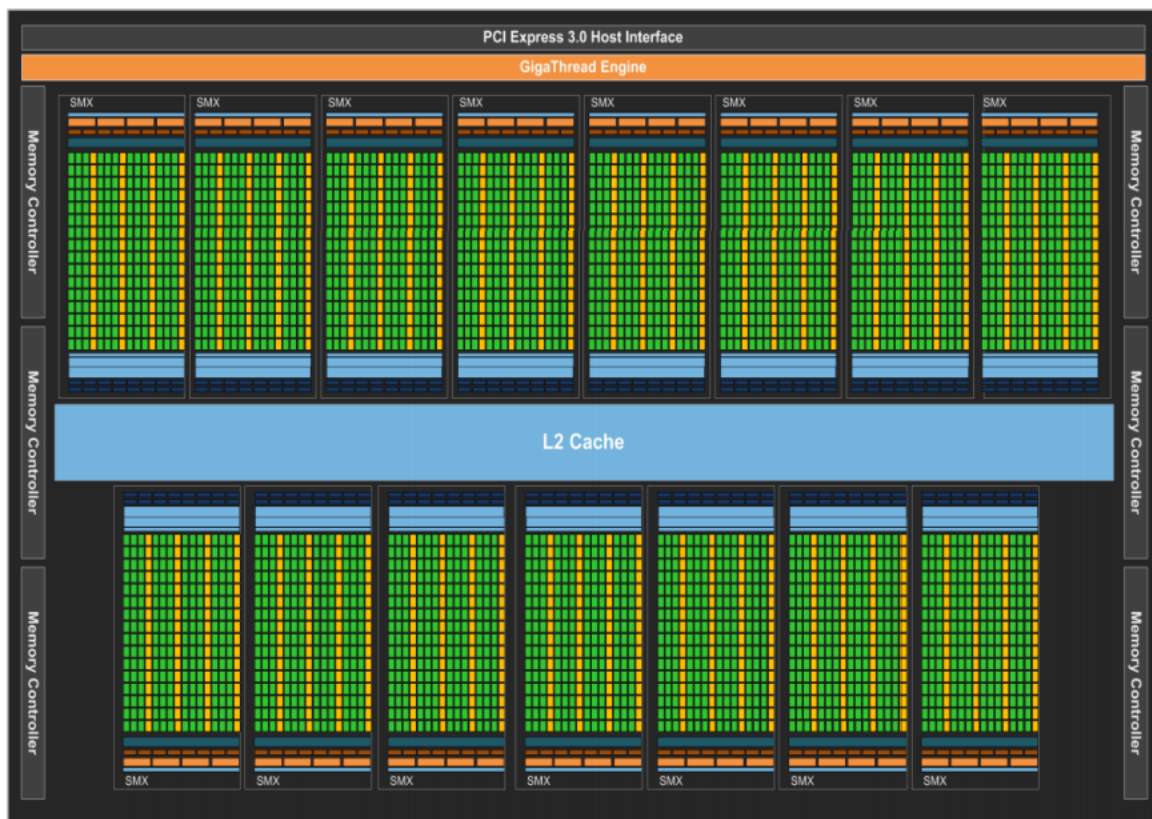
Po tom ako začala spoločnosť Nvidia produkovať karty pre všeobecné výpočty s architektúrou CUDA, neexistoval spôsob ako k novým pridaným vlastnostiam prístupit', okrem použitia OpenGL alebo DirectX technológií. To by pre užívateľov znamenalo, že by výpočty museli riešiť ako grafické problémy a naďalej pokračovať v písaní v graficko-orientovanom shading jazyku (využívanie shaderov) ako OpenGL GLSL alebo Microsoft HLSL. Spoločnosť Nvidia tak zobrala štandardný jazyk C a pridala menšie množstvo kľúčových slov pre vykonávanie špeciálnych vlastností CUDA architektúry a vytvorila tak prvý jazyk navrhnutý pre všeobecné výpočty GPGPU na grafických výpočtových jednotkách. Predstavila taktiež kompilátor CUDA C vytvorený pre tento jazyk. V nadväznosti na tento jazyk Nvidia poskytla špecializovaný hardvérový ovládač pre využitie výpočtovej sily CUDA architektúry. Pre zjednodušenie vývoja Nvidia vytvorila unifikovanú a výpočtovú architektúru Tesla, za ňou nasledujú architektúry Fermi a Kepler. Obr. 15 popisuje aplikáciu GPGPU Nvidia. [22]



Obr. 15: Aplikácia GPGPU Nvidia [23]

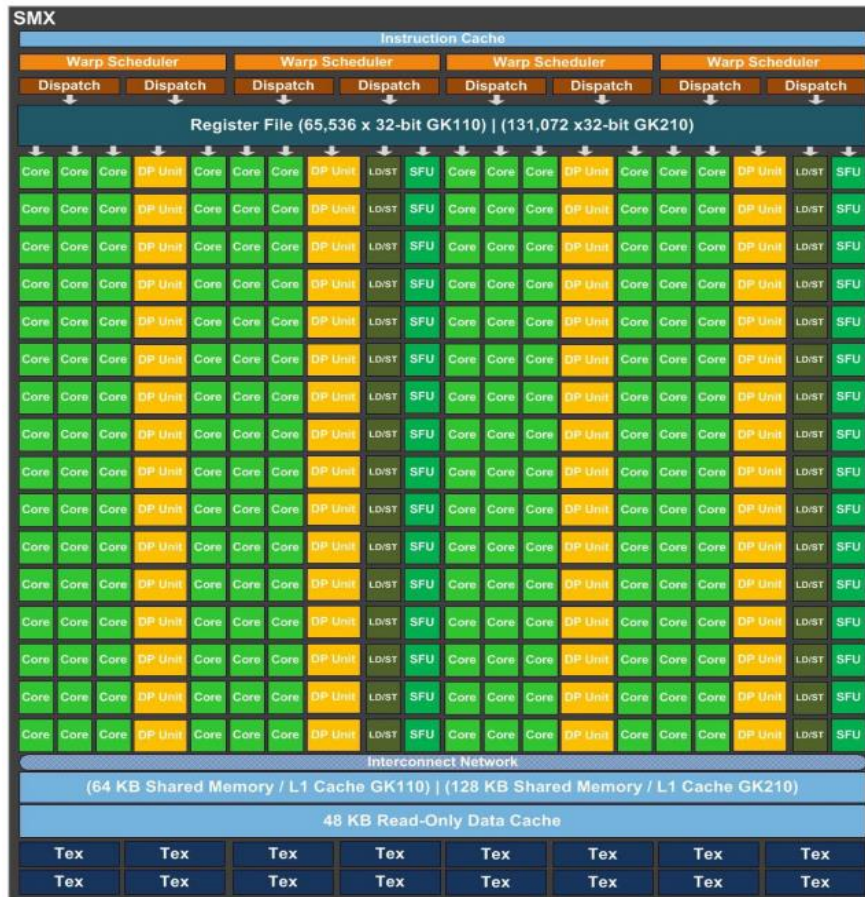
4.1.2 Vlastnosti architektúry Kepler

Architektúra Kepler je prvá architektúra spoločnosti Nvidia zameraná na úsporu energie. Všetky grafické karty architektúry Kepler sú vyrobené 28 nm technológiou, teda šetrí energiu a generujú menej tepelného výstupu. Cieľom tejto architektúry je poskytovať vysoko výkonné paralelné výpočtové mikroprocesory. Zahŕňa rýchle výpočty s plávajúcou desatinnou čiarkou s dvojitou presnosťou pre vysoko výkonné výpočtové zariadenie. Architektúra zahŕňa 15 streamovacích procesorov novej generácie SMX a šesť 64-bitových pamäťových radičov. [24]



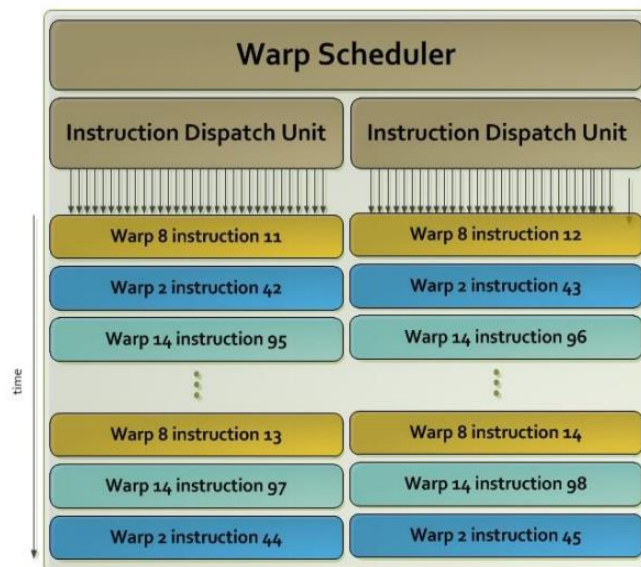
Obr. 16: CUDA Kepler architektúra [24]

Každý streamovací procesor obsahuje 192 CUDA jadier a 4 plánovače warpov. CUDA jadro obsahuje samostatnú jednotku ALU a jednotku FPU s podporou zret'azeného spracovania (Obr. 17).



Obr. 17: Streamovací procesor SMX [24]

Každý plánovač warpov může v jednom okamžiku vykonávat dvě instrukce z warpu (Obr. 18).



Obr. 18: Plánovač warpov architektury Kepler [24]

Kepler architektúra podporuje CUDA Compute Capability (výpočtové možnosti) verzie 3.5.

4.1.3 Programovací model

Základom programovacieho modelu je vlákno. Vlákna sú združované do blokov, ktoré môžu byť 1-rozmerné, 2-rozmerné alebo 3-rozmerné. Počet vlákien v bloku sa líši od architektúry a výpočtových možností. Každý blok sa vykonáva na jednom streamovacom procesore. Plánovacou jednotkou streamovacieho procesoru je warp. Bloky vlákien sa delia do niekoľkých warpov. Vlákna warpu vykonávajú paralelne jednu inštrukciu. Programový kód na hostiteľskom systéme je zodpovedný za inicializáciu blokov, alokáciu pamäti pre CPU a GPU a volanie kernelov. [23]

4.1.4 Pamäťový model

Pamäťový model GPGPU obsahuje niekoľko typov pamätí. Pamäte majú rôznu veľkosť, rýchlosť prístupu a taktiež viditeľnosť. Adresný priestor architektúry Kepler je unifikovaný, čo znamená, že programátor má prístup k operačnej pamäti systému a grafickej pamäti pomocou jedného adresného priestoru. O presuny medzi pamäťami sa starajú ovládače a prekladače programového kódu.

Delenie pamätí GPGPU:

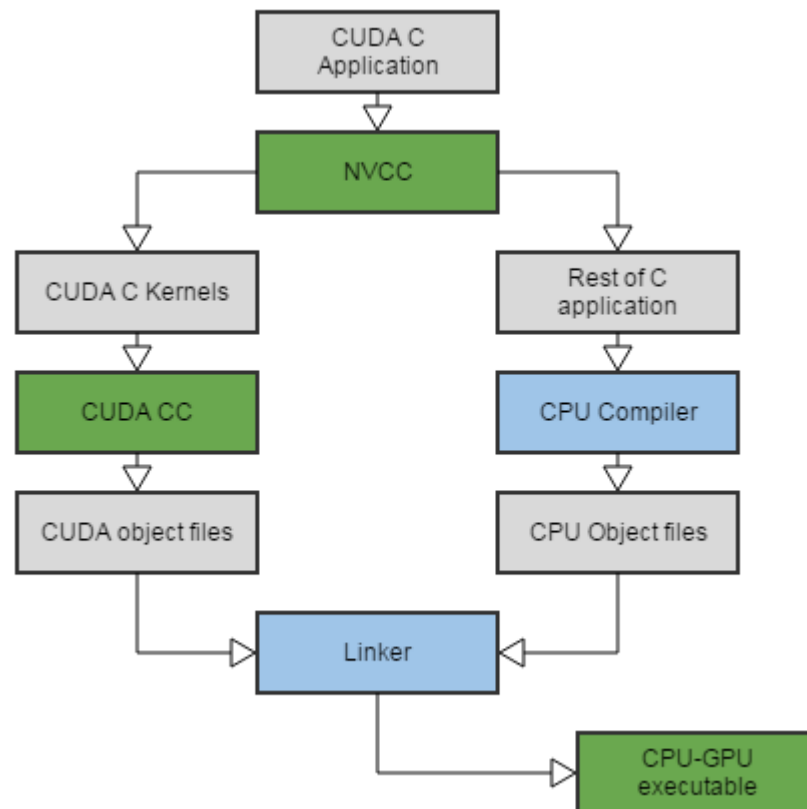
- globálna pamäť,
- zdieľaná pamäť,
- lokálna pamäť.

Globálna pamäť má veľkú kapacitu a je dostupná pre všetky vlákna a aj pre hosťujúci systém, je však pomalá. Zdieľaná pamäť je dostupná pre vlákna v bloku, má malú kapacitu a rýchly prístup. Lokálna pamäť je pamäť vlákna, je dostupná len pre toto vlákno, má malú kapacitu a veľmi rýchly prístup. [23]

4.2 Preklad aplikácie pre GPU

Kód pre GPU môže byť písaný pomocou inštrukčnej sady PTX alebo pomocou jazyka C. Pre spustenie CUDA kódu je potrebné mať kompilátor pre GPU kód. NVCC je kompilátor pre zdrojový kód a premieňa ho na funkčnú CUDA aplikáciu. Preklad kódu je buď do PTX alebo do binárnej formy. Zaisťuje taktiež volanie štandardných prekladových nástrojov ako GCC kompilátor. NVCC teda rozdeľuje kód na hostiteľský a pre GPGPU a ten zašle na

vykonanie buď na Nvidia kompilátor alebo na štandardný kompilátor. Binárny kód (cubin) je generovaný pre konkrétnu výpočtovú možnosť a špecifikácia je vykonávaná pomocou prepínača `-code`. PTX inštrukcie sú tiež generované pre konkrétnu výpočtovú možnosť a využívajú prepínača `-arch`. [23]



Obr. 19: Kompilácia CUDA kódu

4.3 CUDA C

CUDA C predstavuje rozšírenie jazyku štandardného ANSI C pre heterogénne programovanie a tiež rozhranie pre správu zariadenia. Tiež poskytuje škálovateľný programovací model, ktorý umožňuje programom vykonávať paralelné úlohy na GPU s rôznym počtom jadier, bez potreby programátorov učiť sa novému jazyku, ak sú uvedomelí v programovacom jazyku C. CUDA C implementuje do jazyka C nové modifikátory funkcií:

- `__global__`,
- `__device__`,

- `__host__`.

Modifikátor `__global__` predstavuje kernel, teda GPU kód, ktorý je volaný z CPU. Modifikátor `__device__` je kód, ktorý je volaný v rámci GPU funkcií. Posledný modifikátor `__host__` je explicitný modifikátor. Netreba ho špecifikovať, predstavuje kód volaný z CPU. Ak je potreba, modifikátory `__host__` a `__device__` sa môžu u jednej funkcie kombinovať. Vygenerujú sa jej obe varianty.

CUDA C dovoľuje programátorovi definovať C funkcie, ktoré sa budú vykonávať N krát na N CUDA jadrách. Tieto funkcie sú nazývané kernely. Kernel je definovaný modifikátorom `__global__` a má automaticky definované premenné. Kernely majú prístup ku globálnej pamäti GPU. Nemôžu byť statické a taktiež rekurzívne. Návrátovým typom je `void`.

```
__global__ void add( int *a, int *b, int *c ) {
    int tid = blockIdx.x;    // handle the data at this index
    if (tid < N)
        c[tid] = a[tid] + b[tid];
}
```

Obr. 20: CUDA C kernel [22]

Volanie kernelu je uskutočnené takto: `add<<<N,1>>>(dev _ a, dev _ b, dev _ c);`.

Automaticky definované premenné pri funkciách `__global__` a `__device__` majú prístup k definovaným premenným:

- `dim3 gridDim`,
- `dim3 blockDim`,
- `dim3 blockIdx`,
- `dim3 threadIdx`.

`dim3` predstavuje špeciálny dátový typ, ktorý obsahuje tri zložky, súradnice x , y a z . Z nich je možné vypočítať konkrétny index vlákna. Premenná `gridDim` vyjadruje dimenziu mriežky v blokoch. `BlockDim` predstavuje dimenziu bloku vo vláknach, `blockIdx` pozíciu bloku v mriežke, `threadIdx` pozíciu vlákna v bloku.

K rozšíreniu jazyka C patria štruktúry, ktoré obsahujú zložky `x[, y[, z[, w]]]` a aj špeciálne dátové typy a to:

- *[u]char[1..4]*,
- *[u]short[1..4]*,
- *[u]int[1..4]*,
- *[u]long[1..4]*,
- *float[1..4]*.

Medzi rozšírenie jazyka C patrí aj správa pamäti. CUDA C obsahuje tieto základné funkcie pre prácu s globálnou pamäťou:

- *cudaMalloc(void **pointer, size_t nbytes)*,
- *cudaMemset(void *pointer, int value, size_t count)*,
- *cudaFree(void *pointer)*,
- *cudaMemcpy(void *dst, void *src, size_t nbytes, enum cudaMemcpyKind direction)*.

Pre použitie kopírovania z alebo do globálnej pamäte je definovaný výčtový typ, ktorý určuje smer kopírovania:

- *cudaMemcpyHostToDevice*,
- *cudaMemcpyDeviceToHost*,
- *cudaMemcpyDeviceToDevice*.

Taktiež sú definované nové modifikátory premenných pre umiestnenie premenných v pamäti a to:

- *__device__*,
- *__shared__*,
- *__constant__*.

Modifikátor *__device__* určuje, že premenná je uložená v hlavnej globálnej pamäti GPGPU. Priestor je alokovaný pomocou funkcie *cudaMalloc*. *__shared__* určuje umiestnenie zdieľanej pamäti streamovacieho multiprocesoru, *__constant__* v konštantnej pamäti. [22] [23]

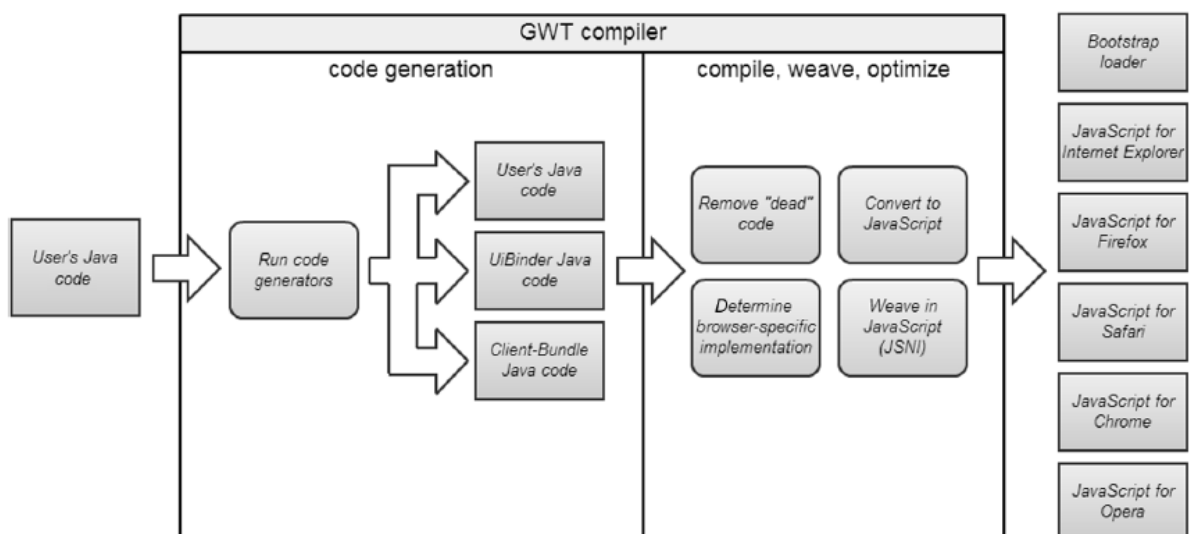
II. PRAKTICKÁ ČASŤ

5 POUŽITÉ TECHNOLOGIE

5.1 Webový framework

Moderná doba sa uberá v duchu webových aplikácií, t.j. takých, ktoré bežia na akomkoľvek počítači, ktorý disponuje webovým prehliadačom. A presne také riešenie ponúka Google Web Toolkit. GWT uskutočňuje konverziu kódu v jazyku Java do JavaScriptu, ktorý je potom možné spustiť vo webovom prehliadači. Získaný kód v jazyku JavaScript je do istej miery veľmi dobre optimalizovaný pre rôzne webové prehliadače. Ako bolo písane v teoretickej časti GWT patrí do sady nástrojov, ktorý vyplňa medzeru medzi jazykom Java a tvorbou užívateľského rozhrania v rámci webovej aplikácie.

Pre chod GWT je potrebné Java SDK a GWT SDK. Taktiež vývojové prostredie a ľubovoľný zásuvný modul pre webový prehliadač. Vo webovej aplikácii CUDA On-line IDE je použité GWT SDK vo verzii 2.7.0 a Java SDK vo verzii 1.7.0_71. V prípade vývojového prostredia Eclipse je nutná inštalácia zásuvného modulu Google plugin for Eclipse. Z hľadiska kompilácie kódu, GWT zoberie kód napísaný v Jave a preloží ho do JavaScriptu. Počas kompilácie kódu dochádza k analýze kódu, optimalizácii a odstránení mŕtvych častí kódu. Vo výsledku je kód v jazyku JavaScript, ktorý je najmenší možný. Kompilátor môže odštartovať jeden alebo viacej rôznych generátorov kódu podľa toho, na čo v kóde narazí. To môže byť napríklad tzv. *UIBinder*, ktorý sa spúšťa v prípade rozhrania v XML súbore *ClientBundle* starajúci sa o rýchle načítanie externých CSS súborov alebo obrázkov a i. [19, s. 7-8]



Obr. 21: Priebeh kompilácie kódu Java do JavaScriptu [19]

5.1.1 Klientske rozhranie

Pre tvorbu klientskeho kódu je použitý jazyk Java, ktorý je prevedený do jazyku JavaScript. Je preto nutné, aby pre beh aplikácie bol v prehliadači povolený JavaScript. Pre tvorbu klientskeho pohľadu aplikácie je potrebné zostaviť *EntryPoint* triedu. Táto trieda obsahuje metódu *onModuleLoad()*, ktorá je vykonaná pri spustení aplikácie. Táto metóda môže obsluhovať tvorbu nového užívateľského rozhrania alebo nastaviť handlers (obsluhu) pre udalosti. [19, s. 36-37]

```
package com.google.gwt.sample.hello.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.Widget;

/**
 * Hello World application.
 */
public class Hello implements EntryPoint {

    public void onModuleLoad() {
        Button b = new Button("Click me", new ClickHandler() {
            public void onClick(ClickEvent event) {
                Window.alert("Hello, AJAX");
            }
        });

        RootPanel.get().add(b);
    }
}
```

Obr. 22: Tvorba klientskeho rozhrania pomocou triedy EntryPoint

5.1.2 GWT RPC

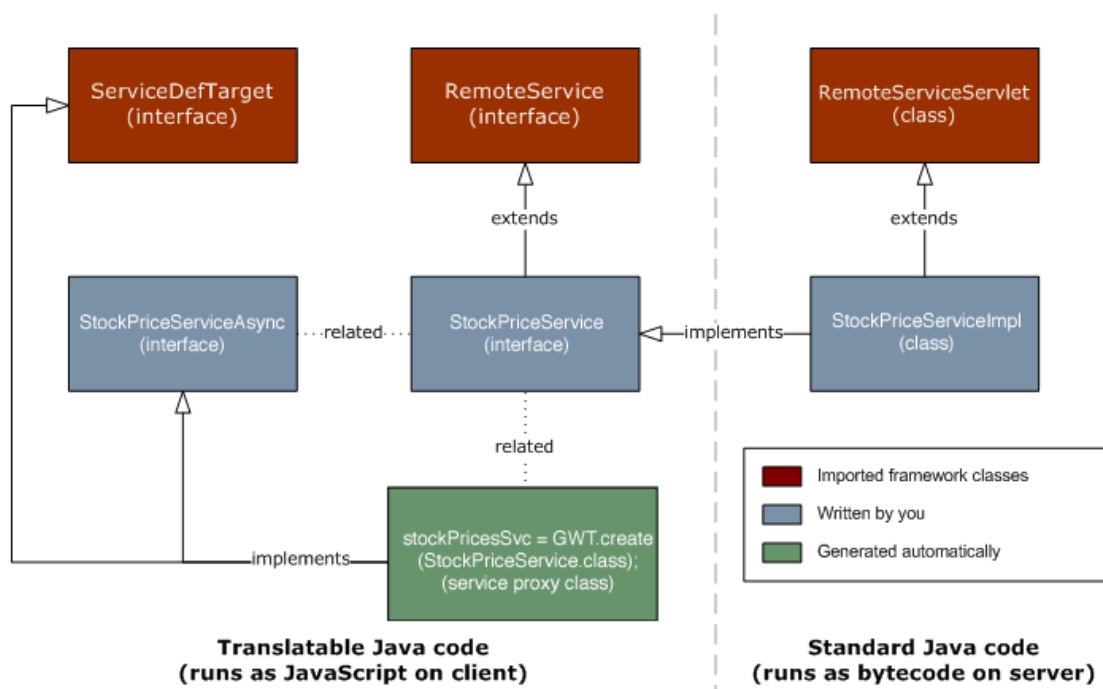
RPC (Remote procedure call) je asynchrónne volanie z HTML stránky na server. Je to neblokujúca operácia, čiže stránka nie je blokována počas vykonávania požiadavky. Požiadavka je vykonávaná na serveri ihneď. Návrátové metódy špecifikujú, čo sa vykoná, ako náhle je uskutočnené návratové volanie. RPC asynchrónne volanie je uskutočňované na základe AJAX technológie. Výhodou tejto techniky je, že stránka nezamŕza pri uskutočňo-

vaní požiadavky. Je možné vykonávať inú prácu popri čakaní na návratové volanie. Tiež je možné vykonávať paralelné operácie, teda posielat' viac serverových volaní v rovnakom čase, čo je ale obmedzované webovými prehliadačmi, ktoré dovoľujú v jednom okamžiku dve odchádzajúce sieťové spojenia.

GWT RPC framework poskytuje jednoduché rozhranie pre komunikáciu klienta a servera pre zasielanie Java objektov cez HTTP. Kód na strane servera je vyvolávaný ako služba, kde pre implementáciu takejto služby je využitá Java servlet architektúra. Na strane klienta je používaná automaticky generovaná proxy trieda pre vykonávanie volania na službu. Serializované Java objekty sú vymieňané pomocou argumentov v metódach volania a ich návratovou hodnotou.

Pre zostavenie GWT RPC rozhrania sú potrebné nasledujúce tri elementy:

- služba bežiaca na serveri,
- klientsky kód, ktorý vykonáva volanie služby,
- serializovateľné Java objekty, ktoré sú vymieňané medzi klientom a serverom. [25]



Obr. 23: Implementácia GWT RPC [25]

Na základe obrázka je viditeľné, že služba bežiaca na serveri musí dediť triedu *RemoteServiceServlet* a implementovať rozhranie služby. Služba neimplementuje asynchrónnu verziu služby. Táto dedená trieda obsluhuje serializáciu dát, ktoré sú posielané medzi klientom a serverom a vyvoláva žiadané metódy služby na strane servera.

5.1.3 JSNI

JSNI (JavaScript Native Interface) je technika pre implementovanie Java metód na strane klienta priamo pomocou javascriptového kódu. JSNI metódy sú deklarované pomocou kľúčového slova *native*, kde javascriptový kód je písaný v špeciálne formátovaných komentárových blokoch. Pomocou JSNI je možné vyvolávať Java metódy a polia priamo z JavaScriptu. Pri práci s JSNI je dôležité vedieť, že ak chceme pristupovať k *window* alebo *document* objektu, v JSNI metódach je potrebné odkazovať *\$wnd* pre *window* alebo *\$doc* pre *document*, pretože kompilovaný javascriptový kód beží vo vnorenom rámci a tieto objekty sú inicializované k *window* a *document* objektu. [26]

```
public static native void alert(String msg) /*- {  
    $wnd.alert(msg);  
}-*/;
```

Obr. 24: Ukážka JSNI metódy

5.1.4 Sencha GXT

Sencha GXT (Ext GWT) je nadstavba pre tvorbu klientskeho rozhrania bohatej webovej aplikácie od firmy Sencha. GXT poskytuje okrem platenej verzie pre komerčné využitie aj verziu GNU/GPL, ktorá môže byť použitá v GNU/GPL projektoch.

Sencha GXT používa GWT kompilátor pre kompiláciu Java kódu do vysoko optimalizovaného HTML5 a javascriptového kódu bežiaceho na všetkých moderných prehliadačoch. Táto nadstavba poskytuje výkonné widgety, šablóny, rozloženia stránky, grafy, prácu s dátami pomocou dátového úložiska a i.

Všetky prvky GXT sú upravovateľné s možnosťou definovania tém. GXT zahrňuje dátové stromy, dátové tabuľky, listy, formuláre, panely pre menu alebo toolbar, okná, nahrávanie súborov a i. Pre túto knižnicu je taktiež dostupných veľa užívateľských rozšírení spracovaných GXT komunitou. [27]

5.2 Vývojové prostredie

Eclipse je voľné a open-source integrované vývojové prostredie (IDE) vydané pod licenciou Eclipse Public License. Je riadené a spravované Eclipse.org konzorciom. Eclipse IDE je vytvorené prevažne v jazyku Java a je multiplatformové. Dovoľuje vývojárom vytvárať a upravovať toto IDE pomocou dostupných zásuvných modulov vytvorených Eclipse komunitou alebo pomocou vlastných. Eclipse poskytuje rozšíriteľné nástroje a frameworky, ktoré postihujú rôzne štádium vývoja softvéru, vrátane podpory pre modelovanie, programové prostredie pre Javu, C, C++, PHP, Python a ďalšie, nástroje pre testovanie, výkonnosť, podnikovú logiku a tiež tvorbu bohatých webových aplikácií alebo vývoj pre vstavané zariadenia. Pre tvorbu CUDA On-line IDE je použité vývojové prostredie Eclipse Java IDE for Web Developers vo verzii Kepler Service Release 2.

5.3 Webový server

Apache Tomcat server je webový aplikačný kontajner, vytvorený pre beh Java servletov a JSP webových aplikácií. Je to open-source multiplatformový server vytvorený Apache Software Foundation. Vydaný je pod licenciou Apache License 2.0. Tomcat je veľmi stabilný a má všetky vlastnosti ako komerčné webové aplikačné kontajnery. Implementuje väčšinu Java EE (Enterprise Edition) špecifikácií, ako boli už zmienené Java servlety, JSP, ale aj Java EL (Expression Language) a WebSocket. Poskytuje taktiež rôzne pridané funkcie, napr. správcu Tomcat aplikácií, ktorý je inštalovaný v kontextovej adrese */manager* a poskytuje funkcionality pre správu web aplikácií bežiacich na serveri. Tomcat akceptuje všetky požiadavky na porte 8080. To odlišuje Java servlety od štandardných HTTP servletov, ktoré prijímajú požiadavky na porte 80. Toto nastavenie je ale možné zmeniť. [28, s. 1-15]

V tejto práci je použitý Apache Tomcat server vo verzii 7.0. V tabuľke sú vidieť špecifikácie pre jednotlivé verzie Apache Tomcat serveru.

Tomcat verzia	Servlet API	JSP API	WebSocket API	Java verzia
8.0.x	3.1	2.3	1.1	7 a novšia
7.0.x	3.0	2.2	1.1	6 a novšia
6.0.x	2.5	2.1	N/A	5 a novšia
5.5.x	2.4	2.0	N/A	1.4 a novšia
4.1.x	2.3	1.2	N/A	1.3 a novšia
3.3.x	2.2	1.1	N/A	1.1 a novšia

Tab. 2: Tabuľka špecifikácií jednotlivých Tomcat server verzií [28]

5.4 Použité knižnice a nástroje tretích strán

5.4.1 Apache Commons

Apache Commons predstavujú knižnice, ktoré sú vytvárané Apache komunitou, kde je zabezpečené, že majú minimálnu závislosť na iných knižniciach. Tieto komponenty udržiavajú stabilné rozhrania, preto je výhodné ich použitie do budúcnosti. Knižnice sú vydávané pod Apache License 2.0. V tomto projekte je použitá knižnica FileUpload, ktorá poskytuje rozhranie pre nahrávanie súborov pre servlety a webové aplikácie, tiež knižnica IO pre obsluhu vstupno-výstupných funkcií. [29]

5.4.2 Apache HttpComponents

Apache HttpComponents je projekt pre vytváranie a obsluhu nástrojov zameraných na protokol HTTP a k nemu súvisiacich. HttpComponents slúžia na stavbu klientskych a serverových aplikácií ako webové prehliadače, webové služby pre prenos dát alebo rozširujú HTTP protokol pre distribuovanú komunikáciu. Projekt je vydávaný pod licenciou Apache License 2.0. [30]

5.4.3 Google Gson

Google Gson je knižnica pre konverziu Java objektov do ich JSON reprezentácií. Taktiež môže byť použitá pre konverziu JSON reťazcov do Java objektov. Gson je vydávaná pod licenciou Apache License 2.0.

5.4.4 SLF4J

SLF4J je jednoduchá fasáda pre zaznamenávanie informácií o behu aplikácie, chybách alebo výnimkách, ktorá implementuje rôzne logovacie frameworky ako *java.util.logging*, *log4j* a iné. Táto knižnica je vydávaná pod MIT licenciou. [31]

5.4.5 Zip4j

Zip4j je open-source knižnica pre obsluhu ZIP súborov. Hlavnými vlastnosťami tejto knižnice sú vytváranie, rozbaľovanie, rozširovanie ZIP súborov, práca so zabezpečenými ZIP archívami, vrátane podpory AES, štandardného ZIP šifrovania, rozdeľovania balíčkov na viac častí a zaznamenávanie priebehu. Knižnica spadá pod licenciou Apache License 2.0. [32]

5.4.6 Zt-zip

Zt-zip je nástroj, ktorý poskytuje jednoduchú a bezpečnú prácu so ZIP súborami ako je rozbaľovanie a balenie ZIP archívov, porovnávanie, kompresiu alebo pridávanie súborov do už zabelených archívov. Projekt je postavený na *java.util.zip.** balíčkoch pre prístup pomocou prúdov. Taktiež je podporovaná väčšina metód pre prácu so súborovým systémom. Nástroj je vydávaný pod Apache License 2.0.

5.4.7 JavaMail

JavaMail API predstavuje platformovo a protokolovo nezávislý nástroj pre tvorbu mailových aplikácií. Poskytuje možnosť overovania užívateľov alebo posielania a prijímania mailov pomocou SMTP, POP3 alebo IMAP protokolu. Je vydávaný pod CDDL a GNU/GPL licenciou.

5.4.8 ACE

ACE (Ajax.org Code Editor) je vysoko výkonný editor kódu napísaný v JavaScripte a môže byť ľahko integrovateľný do každej webovej aplikácie. ACE GWT predstavuje integráciu tohto editora do GWT. Tento nástroj poskytuje zvýrazňovač syntaxe pre viac

ako 100 programovacích jazykov, možnosť zmeny rôznych tém, automatické dopĺňanie kódu, vyhľadávanie a nahrádzanie v kóde. Tiež je možné vykonávať funkcionality editoru pomocou kombinácie tlačidiel (napr. Ctrl a + pre zväčšenie písma) a i. Použitie tohto editoru spadá pod MIT licenciu. [33]

5.4.9 Exuberant Ctags

Exuberant Ctags vykonáva generovanie súborov s indexmi objektov dostupných v súboroch so zdrojovými kódmi. Podporuje indexovanie viac ako 40 jazykov, pod ktoré patrí aj C a C++ syntax. Vygenerovaný súbor obsahuje značky špecifické pre programovací jazyk a poskytuje tak informácie o použitých triedach, parametroch, ich umiestnení, prístupnosti a ďalších vlastnostiach. Nástroj je vydávaný pod licenciou GNU/GPL verzie 3. [34]

5.4.10 CMake

CMake je open-source multiplatformový nástroj, ktorý spracováva proces prekladu aplikácií. Poskytuje jednoduchú tvorbu konfiguračného súboru pre preklad CMakeLists.txt. Na základe neho CMake môže generovať natívne aplikácie a vytvárať statické a dynamicke knižnice pre rôzne programovacie jazyky vrátane CUDA C. Nástroj je používaný v spojení s natívnymi prekladovými nástrojmi ako je Make. CMake je distribuovaný pod licenciou BSD. [35]

6 ANALÝZA A NÁVRH RIEŠENIA

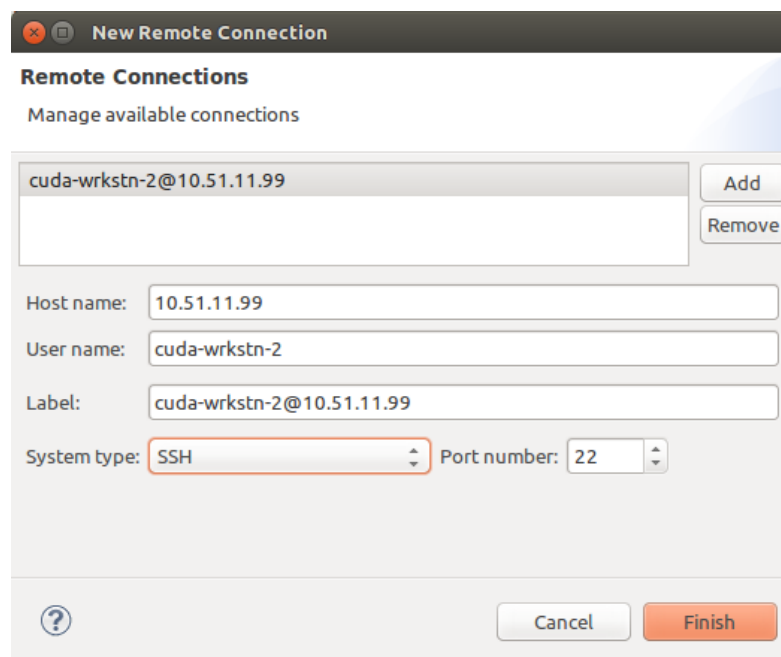
6.1 Súčasný stav

Súčasný stav riešenia programovania CUDA aplikácií v predmete Paralelné výpočty na grafických kartách v skratke PVG obsahuje obmedzenia, kvôli ktorým sa uskutočňuje tento projekt.

Keďže jednotlivé pracovné stanice používané študentmi neobsahujú grafické karty s podporou CUDA technológie, preklad a spúšťanie aplikácií musí prebiehať na vzdialenom serveri, ktorý takúto grafickú kartu obsahuje.

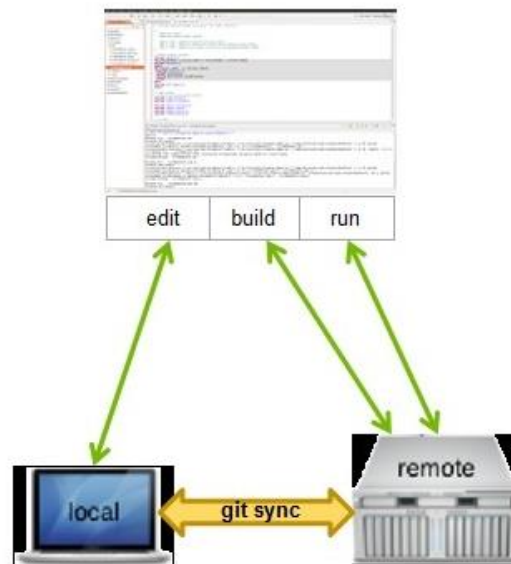
Prístup k tomuto serveru je teda uskutočnený cez programovacie prostredie Nvidia Nsight Eclipse Edition, navrhnuté pre tvorbu CUDA aplikácií. Toto prostredie je možné spúšťať z operačných systémov Linux a Mac OS X. Keďže väčšina pracovných staníc na Fakulte aplikovanej informatiky beží na operačnom systéme Microsoft Windows 7, je potrebné spúšťať virtuálny operačný systém Ubuntu cez virtualizačný nástroj VMware. Následne je v ňom možné prísť k tvorbe CUDA projektu cez toto prostredie.

Keďže grafická karta sa nachádza vo vzdialenom serveri, preklad a spustenie projektu musí byť vykonaný vo vzdialenom systéme. Nsight prostredie toto dovoľuje. Pri tvorbe projektu musí byť nastavené spojenie so vzdialeným serverom, na ktorom prebehne preklad, na ňom cesta k zdrojovým kódom projektu a cesta ku CUDA sade nástrojov.



Obr. 25: Nastavenie SSH spojenia so vzdialeným prekladovým serverom

Nsight sa následne pripojí na vzdialený server a synchronizuje kópie súborov projektu medzi lokálnym a vzdialeným systémom. Pre preklad projektu je vyžadovaný nastavený Git, tak ako na lokálnom systéme, tak aj na vzdialenom. Po vykonaní prekladu sú výsledky prekladu kopírované späť na lokálny systém. Na obrázku je znázornený princíp prekladu na vzdialenom systéme.



Obr. 26: Princíp prekladu na vzdialenom systéme [36]

6.1.1 Výhody a nevýhody súčasného riešenia

Medzi výhody tohto riešenia tvorby CUDA aplikácií určite patrí programovacie prostredie Nvidia Nsight Eclipse Edition. Tento nástroj značne urýchľuje vývoj softvéru, podporuje výkonné nástroje pre ladenie a profilovanie projektu s cieľom jeho optimalizovania, tak ako na CPU, tak aj GPU a preberá všetky užitočné vlastnosti programovacieho prostredia Eclipse. Nevýhodou tohto riešenia je komplikovanejšie nastavovanie spojenia so vzdialeným serverom, nastavenie nástroju Git a používanie virtuálneho systému Ubuntu.

6.2 Inovácia riešenia

Pod inováciou riešenia je predstavovaný návrh programovacieho prostredia vytvoreného pomocou webových technológií, teda web aplikácia, ktorá bude bežať na vzdialenom prekladovom serveri a umožní tak študentom vytvárať CUDA aplikácie priamo na nej, bez potreby zložitého nastavovania lokálneho projektu pri preklade na vzdialenom serveri.

Cieľom tejto diplomovej práce je poskytnúť on-line webové programovacie prostredie pre tvorbu a testovanie CUDA aplikácií, ktoré nahradí súčasný stav riešenia. Aplikácia bude umožňovať správu pracovného prostredia projektov tzv. workspace, správu a nastavenie projektov, ich následný preklad a spustenie priamo na vzdialenom serveri.

Užívatelia budú pristupovať k aplikácií v rámci jedného postavenia študent, kde je požadované bezpečné prihlásenie pomocou ich údajov do univerzitnej siete a každý užívateľ bude mať oddelený pracovný priestor na prekladovom serveri.

Prístup k aplikácií bude uskutočňovaný pomocou webových prehliadačov v pracovných staniciach napojených na univerzitnú sieť na určenej webovej adrese web aplikácie.

6.3 Definícia požiadaviek

Požiadavky na softvér určujú funkcionality systému. Delia sa na funkčné a nefunkčné. Funkčné požiadavky poskytujú funkcie a služby systému, nefunkčné tie, ktoré nesúvisia s funkciami z pohľadu užívateľa. Všetky tieto požiadavky boli najprv zozbierané, potom analyzované a špecifikovala sa ich priorita.

6.3.1 Funkčné požiadavky

Požiadavky definujúce funkcie a služby systému:

- užívateľské prihlásenie,
- správa pracovného priestoru,
- správa projektu,
- vytváranie a správa súborov,
- preklad projektu,
- spustenie projektu,
- vytvorenie a editovanie konfiguračného súboru pre preklad,
- zvýrazňovanie syntaxe kódu,
- sťahovanie súčastí (pracovné prostredie, projekt, súbor),
- nahrávanie súčastí (pracovné prostredie, projekt, súbor),
- tvorba štruktúry kódu,
- správa editoru kódu.

6.3.2 Nefunkčné požiadavky

Požiadavky definujúce vlastnosti systému:

- webová aplikácia,
- komunikácia prostredníctvom protokolu HTTP,
- beh na vzdialenom Linuxovom serveri,
- preklad a spustenie projektu na grafickej karte s CUDA technológiou,
- využitie webového serveru pre integráciu aplikácie,
- overovanie užívateľov na základe prihlasovacích údajov do univerzitickej siete,
- oddelenie pracovného priestoru pre každého užívateľa,
- definovanie štruktúry pracovného prostredia,
- nezávislé riešenie pre moderné prehliadače,
- užívateľsky známe a prívetivé programovacie prostredie,
- využitie voľne dostupného softvéru.

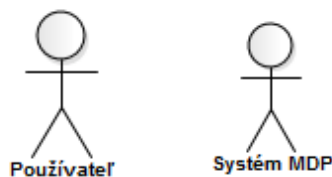
6.4 Model prípadov užitia

Model prípadov užitia predstavuje formu zachytenia požiadaviek, predstavuje hľadanie aktérov a hľadanie prípadov použitia.

6.4.1 Aktéri systému

Pod aktérmi systému je možno chápať všetko čo komunikuje so systémom. Aktéri spadajú mimo modelovaný systém.

V aplikácii CUDA On-line IDE sú dvaja aktéri a to Používateľ, ktorý obsluhuje programovacie prostredie, teda správu pracovného prostredia, správu projektov, vytváranie CUDA aplikácií a Systém MDP, pre vykonávanie prekladania a spúšťania aplikácií. Tento systém bude popísaný v kapitole 7.2.3.1 Správa pracovného prostredia, Systém MDP.



Obr. 27: Aktéri webovej aplikácie

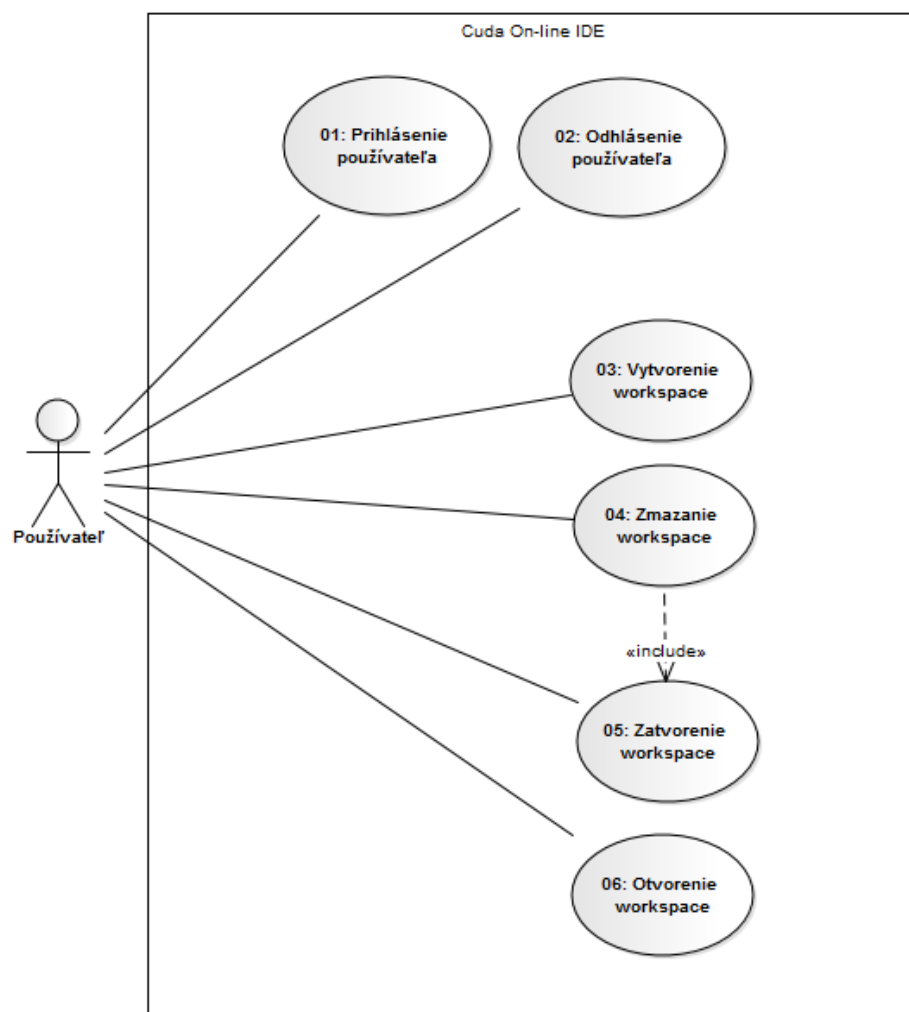
6.4.2 Diagram prípadov užitia

Diagram prípadov užitia popisuje funkcionality systému, teda to, čo aktér požaduje. Pre lepšie znázornenie funkcionality systému CUDA On-line IDE je diagram prípadov užitia rozdelený na štyri časti, ktoré však tvoria jeden celok.

Tieto štyri časti sú:

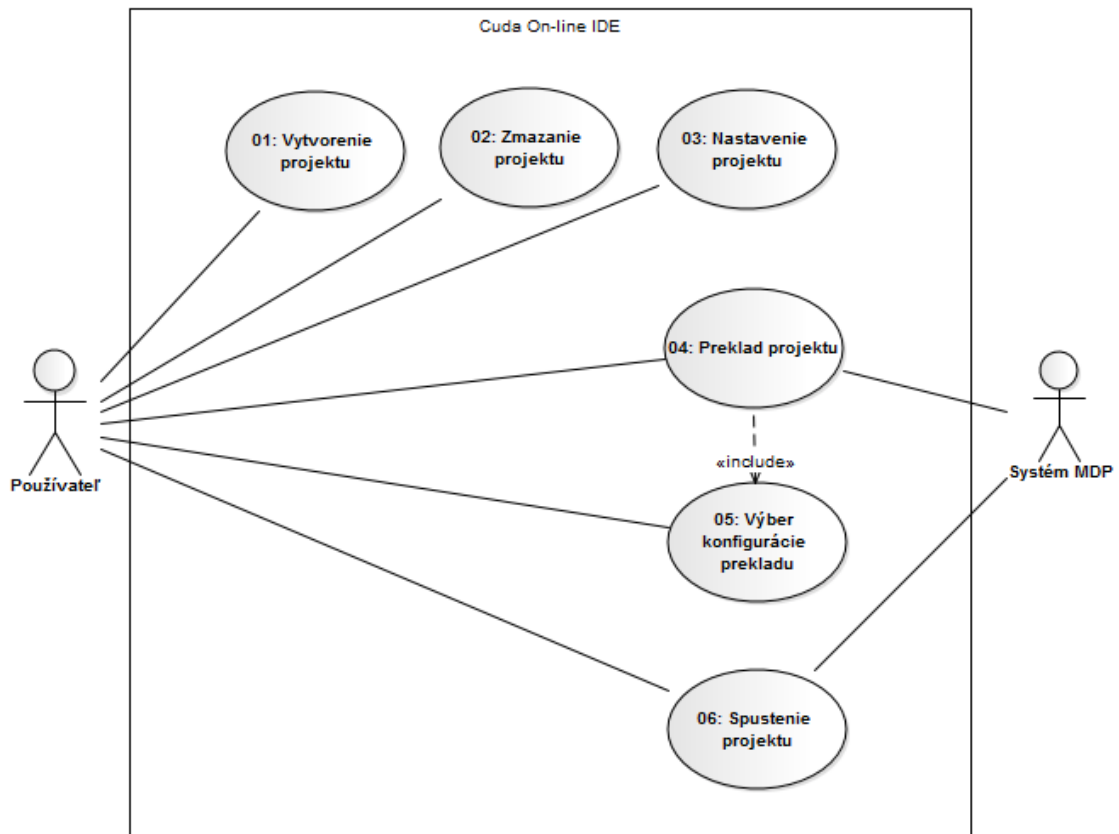
- prihlásenie a správa pracovného prostredia,
- správa projektu,
- správa súboru,
- ostatné funkcionality.

Model prípadov užitia na Obr. 28 zobrazuje, že používateľ sa bude môcť prihlásiť, odhlásiť a taktiež vykonávať kompletnú správu nad workspace.



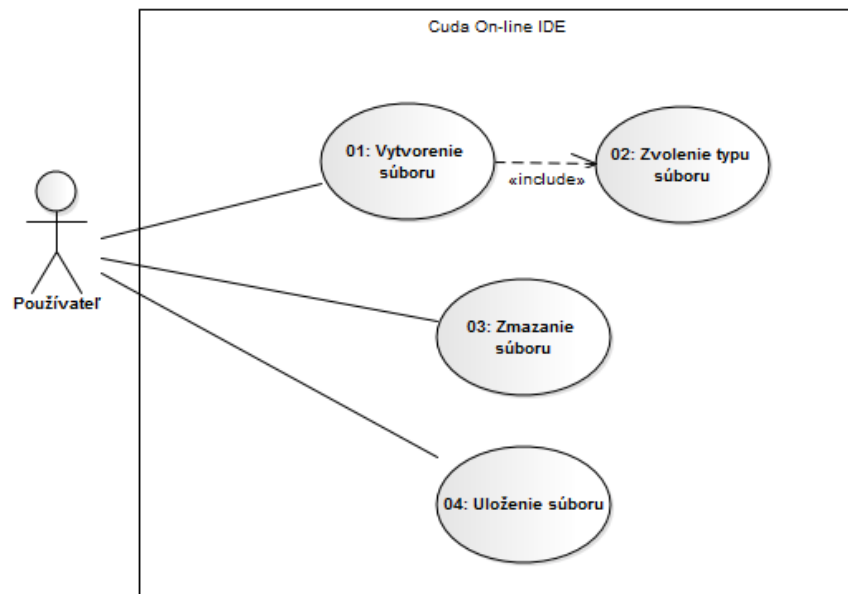
Obr. 28: Model prípadov užitia pre prihlásenie a správu workspace

Ďalším modelom prípadu užitia je správa projektu (Obr. 29). Používateľ preberá funkcionality kompletnej správy projektu, spúšťa preklad a spustenie projektu. Systém MDP bude vykonávať tento preklad a spustenie projektu.



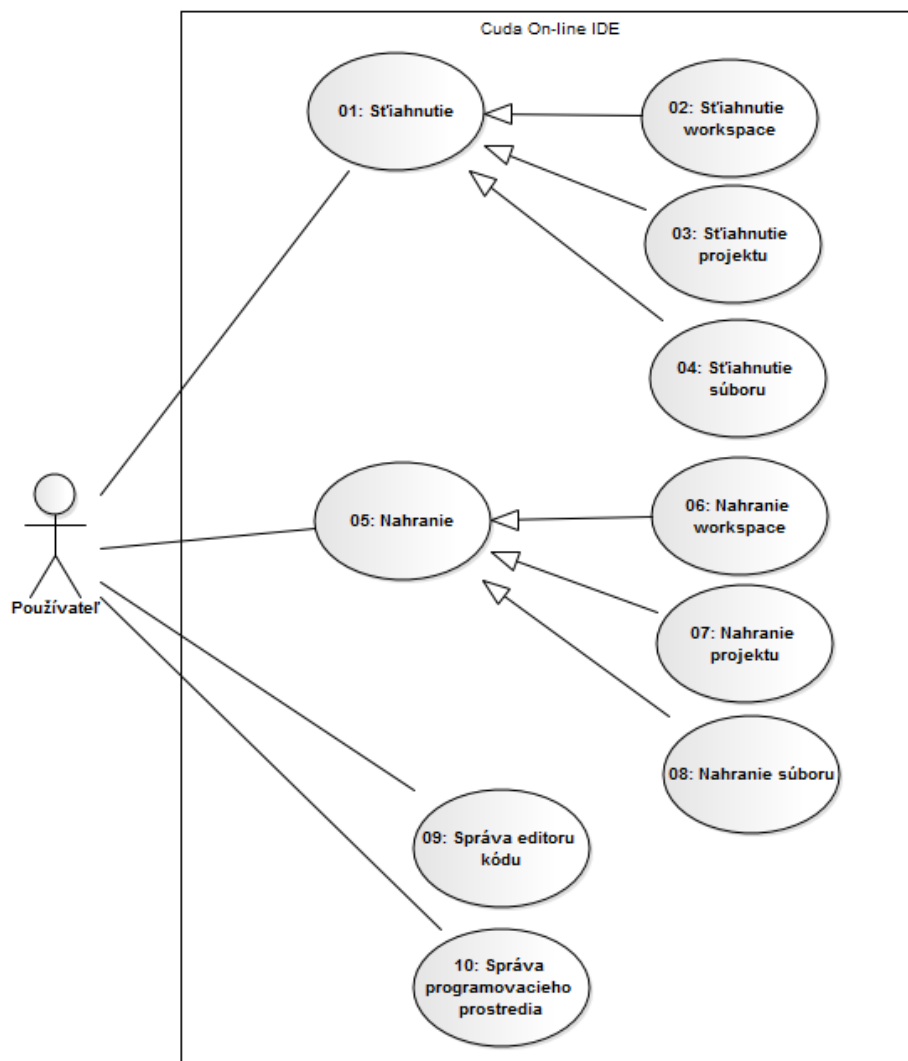
Obr. 29: Model prípadov užitia pre správu projektu

Ďalší model prípadu užitia znázorňuje, že používateľ preberá kompletnú správu nad súbormi. (Obr. 30).



Obr. 30: Model prípadov užitia pre správu súborov

Posledným modelom prípadu užitia sú ostatné funkcionality systému (Obr. 31). Tieto zahŕňajú sťahovanie a nahrávanie všetkých súčasti programového pracovného prostredia, projektu a súboru používateľom. Používateľ bude môcť spravovať editor kódu a nastavenie programového prostredia.



Obr. 31: Model prípadov užitia pre ostatné funkcionality

6.5 Návrh riešenia

Riešenie tohto projektu musí vychádzať z požiadaviek na systém. Keďže sa jedná o webovú aplikáciu, je potrebné, aby táto bohatá webová aplikácia bola navrhnutá s cieľom poskytnúť používateľovi prívetivé programovacie prostredie, ktoré sa drží zaužívaných štandardov pri desktopových programovacích prostrediach a musí byť dostupné pre použitie v rámci univerzitnej siete. Tzn., že webová aplikácia musí bežať na serveri v rámci tejto univerzitnej siete.

V rámci bezpečnosti prihlasovania je potrebné implementovať modul overovania prihlasovania užívateľov pomocou údajov do univerzitnej siete, tak aby mohol ktorýkoľvek študent alebo zamestnanec UTB pristupovať k tomuto IDE. Ďalej bude potrebné implementovanie

definovania vlastných užívateľov na základe konfiguračného súboru, ktorí dostanú oprávnenie pracovania s týmto programovacím prostredím.

Správa programového prostredia používateľa musí byť implementovaná nad jedinečným užívateľským priestorom tak, aby práca viacerých používateľov naraz neznamerala prepisovanie súborov navzájom, a aby IDE bolo schopné ukladať a obnovovať činnosť používateľa na základe ukladania štruktúry workspace, buď pomocou databáze alebo súboru pre zaznamenávanie štruktúry vo formáte XML. Štruktúra workspace bude zahŕňať údaje o aktuálnom workspace, všetkých projektoch a ich nastaveniach v rámci tohto workspace a o obsahu každého projektu. Táto štruktúra bude aktualizovaná pri každej zmene v rámci správy daného workspace, správy jeho projektov alebo správy jeho súborov.

Preklad a spúšťanie projektu bude riešené pomocou systému pre vzdialené spúšťanie kódu, kde projekt bude zasielaný na tento systém, ktorý následne vykoná jeho preklad a bude informovať webovú aplikáciu o jeho dokončení. Následne bude možné po úspešnom preklade vykonať spustenie projektu s vráteným výstupom aplikácie.

Nahrávanie workspace alebo projektov do aplikácie bude vo formáte ZIP, kde bude musieť takto zabalený archív obsahovať povinnú štruktúru, buď workspace alebo projektu, inak bude proces zamietnutý. Sťahovanie workspace alebo projektov do pracovnej stanice bude vo formáte ZIP. Nahrávanie alebo sťahovanie jednotlivých súborov nebude vyžadovať zabalené súbory vo formáte ZIP.

Editor kódu bude poskytovať funkcionality zvýrazňovania syntaxe C a C++ programovacích jazykov, automatické dopĺňanie písaného kódu, číslovania riadkov, vyhľadávania a prepisovania textu. V neposlednom rade bude poskytovať funkcionality tvorby štruktúry o kóde z uloženého textu súboru, teda zostavenie stromu na základe základných C a C++ objektov v texte súboru s podporou odkazovania v jeho rámci.

7 IMPLEMENTÁCIA

Na základe požiadaviek, modelu prípadov užitia a návrhu riešenia bolo možné pristúpiť k implementácii webovej aplikácie. Pre tvorbu bol vybraný vyššie popísaný nástroj Google Web Toolkit. Z pohľadu náročnosti aplikácie, GWT značne urýchľuje proces vývoja, keďže kód je písaný prevažne v jazyku Java aj na klientskej vrstve. Popis implementácie aplikácie sa bude skladať z tvorby aplikácie, jej štruktúry a výsledného programovacieho prostredia.

7.1 Tvorba aplikácie

Základným stavebným kameňom pri tvorbe CUDA On-line IDE pomocou GWT sú nasledujúce komponenty:

- hositeľská stránka,
- kaskádové štýly,
- XML modul,
- popisovač nasadenia web.xml,
- trieda vstupného bodu.

Hositeľská stránka

Hositeľská stránka je HTML dokument, ktorý implementuje kaskádové štýly aplikácie, javascriptové závislosti využívané v rámci klientskeho pohľadu webovej aplikácie a obsluhuje udalosť neaktívneho JavaScriptu v prehliadači. Táto stránka je odkazovaná pre automaticky generovaný javascriptový kód pomocou GWT pre zobrazenie dynamického obsahu stránky. Pre tento obsah môže byť definovaný buď celý obsah `<body>` alebo jednotlivá určená značka `<div>`.

`CudaOnlineIDE/war/CudaOnlineIDE.html` predstavuje hositeľskú stránku webovej aplikácie, implementuje kaskádové štýly aplikácie `CudaOnlineIDE/war/CudaOnlineIDE.css`, javascriptové závislosti pre ACE, teda javascriptové jadro editoru, možné témy vzhľadu a zvýrazňovač syntaxe pre jazyky C a C++ (Obr. 32).

```
<link type="text/css" rel="stylesheet" href="CudaOnlineIDE.css">
<link rel="icon" type="image/ico" href="favicon.ico"/>

<script src="ace/ace.js" type="text/javascript"></script>
<script src="ace/theme-eclipse.js" type="text/javascript"></script>
<script src="ace/theme-twilight.js" type="text/javascript"></script>
<script src="ace/theme-github.js" type="text/javascript"></script>
<script src="ace/mode-c_cpp.js" type="text/javascript"></script>
<script src="ace/mode-plain_text.js" type="text/javascript"></script>
<script src="ace/ext-language_tools.js" type="text/javascript"></script>
```

Obr. 32: Implementovanie kaskádových štýlov a javascriptových závislostí v súbore *CudaOnlineIDE.html*

Pre dynamicky generovaný obsah aplikácie je použitý celý obsah `<body>` (Obr. 33).

```
<body>
  <noscript>
    <div style="width: 22em; position: absolute;
      left: 50%; margin-left: -11em; color: red;
      background-color: white; border: 1px solid red;
      padding: 4px; font-family: sans-serif">
      Your web browser must have JavaScript enabled
      in order for this application to display correctly.
    </div>
  </noscript>
</body>
```

Obr. 33: Obsah `<body>` v súbore *CudaOnlineIDE.html*

XML modul aplikácie

XML modul aplikácie predstavuje dokument postavený najvyššie v hierarchii balíčkov GWT webovej aplikácie. Definuje názov modulu, triedu vstupného bodu aplikácie, zahŕňa ďalšie pridružené moduly, ktoré majú byť použité v klientskom rozhraní, použitú tému aplikácie a cesty pre Java kód, ktorý má byť preložený pomocou GWT do JavaScriptu.

Vo webovej aplikácii CUDA On-line IDE je XML modul umiestnený v *CudaOnlineIDE/src/cz/utb/fai/cudaonlineide/CudaOnlineIDE.gwt.xml*. Definuje názov modulu na *cudaonlineide*, zahŕňa knižnice Sencha GXT pre použitie výkonných widgetov, šablón a tému Neptune pre vzhľad programovacieho prostredia. Ako editor kódu je zahrnutý ACE upravený pre potreby GWT. Ako vstupný bod aplikácie je definovaná trieda *cz.utb.fai.cudaonlineide.client.CudaOnlineIDE*. Obsah tohto súboru je zobrazený v Obr. 34.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Google Web Toolkit version reference.
-->
<!DOCTYPE module PUBLIC "-//Google Inc.//DTD Google Web Toolkit 2.7.0//EN"
  "http://gwtproject.org/doctype/2.7.0/gwt-module.dtd">

<!-- Specify the module name. -->
<module rename-to='cudaonlineide'>

<!-- Inherit the core Web Toolkit stuff. -->
<inherits name='com.google.gwt.user.User' />

<!-- Inherit Sencha GXT core and Sencha Neptune theme. -->
<inherits name='com.sencha.gxt.ui.GXT' />
<inherits name='com.sencha.gxt.theme.neptune.Theme' />

<!-- Inherit Ace editor core. -->
<inherits name='edu.ycp.cs.dh.acegwt.AceGWT' />

<!-- Specify the app entry point class. -->
<entry-point class='cz.utb.fai.cudaonlineide.client.CudaOnlineIDE' />

<!-- Specify the paths for translatable code. -->
<source path='client' />
<source path='shared' />

<!-- Specify the style reset file. -->
<stylesheet src="reset.css" />
</module>
```

Obr. 34: Obsah CudaOnlineIDE.gwt.xml

Popisovač nasadenia web.xml

Súbor *web.xml* je používaný Java webovými aplikáciami pre určenie, ako má webový server obsluhovať webové požiadavky pre aplikáciu. Súbor definuje úvodnú HTML stránku webovej aplikácie a po doručení požiadavky mapuje jej URL cestu a určuje servlet s obsluhujúcou metódou.

Súbor je umiestnený v *CudaOnlineIDE/war/WEB-INF/web.xml* a definuje 6 servletov pre obsluhu požiadaviek na server:

- *coiServlet*,
- *loginService*,
- *downloadFileService*,
- *downloadZipService*,
- *uploadFileService*,
- *uploadZipService*.

Tieto servlety budú popísané v kapitole 7.2.3 Serverové rozhranie. *Web.xml* nastavuje úvodnú stránku aplikácie na *CudaOnlineIDE.html*.

Trieda vstupného bodu

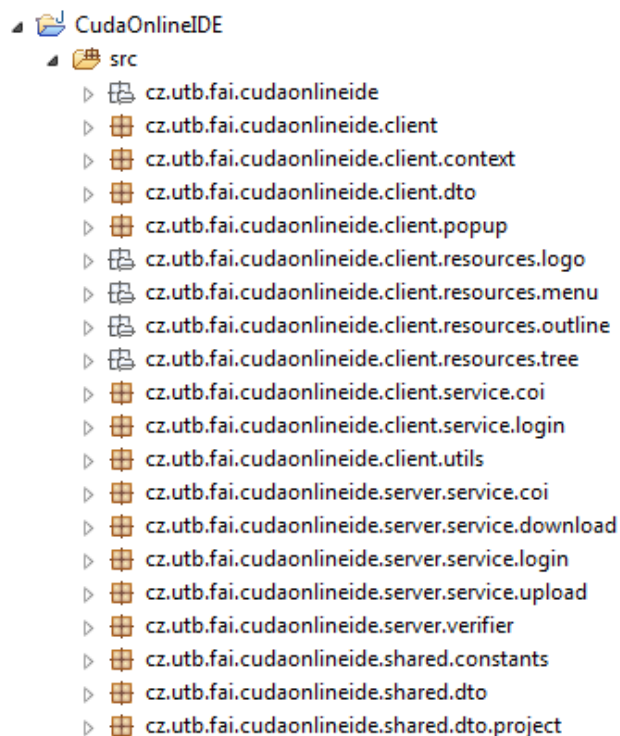
Trieda vstupného bodu implementuje rozhranie *EntryPoint*, ktoré obsahuje metódu *onModuleLoad*. Táto metóda je volaná pri spustení aplikácie CUDA On-line IDE.

Trieda *cz.utb.fai.cudaonlineide.client.CudaOnlineIDE* predstavuje vstupný bod webového IDE a implementuje všetky GWT moduly zahrnuté v XML modulu aplikácie. V metóde *onModuleLoad* je definované, ktorá oblasť hostiteľskej stránky je použitá pre generovanie dynamického obsahu aplikácie, v tomto prípade je použitý celý obsah *<body>*, taktiež Java kód, ktorý má byť generovaný.

7.2 Štruktúra aplikácie

Štruktúra aplikácie je delená na tri hlavné časti a to:

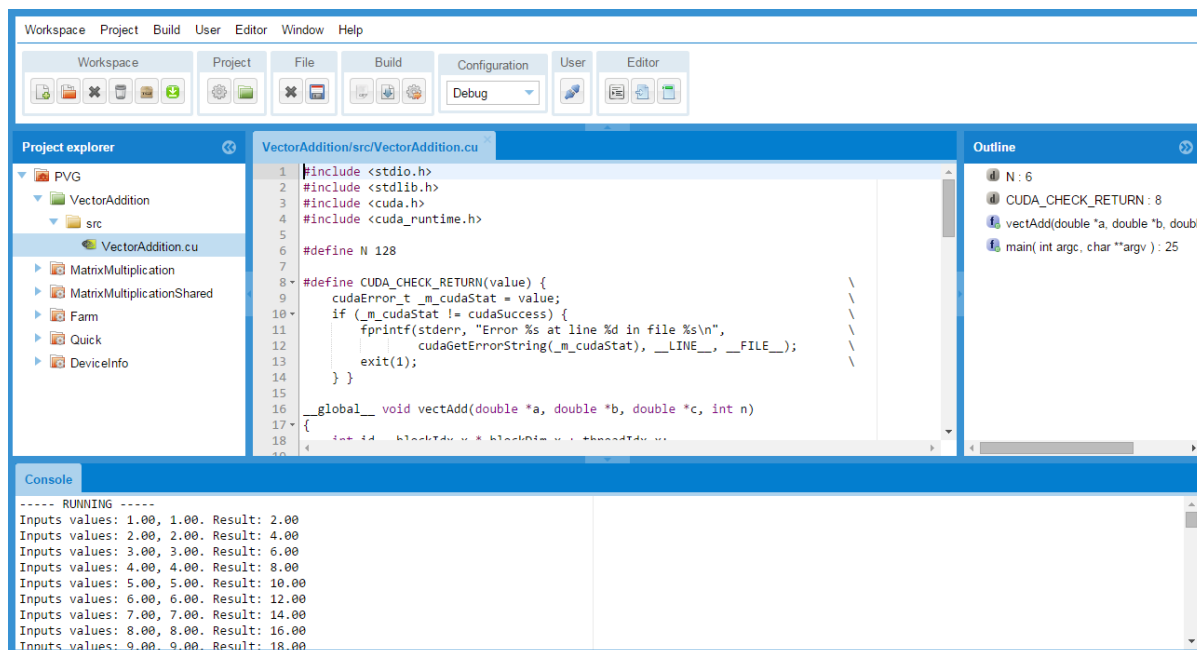
- kód klientskeho rozhrania - *cz.utb.fai.cudaonlineide.client*,
- zdieľaný kód - *cz.utb.fai.cudaonlineide.shared*,
- kód serverového rozhrania - *cz.utb.fai.cudaonlineide.server*.



Obr. 35: Štruktúra CUDA On-line IDE aplikácie

7.2.1 Klientske rozhranie

Klientske rozhranie spadá pod balíček *cz.utb.fai.cudaonlineide.client*. Na základe XML modulu aplikácie je určené, že práve tento Java kód bude pomocou GWT kompilátoru preložený do JavaScriptu. Klientske rozhranie obsahuje triedu vstupného bodu *CudaOnlineIDE*. Táto trieda poskytuje metódu *onModuleLoad* pre načítanie dynamického obsahu do hostiteľskej stránky. Toto rozhranie je postavené nad *BorderLayoutContainer*, ktorý poskytuje klasický vzhľad aplikácie pre programovacie prostredie. Prostredie je tak rozdelené na päť častí. Vrchný panel poskytuje priestor pre menu aplikácie a toolbar s často používanými činnosťami. Ľavý panel slúži ako projektový prieskumník, kde sa zobrazuje štruktúra pracovného prostredia s projektmi. Pravý panel zobrazuje hierarchickú štruktúru o aktuálne otvorenom súbore tzv. Outline. Stredný panel predstavuje editor kódu s možnosťou otvorenia viacerých súborov súčasne. Spodný panel je využitý ako konzola aplikácie pre zobrazenie výstupu aplikácie.



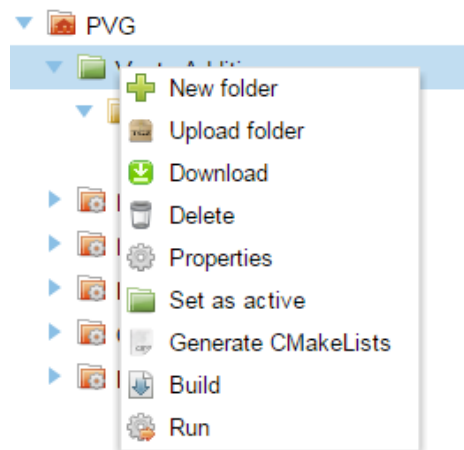
Obr. 36: BorderLayoutContainer aplikácie CUDA On-line IDE

Ďalšie podskupiny obsahujú doplňujúce zdroje a funkcionality k triede vstupného bodu.

cz.utb.fai.cudaonlineide.client.context

Balíček *context* obsahuje triedu *TreeContextMenu*, ktorá poskytuje funkcionality projektovému prieskumníku zobrazovať na položkách štruktúry pracovného prostredia, projektu

alebo súboru možné činnosti a obsluhovať tak zvolené udalosti. Trieda poskytuje metódy, ktoré na základe určenej činnosti pracujú nad zvolenou položkou. Kontextové menu je dynamicky menené na základe zvolenej položky. Rozdielne činnosti sú pri práci s pracovným prostredím, projektom alebo súborom. Jednotlivé položky kontextového menu vykonávajú činnosti, ktoré potrebujú byť spracovávané na serveri, ako tvorba novej zložky, preklad aplikácie a i., a preto musia byť vykonávané asynchrónne RPC volania metód na server s využitím servletu *COIService*.



Obr. 37: Kontextové menu pre projekt

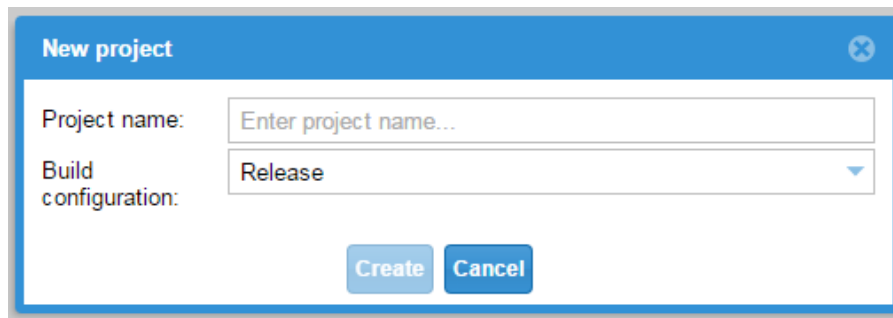
cz.utb.fai.cudaonlineide.client.dto

Balíček *dto* obsahuje triedu *COIData*, ktorá slúži na reprezentáciu objektu v stromovej štruktúre projektového prieskumníka. Na základe tohto objektu je potom docieľené rôzne kontextové menu a rôzne použitie ikon pri položkách pracovného prostredia, projektu alebo súboru.

cz.utb.fai.cudaonlineide.client.popup

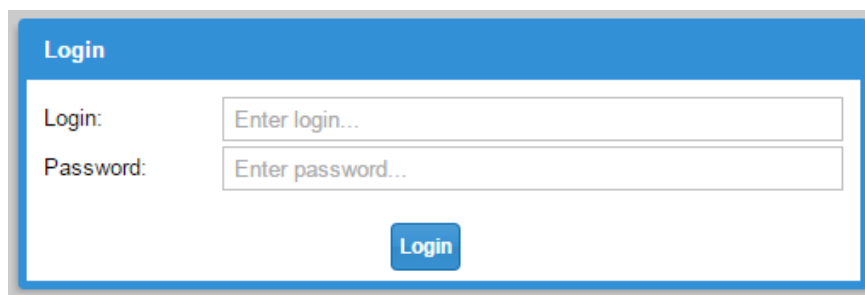
V tomto balíčku sú zastúpené dve triedy a to *PopUpWindow* a *Login*.

Trieda *PopUpWindow* poskytuje vyskakovacie okná a ich obsluhy týkajúce sa pracovného prostredia, projektu a súborov. Vyskakovacie okná sú modálne, teda nie je možné vykonať inú činnosť bez ich potvrdenia alebo zamietnutia.



Obr. 38: Vyskakovacie okno pre tvorbu nového projektu

Trieda *Login* zahŕňa vyskakovacie okno pre prihlásenie užívateľa do systému a vyžíva ser-
vlet *LoginService* pre obsluhu pokusu o prihlásenie.



Obr. 39: Prihlasovacie okno

cz.utb.fai.cudaonlineide.client.resources

Balíček *resources* neobsahuje žiadny Java kód, ale obsahuje zdroje obrázkov, ktoré sú im-
plementované v programovacom prostredí.

cz.utb.fai.cudaonlineide.client.utils

Tento balíček obsahuje utility *MenuToolBarIcons* a *Sha1Utils*.

MenuToolBarIcons predstavuje triedu *ClientBundle*, ktorá pracuje zo zdrojmi pre poskyto-
vanie efektívneho prístupu k dátam počas behu aplikácie.

```
public interface MenuToolBarIcons extends ClientBundle {  
    public MenuToolBarIcons PROVIDER = GWT.create(MenuToolBarIcons.class);  
    @Source("cz/utb/fai/cudaonlineide/client/resources/tree/folder.png")  
    ImageResource folder();  
}
```

Obr. 40: Použitie triedy *ClientBundle*

V priloženom kóde (Obr. 40) je zobrazené použitie triedy pre prácu s obrázkovým zdrojom. Táto trieda teda poskytuje zdroje obrázkov z balíčku *cz.utb.fai.cudaonlineide.resources*.

Sha1Utils poskytuje metódu pre získanie odtlačku zo zadaného textu. Tento odtlačok je potom použitý pre bezpečné prihlasovanie užívateľa na základe konfiguračného súboru.

cz.utb.fai.cudaonlineide.client.service

Balíček *service* obsahuje dve podskupiny *coi* a *login*. V každom z týchto balíčkov je klientska časť RPC mechanizmu, čiže klientske rozhranie pre službu vykonávanú na serveri a asynchrónne klientske rozhranie pre obsluhu návratových metód volania. V balíčku *coi* sa teda nachádzajú rozhrania *COIService* a *COIServiceAsync*, v balíčku *login* rozhrania *LoginService* a *LoginServiceAsync*.

7.2.2 Zdieľané objekty

Zdieľané objekty predstavujú časť aplikácie spadajúcej pod balíček *cz.utb.fai.cudaonlineide.shared*. Tieto objekty sú používané tak ako na strane klienta, tak aj na strane serveru, teda ich využitie je väčšinou viazané pre prácu s RPC službami, buď ako argument asynchrónneho volania, alebo tvoria návratovú hodnotu asynchrónneho volania. Zdieľané objekty musia byť preto serializovateľné, implementovať rozhranie *Serializable* a obsahovať prázdny konštruktor s rôznym prístupovým modifikátorom.

cz.utb.fai.cudaonlineide.shared.dto

Balíček obsahuje serializovateľné objekty *COIObject*, *COIWorkspace*, *COIProject*, *COIFolder*, *COIFile*, *COIEnum*, *COIUser*, *COIFileData* a podskupinu *project*, ktorá obsahuje objekty *COIBuildConfiguration*, *COIConfiguration*, *COICompiler*, *COILinker*. Všetky objekty okrem *COIUser* a *COIFileData* patria pod správu pracovného prostredia.

COIObject predstavuje rodičovskú triedu správy pracovného prostredia a obsahuje názov objektu, fyzickú cestu na serveri a typ objektu. Typ objektu môže byť pracovné prostredie - *WORKSPACE*, projekt - *PROJECT*, zložka - *FOLDER*, súbor - *FILE*. Objekty, ktoré dedia rodičovskú triedu dopĺňajú potrebné údaje podľa tohto typu. Štruktúru pracovného prostredia teda reprezentuje strom, ktorý tvoria:

- *COIWorkspace* - obsahuje 0 až N projektov,
- *COIProject* – obsahuje 0 až N zložiek,

- *COIFolder* – obsahuje 0 až N souborů,
- *COIFile*.

Podskupina *project* zahrnuje objekty pro správu konfigurace překladu a ukládá použité vlastnosti, cesty kompilátoru a linkeru pro vykonání překladu projektu, při použití troch možných konfigurací a to *DEBUG*, *RELEASE* a *CUSTOM*.

Ukládání struktury pracovního prostředí na server bude popsáno v části 7.2.3.1 Správa pracovního prostředí, Štruktúra pracovního prostředí.

Objekt *COIUser* reprezentuje uživatele aplikace a jedná se tedy o objekt, který je ukládán do session prohlíдача a obsahuje údaje o tomto uživateli.

Objekt *COIFileData* předává klientskému rozhraní informace o souborech pro tvorbu outline, čiže hierarchické struktury objektů v něm. Poskytuje tedy informace o třídách, strukturách, parametrech, pozicích výskytu, přístupnosti a dalších vlastnostech.

cz.utb.fai.cudaonlineide.shared.constants

Tento balíček poskytuje dvě třídy *COIConstants* a *WorkspaceConstants*.

COIConstants obsahuje konstanty používané při tvorbě klientského rozhraní a struktury pracovního prostředí. Odstraňuje tak potřebu používání textových řetězců při vytváření programovacího prostředí a zjednocuje ich používání.

WorkspaceConstants obsahuje konstanty pro nastavení pracovního prostředí aplikace, tedy pracovní složka programovacího prostředí Cuda On-line IDE na serveru a název konfiguračního souboru s přidáním uživateli pro dodatečnou autentizaci.

7.2.3 Serverové rozhraní

Serverové rozhraní spadající pod balíček *cz.utb.fai.cudaonlineide.server* poskytuje Java servlety, které vykonávají činnost na webovém serveru. Či už se jedná o servlety implementující mechanismus GWT RPC asynchronního volání metod nebo klasické HTTP servlety obsluhující události GET a POST.

Serverové rozhraní je rozděleno na dvě podskupiny *service*, implementující servlety webové aplikace a *verifier*, obsahující třídu *RPCVerifier*, pro kontrolu serializovaných objektů posílaných parametry asynchronního volání z klienta na server.

cz.utb.fai.cudaonlineide.server.service

Balíček *service* obsahuje 6 dostupných služieb:

- *COIServiceImpl* - servlet *coiServlet*,
- *LoginServiceImpl* - servlet *loginService*,
- *DownloadFileService* - servlet *downloadFileService*,
- *DownloadZipService* - servlet *downloadZipService*,
- *UploadFileService* - servlet *uploadFileService*,
- *UploadZipService* - servlet *uploadZipService*.

7.2.3.1 *Správa pracovného prostredia*

Trieda *COIServiceImpl* je implementácia rozhrania *COIService* pre GWT RPC asynchrónneho volania metód. Poskytuje metódy pre správu pracovného prostredia, ako je vytvorenie nových projektov, zložiek, súborov, ich mazanie alebo aktualizácie, ďalej metódy pre generovanie a úpravu konfiguračných súborov pre preklad projektu, následný preklad a spustenie projektu, a tiež metódu pre získanie objektov štruktúry súboru.

Štruktúra pracovného prostredia

Pod štruktúrou pracovného prostredia je potrebné si predstaviť údaje, ktoré charakterizujú toto aktuálne pracovné prostredie, v ňom zahrnuté projekty, ich nastavenia a súbory zo zdrojovými kódmi. Táto štruktúra musí byť ukladaná ku každému pracovnému prostrediu zvlášť pre jeho následné obnovenie. V projekte CUDA On-line IDE je štruktúra ukladaná do konfiguračného súboru pracovného prostredia vo formáte XML, v koreňovom adresári používateľa. Je postavený vedľa zložky s pracovným prostredím a obsahuje údaje o ňom. Názov je odvodený od tejto zložky, napr. pre pracovné prostredie *CUDATest* je konfiguračný súbor tohto prostredia *CUDATest_workspace.cws2*. Štruktúra konfiguračného súboru je znázornená v priloženom kóde (Obr. 41).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<workspace name="workspacename" path="workspacepath">
  <cws_ver>version</cws_ver>
  <project name="projectname" path="projectpath">
    <buildConfiguration active="Debug">
      <Debug>
        <compiler>
          <options>
            <option value="value1" />
            <option value="value2" />
          </options>
          <include_directories>...</include_directories>
          <preprocessors>...</preprocessors>
        </compiler>
        <linker>
          <options>...</options>
          <library_paths>...</library_paths>
          <library_names>...</library_names>
        </linker>
      </Debug>
      <Release>...</Release>
      <Custom>...</Custom>
    </buildConfiguration>
    <folder name="foldername" path="folderpath">
      <file name="filename1" path="filepath2" />
      <file name="filename1" path="filepath2" />
    </folder>
    <folder>...</folder>
  </project>
</workspace>
```

Obr. 41: Štruktúra konfiguračného súboru pracovného prostredia

Na základe tejto štruktúry môže byť po opätovnom spustení webovej aplikácie vybrané pracovné prostredie, používané pri predchádzajúcej práci v tomto IDE.

Generovanie konfiguračných súborov pre preklad

Pod generovaním konfiguračných súborov je možné si predstaviť tvorbu konfiguračného súboru *CMakeLists.txt* pre nástroj CMake, a na jeho základe vygenerovaný *Makefile* a ďalšie potrebné súbory pre preklad.

Súbor *CMakeLists.txt* dodržiava štruktúru tvorby *Makefile* pre CUDA aplikácie. Používateľ aplikácie má možnosť, okrem automatického generovania tohto súboru, tiež možnosť vykonávať jeho vlastné úpravy. Pri preklade aplikácie sa však vždy vykonáva generácia tohto súboru. Aby sa zabránilo prepísaniu užívateľských zmien, je pri automatickom generovaní vytvorená záloha *CMakeLists.txt.backup*. Pri zmene konfiguračného súboru užívateľom sú teda *CMakeLists.txt* a *CMakeLists.txt.backup* rôzne a aplikácia dáva používateľovi na výber generáciu nového súboru alebo ponechanie užívateľsky upraveného súboru pre následnú tvorbu *Makefile* nástrojom CMake.

Preklad a spustenie projektu

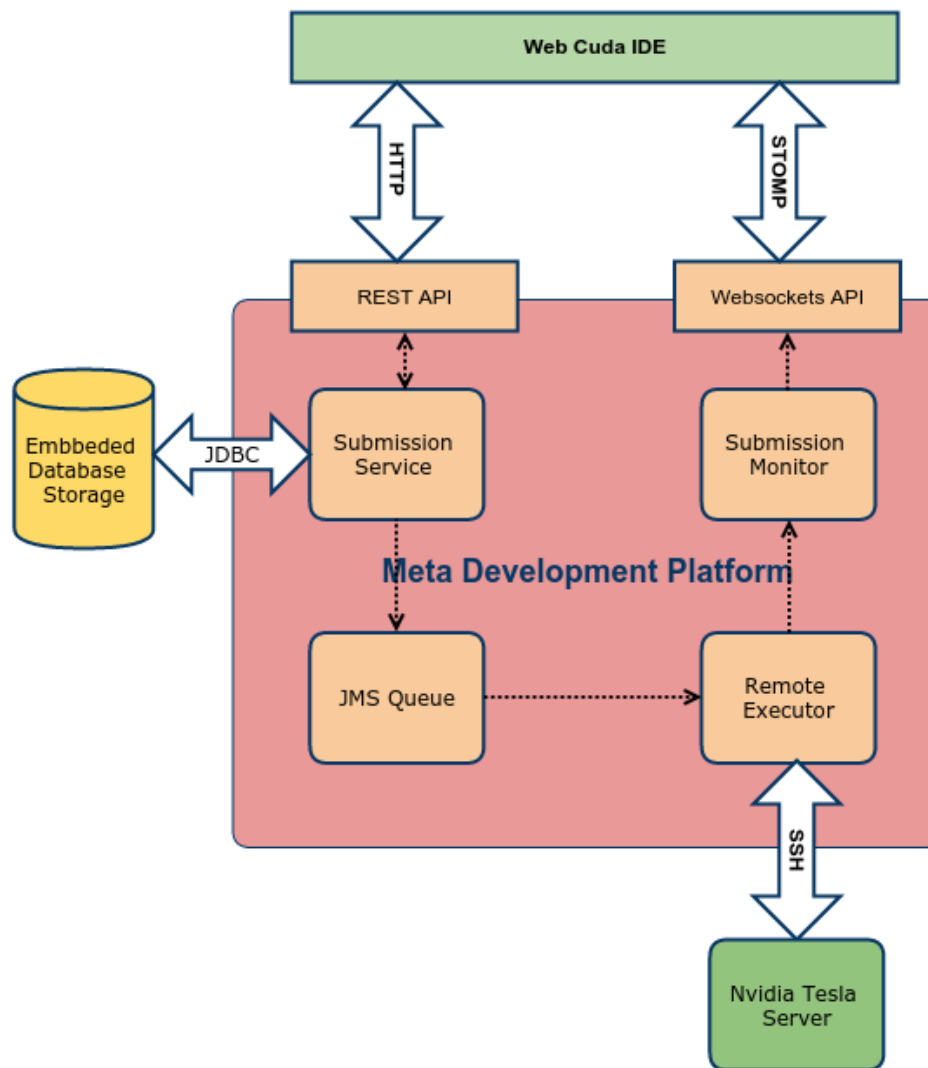
Po vytvorení konfiguračných súborov pre preklad je možné vykonať preklad aplikácie. Pre túto činnosť IDE využíva nižšie popísaný systém MDP.

Systém MDP

Systém MDP (Meta Development Platform) predstavuje platformu pre vzdialené spúšťanie kódu, ktorý je realizovaný Ing. Františkom Špačkom. Prototyp tohto systému vychádza zo systému APAC (Automatic Programming Assignment Checker) [37]. MDP je postavené na populárnom Java frameworku Spring 4. Zjednodušený komponentový diagram systému je vidieť na obrázku.

Systém poskytuje jednoduché integračné REST API pre kompiláciu a spúšťanie kódu. Odoslaný kód je najprv normalizovaný a je vytvorený pracovný priestor, pre kompiláciu a spustenie. Informácie o odovzdanom kóde sú udržiavané vo vstavanej databáze H2, Java SQL databáze.

Po prvotnom spracovaní sa zaradí úloha do fronty pre ďalšie spracovanie. Táto fronta je asynchrónne spracovávaná. Mechanizmus fronty je implementovaný pomocou štandardizovaného JMS API, kde je použitý Apache ActiveMQ.



Obr. 42: Komponentový diagram systému MDP

Samotná kompilácia a spúšťanie kompatibilného Nvidia CUDA kódu prebieha na vzdialenom serveri s grafickou kartou Nvidia Tesla. Tento vzdialený server sa pravidelne dopytuje pomocou bash skriptu na MDP API, aby bolo zaručené, že server je živý a je možné použiť pre spracovanie. Vďaka tomuto mechanizmu je pomerne jednoduché rozšíriť MDP systém o ďalšie vzdialené servery. Na základe zaregistrovaných zariadení (serverov) sa vytvorí tzv. pool SSH pripojení, aby nebolo nutné pre každú úlohu vytvárať nové spojenie. Pool SSH spojení je implementovaný za pomoci knižnice Apache Commons Pool. Pri spracovaní sa získa platná inštancia SSH spojenia a nad získanou inštanciou sa vykonávajú požadované operácie.

Pri preklade CUDA zdrojového kódu, je vygenerovaný príkaz pre preklad a spustenie kódu

pomocou SSH *exec* kanálu. Zdrojové kódy sú so vzdialeným serverom zosynchronizované pomocou SFTP kanálu SSH spojenia.

Ak je kompilácia úspešná prebehne spätná synchronizácia s MDP, aby bolo možné preložený program na požiadanie spustiť. Po spracovaní kódu je inštancia SSH spojenia vrátená do poolu pre ďalšie použitie. Systém automaticky ukončí program, ktorý beží viac ako 30 sekúnd. Do budúca je uvažované o izolácii vzdialeného spúšťania CUDA kódu na serveri s Tesla kartou za pomoci kontajnerovej platformy Docker, ktorá je využívaná aj v systéme APAC.

REST API systému MDP

Pre kompiláciu a spustenie kódu je využívané REST API dostupné na adrese <http://195.178.90.55:8080/mdp/api/submissions>, kde API vykonáva nasledujúce činnosti.

Nahratie a preklad projektu <http://195.178.90.55:8080/mdp/api/submissions?lang=CUDA>, kde *content-type* je nastavený na *application/octet-stream* a telo požiadavky obsahuje ZIP projektu. Požiadavka vráti identifikátor *uuid* tohto procesu v JSON formáte.

Spustenie projektu obstaráva <http://195.178.90.55:8080/mdp/api/submissions/uuid/execute>, pri nahradení *uuid* identifikátorom procesu. V tele požiadavky je možné predať argumenty spustenia v JSON formáte, prázdne bez argumentov `{}` alebo pri použití argumentov, napr. `{"arguments": "-a 5"}`.

Výsledok prekladu a spustenia je vrátený v JSON formáte a dostupný na adrese <http://195.178.90.55:8080/mdp/api/submissions/uuid>, pri nahradení *uuid* identifikátorom procesu.

Obsluhu tohto API predstavuje Java projekt *BuildServer*, priložený k diplomovej práci. Vykonáva kontrolu parametrov, ktoré majú byť zaslané na systém MDP.

Tvorba Outline - štruktúra súboru so zdrojovým kódom

Tvorba štruktúry súboru pre lepšie navigovanie v rozsiahlom súbore s kódom je vytváraná pomocou nástroja Exuberant Ctags. Pri otvorení súboru, jeho uložení alebo prepínaní medzi jednotlivými otvorenými súbormi je pri súboroch s koncovkou `.h`, `.c`, `.cpp`, `.cuh`, `.cu` využitý tento nástroj a príkazom `„ctags -R -u --fields=+aKmnsSztI-klf --langmap=c++:+.h.cuh.cu“` je vygenerovaný súbor s indexmi objektov dostupných v zdrojovom kóde. Medzi informácie, ktoré tento nástroj získa patrí názov objektu, typ

objektu, pozícia riadku, prístupnosť a i. Na základe týchto informácií je získané pole objektov *COIFileData*, ktoré túto štruktúru charakterizuje.

7.2.3.2 Správa prihlasovania používateľa

Trieda *LoginServiceImpl* spravuje prihlasovanie používateľa. Pre prihlasovanie používateľov pomocou údajov pre pripojenie do univerzitnej siete bol využitý mailový server *imap.utb.cz*, ktorý autentifikuje užívateľov na základe týchto údajov. Ak táto autentifikácia zlyhá, prichádza na rad autentifikácia užívateľa na základe konfiguračného súboru *userToLog.cuf*, ktorý sa nachádza v koreňovom adresári programovacieho prostredia. Tento súbor má štruktúru XML (Obr. 43).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<users>
  <user name="test" password="a94a8fe5ccb19ba61c4c0873d391e987982fbbd3" />
  <user>...</user>
</users>
```

Obr. 43: Štruktúra konfiguračného súboru pre dodatočné prihlasovanie používateľov

Heslo je uložené ako odtlačok jednosmerného hashovacieho algoritmu SHA1. Ak aj táto autentifikácia zlyhá, používateľ nie je pripustený k obsluhu programu. Ak je jedna z autentifikácií úspešná, používateľovi je vytvorený priestor na základe jeho mena, v ktorom môže vytvárať svoje pracovné prostredia. Priestor je vytvorený iba v tom prípade, ak doteraz neexistuje, inak je do už vytvoreného priestoru používateľ presmerovaný. Následne je objekt *COIUser* uložený do session prehliadača a je pripustený k práci v programovacom prostredí. Pri odhlásení používateľa je objekt *COIUser* odobraný zo session.

7.2.3.3 Sťahovanie a nahrávanie súborov

Sťahovanie jednotlivých súborov a zložiek prebieha pomocou servletov *DownloadFileService* a *DownloadZipService*. Zložky sú pred stiahnutím zabalené do archívu ZIP. Nahrávanie jednotlivých súborov na server je vykonávané servletom *UploadFileService*. Nahrávanie zložiek servletom *UploadZipService*, tie však musia dodržiavať štruktúru pracovného prostredia a byť zabalené do archívu ZIP.

7.3 Nasadenie aplikácie

Nasadenie aplikácie je uskutočnené na webovom serveri s grafickou kartou Nvidia Tesla a so systémom Ubuntu. V prípade webového serveru je použitý Apache Tomcat vo verzii 7.0.52 s nastaveným prístupom pre webové administrátorské rozhranie. Pre nasadenie aplikácie je vytvorená pracovná zložka programovacieho prostredia, kde vlastníkom zložky je nastavený používateľ *tomcat*. V tejto koreňovej zložke je vytvorený konfiguračný súbor používateľov *userToLog.cuf* s rovnakým oprávnením. Ďalšie potrebné nástroje, ktoré sú nainštalované pre správne fungovanie IDE sú CMake a Exuberant Ctags. CUDA On-line IDE je dostupné v rámci univerzitnej siete na adrese <http://10.51.11.99:8080/CudaOnlineIDE>. Používateľská dokumentácia pre popis fungovania IDE a zoznámenie sa s týmto programovacím prostredím je priložená v prílohách diplomovej práce (Príloha P I: Používateľská dokumentácia). Zdrojové kódy aplikácie sú dostupné v prílohách na CD (Príloha P II: Popis priložených súborov na CD) alebo v Git repozitároch:

- <https://github.com/rough23/cudaonlineide>,
- <https://github.com/rough23/buildserver>,
- <https://github.com/rough23/imapauthentication>.

8 ZHODNOTENIE

8.1 Zhodnotenie CUDA On-line IDE

Riešenie diplomovej práce poskytuje veľa výhod, ale aj nevýhod. Medzi výhody určite patrí, že pri využití CUDA On-line IDE študentmi odpadá povinnosť zložitého nastavovania prostredia Nvidia Nsight a využívania virtuálneho systému pre prácu s ním. Stačí otvoriť webový prehliadač, zadať webovú adresu IDE a programovať. Webové IDE poskytuje základné funkcionality desktopového IDE, preto je práca s ním jednoduchá a dostačujúca pre projekty vytvárané v predmete Paralelné výpočty na grafických kartách. Programovacie prostredie dovoľuje prístup k programovaniu na grafických kartách študentom a zamestnancom UTB na základe ich prihlasovacích údajov do univerzitnej siete alebo zadaným používateľom v konfiguračnom súbore pre prístup. Pokrýva teda širší okruh ľudí pre vývoj CUDA aplikácií, ktorí nemusia mať na vzdialenom serveri s grafickou kartou s podporou CUDA technológie vytvorené unikátne používateľské kontá pre vymedzenie priestoru pre programovanie.

Medzi nevýhody doterajšieho riešenia patrí nemožnosť interakcie s aplikáciou počas jej behu. Systém MDP zdrojový kód preloží alebo vykoná spustenie a výstup z týchto akcií je poskytnutý až po ich dokončení. To obmedzuje prácu pri aplikáciách, ktoré obsahujú konzolové menu s výbermi možností alebo čakaním na potvrdenie a i. Ďalšou nevýhodou aplikácie je nemožnosť využitia pokročilých ladiacich a profilovacích nástrojov, ktoré ponúka programovacie prostredie Nvidia Nsight.

8.2 Odporúčania k budúcemu vývoju

CUDA On-line ide ma určité možnosti budúceho vývoja, ktoré by tomuto IDE poskytovali viac možností využitia, jednoduchšiu a lepšiu prácu s ním. Nasledujúce odporúčania teda určujú smer, akým by sa vývoj tohto systému mal uberať.

8.2.1 Implementácia interakcie s CUDA aplikáciou

CUDA On-line IDE doteraz nedisponuje interakciou s CUDA aplikáciou počas jej behu. Systém MDP už teraz obsahuje websocketové API pre zobrazovanie výsledku v reálnom čase, ale nie je ešte vo webovom rozhraní implementované. Je potreba teda zakomponovať toto API a vytvoriť rozhranie pre komunikáciu s aplikáciou.

8.2.2 Rozšírenie o ďalšie vzdialené servery

Využitie systému pre vzdialené spúšťanie kódu MDP prináša výhodu, že tento systém môže byť pomerne jednoducho rozšírený o ďalšie vzdialené servery. Vývoj CUDA aplikácií tak nemusí prebiehať len na teraz použitej Nvidia Tesla karte.

8.2.3 Rozšírenie automatického dopĺňania kódu

Automatické dopĺňanie v editore kódu teraz implementuje funkcionality, kde toto dopĺňanie prebieha na základe objektov použitých v aktuálnom otvorenom súbore. Teda napr. po vložení CUDA hlavičiek z nich nie sú v automatickom dopĺňovaní poskytované parametre a metódy. Pre pohodlnejšie použitie programovacieho prostredia je táto funkcionality veľmi žiadaná.

8.2.4 Obsluha chybových oznámení

Po zobrazení chybového oznámenia pri zostavovaní aplikácie v konzole programovacieho prostredia, nie je možné navigovať po kliknutí na chybu na miesto v kóde, ktoré túto chybu spôsobilo. Z hľadiska komfortnosti a rýchlejšieho vývoja je implementácia tejto obsluhy vhodná.

8.2.5 Možnosť rozšírenia na rôzne programovacie jazyky

Z hľadiska jednoduchosti použitia by sa webové programovacie prostredie mohlo zo zameraného prostredia pre tvorbu CUDA aplikácií zmeniť na univerzálne prostredie. To by študentom zjednodušovalo prácu v kurzoch programovania, odpadla by im časť zdĺhavého nastavovania behového či vývojového prostredia a mohli by tak hneď pristúpiť k programovaniu zadaných úkolov.

ZÁVER

V tejto diplomovej práci som sa zaoberal tvorbou CUDA On-line programovacieho prostredia pre tvorbu a testovanie CUDA aplikácií. Cieľom bolo poskytnúť IDE, ktoré v plnom rozsahu nahradí doteraz používané technológie pri tvorbe CUDA aplikácií v predmete Paralelné výpočty na grafických kartách. Po analýze dostupných technológií a frameworkov bol pre vývoj využitý nástroj Google Web Toolkit, ktorý poskytol sofistikovaný nástroj pre tvorbu programovacieho prostredia.

Na základe požiadaviek a modelov prípadov užitia bolo zostavené programovacie prostredie spĺňajúce základné charakteristiky desktopových programovacích nástrojov. Klientske prostredie aplikácie využilo nadstavbu Sencha GXT pre docielenie výsledného vzhľadu a ďalších funkcionalít ako vyskakovacie okná alebo menu aplikácie. Na serverovej strane webovej aplikácie boli implementované servlety pre správu štruktúry pracovného prostredia aplikácie, používateľského prihlásenia, prekladu aplikácie a doplnkových funkcionalít. Pre správu štruktúry pracovného prostredia bol vybraný súbor vo formáte XML, ktorý obsahuje všetky údaje potrebné k popísaniu a obnoveniu pracovného prostredia. Používateľské prihlásenie bolo navrhnuté využitím IMAP autentifikácie s overovaním proti mailovému serveru `imap.utb.cz`, kde používatelia pre prístup k aplikácii využívajú prihlasovacie údaje k prístupu do univerzitetnej siete. Nadstavbou tejto autentifikácie je konfiguračný súbor používateľov, ktorý poskytuje prihlásenie aj novo nadefinovaným používateľom. Hlavnou časťou IDE je preklad a spustenie vytvorených CUDA aplikácií, kde bol implementovaný systém Meta Development Platform pre vzdialené spúšťanie kódu, vytvorený Ing. Františkom Špačkom. Všetky časti aplikácie, teda popis jej štruktúry, boli v diplomovej práci riadne popísané.

Diplomová práca tiež poskytuje návrhy na vylepšenie programovacieho prostredia a smerovanie vývoja aplikácie do budúcnosti.

Ciele diplomovej práce boli splnené a výstupy aplikácie prinášajú požadované výsledky. Systém bol preto nasadený do testovacieho režimu na server s grafickou kartou Nvidia Tesla dostupný na adrese <http://10.51.11.99:8080/CudaOnlineIDE>.

ZOZNAM POUŽITEJ LITERATURY

- [1] Web Application. 2014. *TechTerms.com* [online]. [cit. 2015-03-15]. Dostupné z: http://techterms.com/definition/web_application.
- [2] Web Application Development. © 2006-. *VTech Solution* [online]. [cit. 2015-03-15]. Dostupné z: <https://vtechsolution.com/web-application-development/>.
- [3] 7 Difference Between RIA and Traditional Web Application. 2009. *Flex Tutorial: Rich Internet Application Development by Adobe Flex* [online]. [cit. 2015-03-15]. Dostupné z: <http://flectutorial.org/2009/07/31/7-difference-between-ria-and-traditional-web-application/>.
- [4] Rich Internet Applications (RIA): Web Applications having Features & Functionality of Desktop Application. 2007. *MS TechTalk* [online]. [cit. 2015-03-15]. Dostupné z: <http://mstechtalk.com/rich-internet-applicationsria-web-applications-having-features-functionality-of-desktop-application/>.
- [5] Whitepaper: Rich Internet Applications. 2008. *The Server Labs: IT Architects* [online]. [cit. 2015-03-15]. Dostupné z: <http://www.theserverlabs.com/resources/whitepapers.html?download=4:rich-internet-applications-frameworks-evaluation>.
- [6] HOGAN, Brian P. 2011. *HTML5 a CSS3: výukový kurz webového vývojáře*. Vyd. 1. Brno: Computer Press, 272 s. ISBN 978-80-251-3576-1.
- [7] SCHMITT, Christopher. 2010. *CSS cookbook*. 3rd ed. Sebastopol, CA: O'Reilly. ISBN 978-0-596-15593-3.
- [8] Introduction to XML. 2015. *W3schools.com* [online]. [cit. 2015-03-18]. Dostupné z: http://www.w3schools.com/xml/xml_what_is.asp.
- [9] JavaScript: Introduction. 2015. *MDN: Mozilla Developer Network* [online]. [cit. 2015-03-18]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript.3F.
- [10] AJAX Introduction. 2015. *W3schools.com* [online]. [cit. 2015-03-18]. Dostupné z: http://www.w3schools.com/ajax/ajax_intro.asp.
- [11] Flash Tutorial. 2015. *Site Ground* [online]. [cit. 2015-03-18]. Dostupné z: <https://www.siteground.com/tutorials/flash/>.

- [12] PHP 5 Introduction. 2015. *W3schools.com* [online]. [cit. 2015-03-18]. Dostupné z: http://www.w3schools.com/php/php_intro.asp.
- [13] GONCALVES, Antonio. 2013. *Beginning Java EE 7*. Berkley: APress. ISBN 978-143-0246-268..
- [14] The Python Tutorial. 2015. *Python* [online]. [cit. 2015-05-11]. Dostupné z: <https://docs.python.org/3/tutorial/index.html>.
- [15] An Introduction to the jQuery UI Library: Getting Started. 2015. *HTML Goodies: The ultimate html resource* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.htmlgoodies.com/beyond/javascript/an-introduction-to-the-jquery-ui-library-getting-started.html>.
- [16] BHAVA, Nagarajan. 2013. *Instant Ext JS starter find out what Ext JS actually is, what you can do with it, and why it's so great*. Online-Ausg. Birmingham, England: Packt Publishing. ISBN 978-178-2166-108.
- [17] Django. 2015. *Full Stack Python* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.fullstackpython.com/django.html>.
- [18] Rich Ajax Platform, Part 2: Developing applications. 2007. *IBM developerWorks: IBM's resource for developers and IT professionals* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.ibm.com/developerworks/library/os-eclipse-richajax2/>.
- [19] ADAM TACY, Robert Hanson. 2012. *GWT in action*. Second edition. Greenwich, Conn: Manning. ISBN 978-193-5182-849.
- [20] GWT Getting Started. 2009. *EMC Community Network* [online]. [cit. 2015-05-11]. Dostupné z: <https://community.emc.com/docs/DOC-3005>.
- [21] CUDA. 2015. *NVIDIA* [online]. [cit. 2015-05-11]. Dostupné z: http://www.nvidia.com/object/cuda_home_new.html.
- [22] SANDERS, Jason. *CUDA by example*. 1st print. Upper Saddle River. ISBN 978-0-13-138768-3.
- [23] CUDA C Programming Guide. 2015. *NVIDIA* [online]. (v7.0) [cit. 2015-05-11]. Dostupné z: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.
- [24] NVIDIA's Next Generation CUDATM Compute Architecture: Kepler TM GK110/210. 2014. *NVIDIA* [online]. (v1.1) [cit. 2015-05-11]. Dostupné z: <http://international.download.nvidia.com/pdf/kepler/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf>.

- [25] Using GWT RPC. 2014. *GWT Project* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.gwtproject.org/doc/latest/tutorial/RPC.html>.
- [26] JSNI. 2014. *GWT Project* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJSNI.html>.
- [27] Sencha GXT. 2015. *Sencha* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.sencha.com/products/gxt/>.
- [28] ALEKSA VUKOTIC, James Goodwill. 2011. *Apache Tomcat 7*. New Edition. S.l.: Apress. ISBN 978-143-0237-235.
- [29] Apache Commons Proper. *Apache Commons* [online]. [cit. 2015-05-11]. Dostupné z: <https://commons.apache.org/>.
- [30] Apache HttpComponents. *Apache Software Foundation* [online]. [cit. 2015-05-11]. Dostupné z: <https://hc.apache.org/>.
- [31] Simple Logging Facade for Java. *SLF4J* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.slf4j.org/>.
- [32] Zip4j. *Zip4j* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.lingala.net/zip4j/>.
- [33] ACE: Built for Code. 2015. *ACE* [online]. [cit. 2015-05-11]. Dostupné z: <http://ace.c9.io/#nav=about>.
- [34] What is ctags? *Exuberant Ctags* [online]. [cit. 2015-05-11]. Dostupné z: <http://ctags.sourceforge.net/whatis.html>.
- [35] About CMake. *CMake* [online]. [cit. 2015-05-11]. Dostupné z: <http://www.cmake.org/overview/>.
- [36] Nsight Eclipse Edition Getting Started Guide. 2015. *NVIDIA Developer Zone: CUDA Toolkit Documentation* [online]. [cit. 2015-05-11]. Dostupné z: <http://docs.nvidia.com/cuda/nsight-eclipse-edition-getting-started-guide/#axzz3ZqIfTWND>.
- [37] ŠPAČEK, František, Radomír SOHLICH a Tomáš DULÍK. 2015. Docker as Platform for Assignments Evaluation. *Procedia Engineering, Volume 100, Pages 1665-1671* [online]. [cit. 2015-05-11]. ISSN 1877-7058. Dostupné z: <http://www.sciencedirect.com/science/article/pii/S1877705815005688>.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

ACE	Ajax.org Code Editor
AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript and XML
ALU	Arithmetic Logic Unit
ANSI	American National Standards Institute
APAC	Automatic Programming Assignment Checker
API	Application Programming Interface
BSD	Berkeley Software Distribution
CDDL	Common Development and Distribution License
CMS	Content Management System
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CUDA	Compute Unified Device Architecture
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
EE	Enterprise Edition
EL	Expression Language
FPU	Floating-point Unit
GCC	GNU Compiler Collection
GLSL	OpenGL Shading Language
GNU	GNU's Not Unix
GPGPU	General-purpose Computing on Graphics Processing Units
GPL	General Public License

GPU	Graphics Processing Unit
GUI	Graphical User Interface
GWT	Google Web Toolkit
GXT	Ext Google Web Toolkit
HLSL	High-Level Shader Language
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
IT	Information Technology
JSNI	JavaScript Native Interface
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MDP	Meta Development Platform
MIT	Massachusetts Institute of Technology
MVC	Model View Controller
MVVM	Model View ViewModel
NVCC	Nvidia CUDA Compiler
OS	Operating System
PHP	Hypertext Preprocessor
POP	Post Office Protocol
PTX	Parallel Thread Execution
RAP	Remote Application Platform
RCP	Rich Client Platform
REST	Representational State Transfer

RIA	Rich Internet Application
RSS	Rich Site Summary
RWT	RAP Widget Toolkit
SDK	Software Development Kit
SE	Standard Edition
SFTP	SSH File Transfer Protocol
SIMT	Single Instruction, Multiple Thread
SMTP	Simple Mail Transfer Protocol
SMX	Streaming Multiprocessor
SQL	Structured Query Language
SSH	Secure Shell
SWT	Standard Widget Toolkit
UI	User Interface
XML	Extensible Markup Language
A i.	A iné
Napr.	Napríklad
Obr.	Obrázok
Tab.	Tabuľka
Tzn.	To znamená
Tzv.	Takzvaný
Vs.	Verzus

ZOZNAM OBRÁZKOV

<i>Obr. 1: Komponenty webovej aplikácie [2]</i>	12
<i>Obr. 2: Rozhodovací strom pre výber RIA technológie [5]</i>	15
<i>Obr. 3: Typy architektúr RIA technológií [5]</i>	15
<i>Obr. 4: Tvorba štruktúry stránky pomocou HTML</i>	17
<i>Obr. 5: CSS formátovanie</i>	18
<i>Obr. 6: Asynchrónna komunikácia zo serverom [10]</i>	20
<i>Obr. 7: Logo javascriptového frameworku jQuery UI</i>	23
<i>Obr. 8: Logo javascriptového frameworku ExtJS</i>	24
<i>Obr. 9: Logo Python frameworku Django</i>	24
<i>Obr. 10: RAP architektúra [18]</i>	25
<i>Obr. 11: Logo frameworku Remote Application Platform</i>	25
<i>Obr. 12: GWT architektúra [20]</i>	26
<i>Obr. 13: Logo nástroja Google Web Toolkit</i>	26
<i>Obr. 14: CPU vs. GPU architektúra [23]</i>	27
<i>Obr. 15: Aplikácia GPGPU Nvidia [23]</i>	28
<i>Obr. 16: CUDA Kepler architektúra [24]</i>	29
<i>Obr. 17: Streamovací procesor SMX [24]</i>	30
<i>Obr. 18: Plánovač warpov architektúry Kepler [24]</i>	30
<i>Obr. 19: Kompilácia CUDA kódu</i>	32
<i>Obr. 20: CUDA C kernel [22]</i>	33
<i>Obr. 21: Priebeh kompilácie kódu Java do JavaScriptu [19]</i>	36
<i>Obr. 22: Tvorba klientskeho rozhrania pomocou triedy EntryPoint</i>	37
<i>Obr. 23: Implementácia GWT RPC [25]</i>	38
<i>Obr. 24: Ukážka JSNI metódy</i>	39
<i>Obr. 25: Nastavenie SSH spojenia so vzdialeným prekladovým serverom</i>	44
<i>Obr. 26: Princíp prekladu na vzdialenom systéme [36]</i>	45
<i>Obr. 27: Aktéri webovej aplikácie</i>	47
<i>Obr. 28: Model prípadov užitia pre prihlásenie a správu workspace</i>	48
<i>Obr. 29: Model prípadov užitia pre správu projektu</i>	49
<i>Obr. 30: Model prípadov užitia pre správu súborov</i>	50
<i>Obr. 31: Model prípadov užitia pre ostatné funkcionality</i>	51

<i>Obr. 32: Implementovanie kaskádových štýlov a javascriptových závislostí v súbore CudaOnlineIDE.html</i>	54
<i>Obr. 33: Obsah <body> v súbore CudaOnlineIDE.html</i>	54
<i>Obr. 34: Obsah CudaOnlineIDE.gwt.xml</i>	55
<i>Obr. 35: Štruktúra CUDA On-line IDE aplikácie</i>	56
<i>Obr. 36: BorderLayoutContainer aplikácie CUDA On-line IDE</i>	57
<i>Obr. 37: Kontextové menu pre projekt</i>	58
<i>Obr. 38: Vyskakovacie okno pre tvorbu nového projektu</i>	59
<i>Obr. 39: Prihlasovacie okno</i>	59
<i>Obr. 40: Použitie triedy ClientBundle</i>	59
<i>Obr. 41: Štruktúra konfiguračného súboru pracovného prostredia</i>	63
<i>Obr. 42: Komponentový diagram systému MDP</i>	65
<i>Obr. 43: Štruktúra konfiguračného súboru pre dodatočné prihlasovanie používateľov</i>	67

ZOZNAM TABULIEK

<i>Tab. 1: Porovnanie programovacích jazykov JavaScript a Java [9]</i>	<i>19</i>
<i>Tab. 2: Tabuľka špecifikácií jednotlivých Tomcat server verzií [28]</i>	<i>41</i>

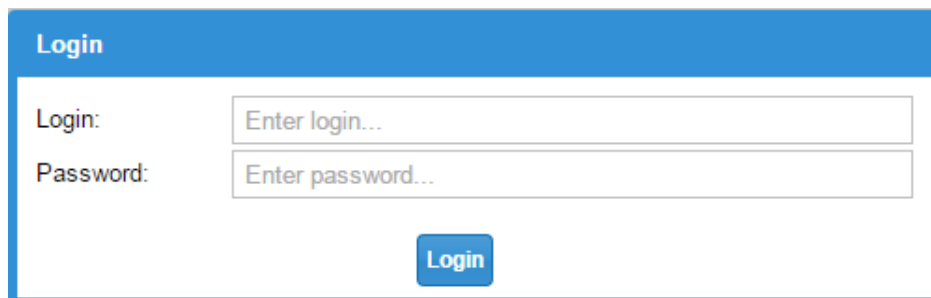
ZOZNAM PRÍLOH

- P I Používateľská dokumentácia
- P II Popis priložených súborov na disku CD

PRÍLOHA P I: POUŽÍVATEĽSKÁ DOKUMENTÁCIA

Spustenie

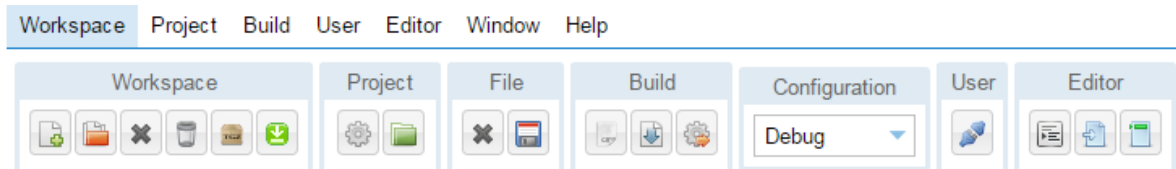
Po spustení aplikácie na URL adrese <http://10.51.11.99:8080/CudaOnlineIDE>, programové prostredie žiada užívateľa o prihlásenie. K prihláseniu je potrebné, aby študent použil svoje prihlasovacie údaje do univerzitickej siete. Po úspešnom prihlásení aplikácia automaticky vytvorí používateľský priestor na vzdialenom prekladovom serveri.



Prihlasovacie okno aplikácie

Programovacie prostredie

Po úspešnom prihlásení je študentovi dovolené pracovať v programovacom prostredí. Toto prostredie je rozdelené na päť častí.

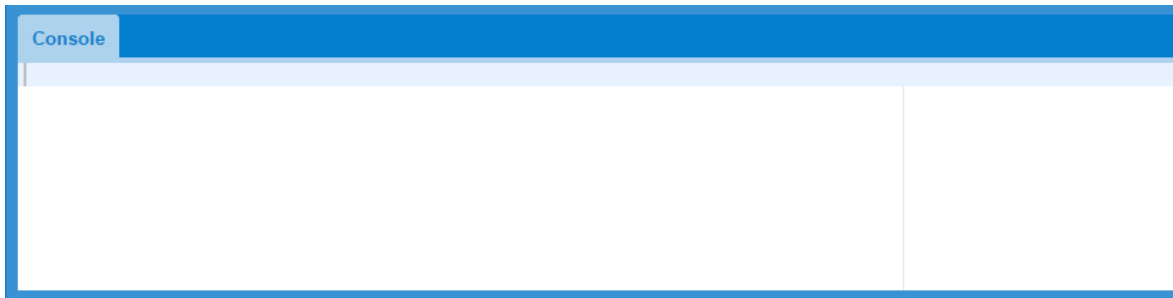


Menu a toolbar aplikácie

Menu poskytuje všetky funkcionality aplikácie, toolbar najčastejšie používané. Menu je rozdelené na sedem častí:

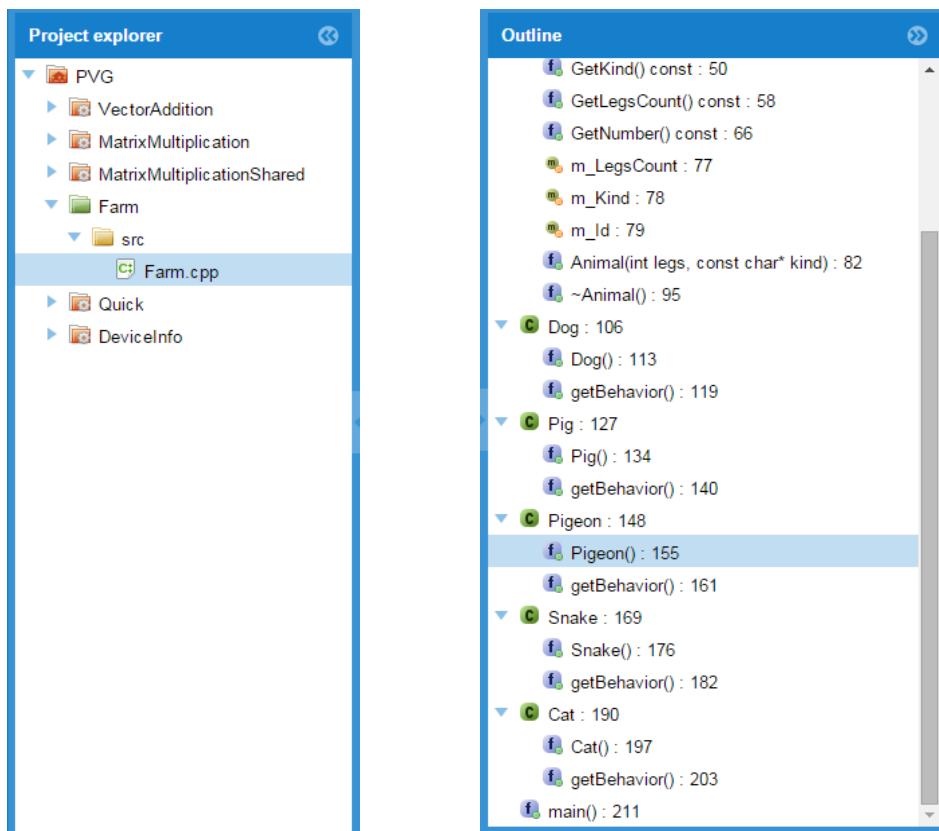
- pracovné prostredie (Workspace),
- projekt (Project),
- preklad a spustenie (Build),
- používateľ (User),
- editor (Editor),
- okná (Window),
- pomoc (Help).

Toolbar obsahuje navyše ešte časť konfigurácia (Configuration).



Konzola aplikácie (Console)

Konzola aplikácie zobrazuje výstup po preklade a spustení projektu. Služi len na čítanie.



Projektový prieskumník (Project explorer) a štruktúra aktuálneho súboru (Outline)

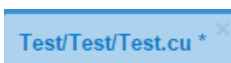
Projektový prieskumník poskytuje stromovú štruktúru aktuálneho pracovného prostredia. Táto stromová štruktúra obsahuje kontextové menu, ktoré je dynamicky menené na základe zvolenej položky.

Outline spracováva štruktúru aktuálne otvoreného súboru so zdrojovým kódom. Zobrazuje údaje o triedach, štruktúrach, členoch a ich umiestnení v súbore. Po dvoj kliku na položku prebieha navigácia v editore kódu.

```
Farm/src/Farm.cpp
138     * @return const char* Animal sounds.
139     */
140     const char* getBehavior(){
141         return "\"kvik kvik\"";
142     }
143 };
144
145 /**
146  * Pigeon class inherits Animal.
147  */
148 class Pigeon : public Animal
149 {
150 public:
151
152     /**
153     * Constructor.
154     */
155     Pigeon() : Animal(2, "holub") {}
156
```

Editor kódu s možnosťou otvorenia viacerých súborov súčasne

Editor kódu je posledná časť prostredia. Editor dovoľuje možnosť otvorenia viacerých súborov súčasne. Zvýrazňuje syntax jazyku C a C++. Po stlačení CTRL + Medzerník zobrazuje automatické dopĺňanie kódu na základe parsovaných častí kódu. Nastavenia editoru sú menené pomocou menu Editor. Neuložený zmenený súbor obsahuje pri názve hviezdičku. Pre zápis zmeny na disk vzdialeného servera je potrebné vykonať uloženie.

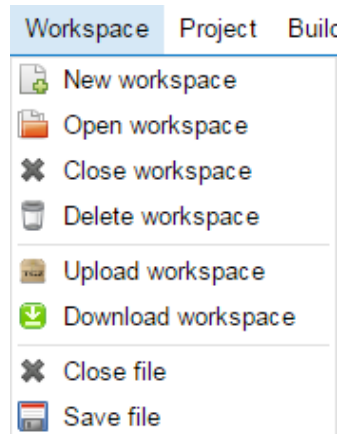


Zmena v súbore označená hviezdičkou

Samozrejmosťou editoru sú akcie na základe kombinácií tlačidiel:

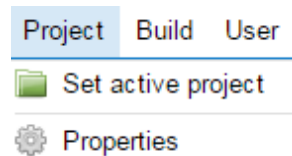
- CTRL + S => uloženie súboru,
- CTRL + + => zväčšenie textu,
- CTRL + - => zmenšenie textu,
- CTRL + 0 => základná veľkosť textu,
- CTRL + / => riadkový komentár,
- CTRL + I => zapnutie zobrazenia na celú obrazovku,
- CTRL + SHIFT + I => ukončenie zobrazenia na celú obrazovku,
- CTRL + Q => základná téma,
- CTRL + W => nočná téma,
- CTRL + E => XCode téma.

Menu a toolbar



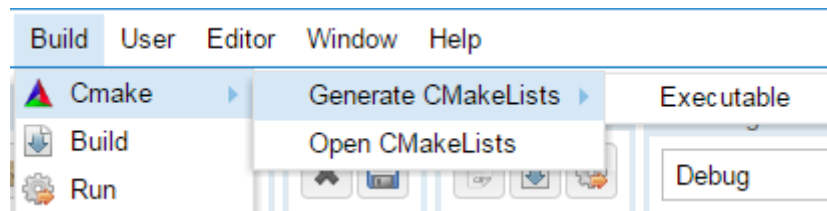
Menu workspace

Menu Workspace spravuje pracovné prostredia používateľa. Poskytuje možnosť vytvorenia (New workspace), otvorenia (Open workspace), zatvorenia (Close workspace) alebo zmazania (Delete workspace) pracovného prostredia. Toto menu obsahuje taktiež uloženie (Save file) alebo zatvorenie (Close file) aktuálne aktívneho otvoreného súboru.



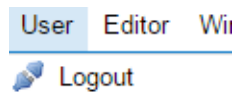
Menu project

Projektové menu ponúka možnosť nastavenia aktívneho projektu (Set active project) a možnosť zmien vlastností aktívneho projektu (Properties). Pri nastavení aktívneho projektu je možné využívať položky pre prácu s projektom, ktoré sú spojené s touto funkcionalitou.



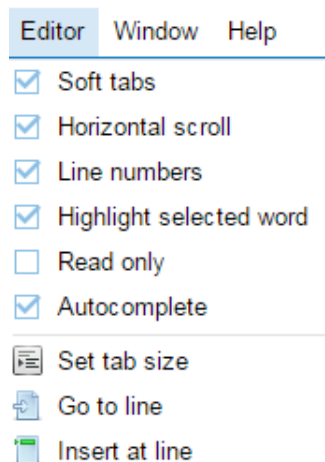
Menu build

Položka Build pokrýva generovanie (Generate CMakeLists) a editáciu (Open CMakeLists) konfiguračného súboru pre preklad. Ďalšími možnosťami sú preklad (Build) a spustenie (Run) projektu.



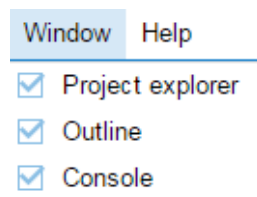
Menu user

Menu User zabezpečuje odhlásenie prihláseného užívateľa (Logout).



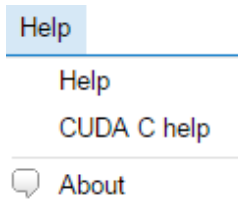
Menu editor

Nastavenie editoru ponúka menu Editor. Možnosti tohto menu sú zapnutie/vypnutie mäkkých tabulátorov (Soft tabs), vodorovného posuvníka (Horizontal scroll), číslovania riadkov (Line numbers), zvýrazňovania vybraného slova (Highlight selected word), módu len pre čítanie (Read only) a automatického dopĺňania kódu (Autocomplete). Tiež je možné nastaviť veľkosť tabulátorov (Set tab size), skočiť na riadok v kóde (Go to line) alebo pridať text k aktuálnemu miestu kurzora (Insert at line).



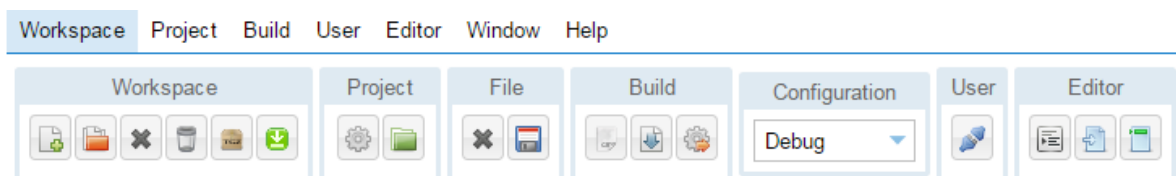
Menu window

Položka Window spravuje okná aplikácie, teda zapnutie/vypnutie projektového prieskumníka (Project explorer), štruktúry aktuálneho súboru (Outline) a konzole (Console).



Menu help

Posledné menu aplikácie je Help. Toto menu obsahuje túto používateľskú príručku (Help), príručku programovania v CUDA C (CUDA C help) a informácie o aplikácii (About).

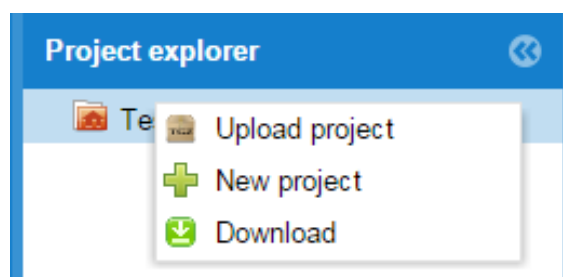


Toolbar aplikácie

Toolbar poskytuje rýchlu prístupnosť k položkám menu. Obsahuje navyše výber aktuálnej konfigurácie projektu (Configuration). Položky pre projekt, preklad, spustenie a nastavenie konfigurácie sú použiteľné až po zvolení aktívneho projektu.

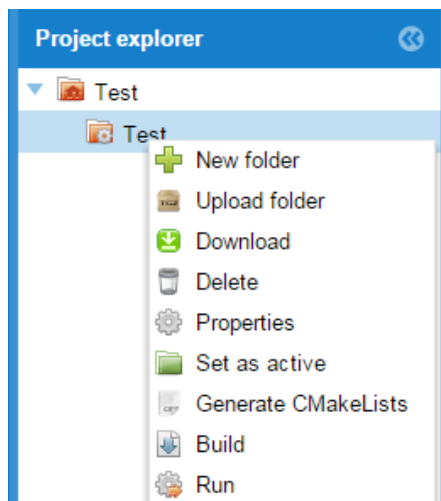
Kontextové menu

Kontextové menu položiek je dostupné v projektovom prieskumníku. Toto menu je automaticky generované na základe zvolenej položky. Rôzne menu je pre pracovné prostredie, projekt, zložku alebo súbor.



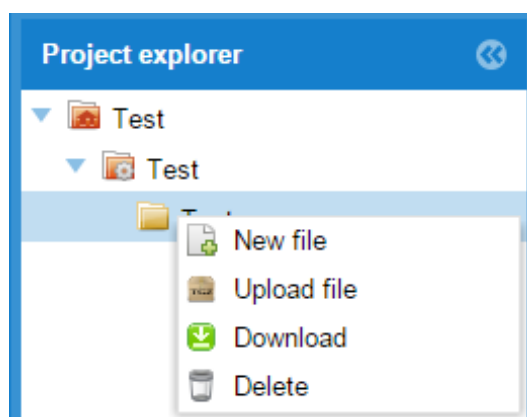
Kontextové menu pracovného prostredia

Kontextové menu pracovného prostredia ponúka možnosti vytvorenia nového projektu (New project), nahrania projektu (Upload project) alebo stiahnutie tohto pracovného prostredia (Download).



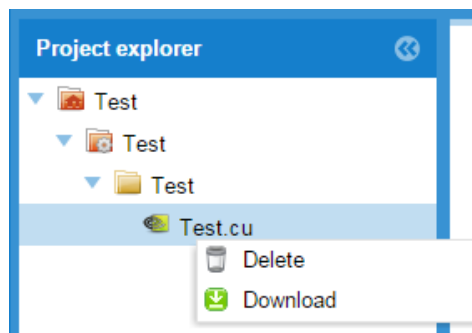
Kontextové menu projektu

Kontextové menu projektu obsahuje vytvorenie zložky (New folder), nahranie zložky (Upload folder) a prácu s týmto projektom ako stiahnutie (Download), zmazanie (Delete), nastavenie vlastností (Properties), nastavenie ako aktívny projekt (Set as active), generovanie konfiguračného súboru pre preklad (Generate CMakeLists), preklad (Build) a spustenie (Run).



Kontextové menu zložky

Kontextové menu zložky poskytuje vytvorenie súboru (New file), nahranie súboru (Upload file), stiahnutie tejto zložky (Download) a jej zmazanie (Delete).



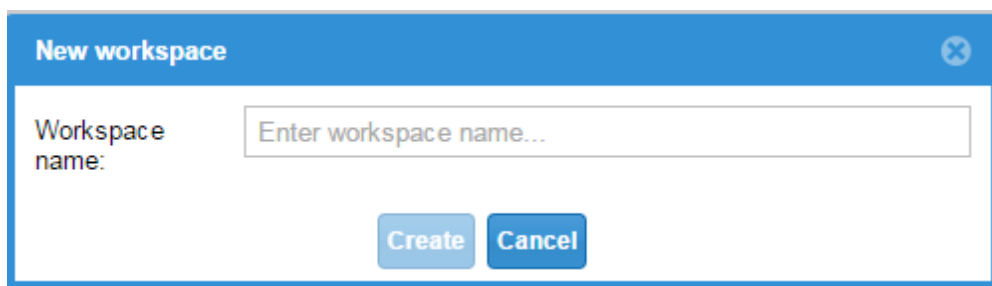
Kontextové menu súboru

Posledné kontextové menu umožňuje zmazanie tohto súboru (Delete) a jeho stiahnutie (Download).

Z kontextových menu aplikácie je vidieť, že štruktúra pracovného prostredia je pevne daná. Pracovné prostredia obsahuje projekty, tie obsahujú zložky a až tie súbory so zdrojovými kódmi.

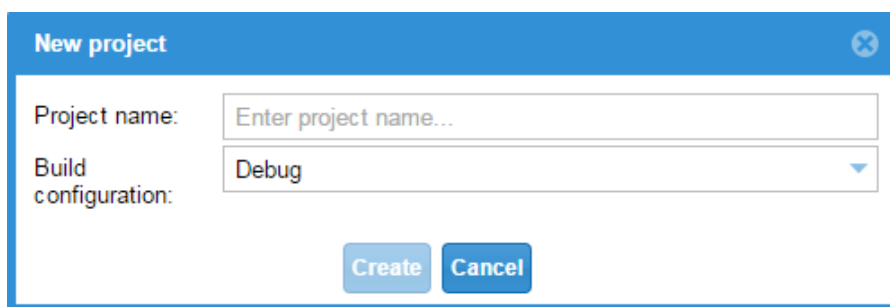
Vyskakovacie okná

Vyskakovacie okná aplikácie sú modálne, takže práca v programovacom prostredí môže pokračovať až po ich potvrdení alebo zrušení.



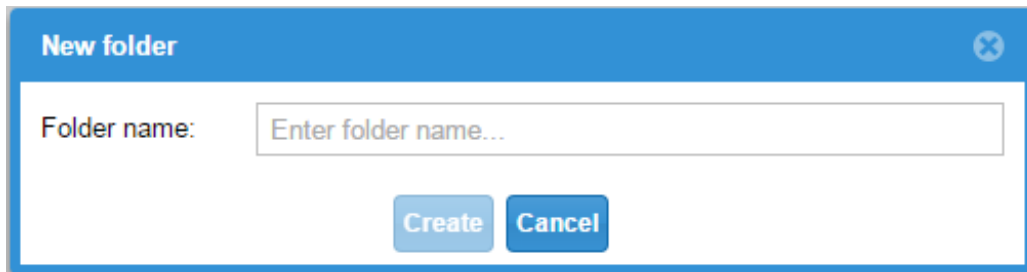
Okno pre vytvorenie nového pracovného prostredia

Okno je zobrazené po žiadosti o nové pracovné prostredie. Obsahuje jedno povinne vyplnené pole a to jeho názov.



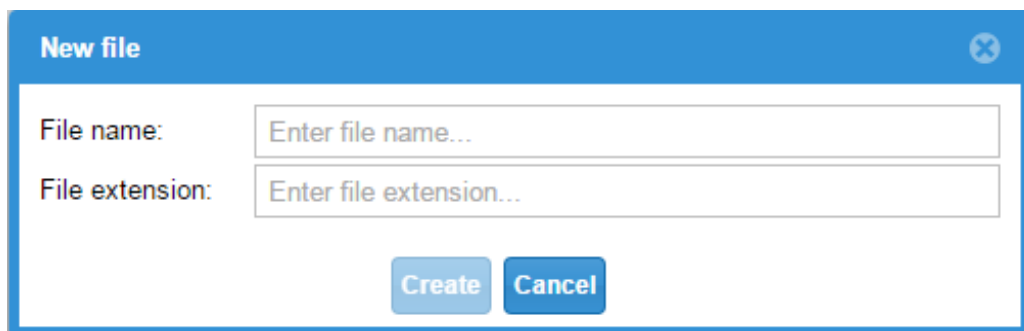
Okno pre vytvorenie nového projektu

Okno pre nový projekt obsahuje povinné pole názvu, taktiež je možné zvoliť aktívnu konfiguráciu projektu (Debug alebo Release).



Okno pre vytvorenie novej zložky

Vytvorenie novej zložky obsahuje povinné pole pre názov.



Okno pre vytvorenie súboru

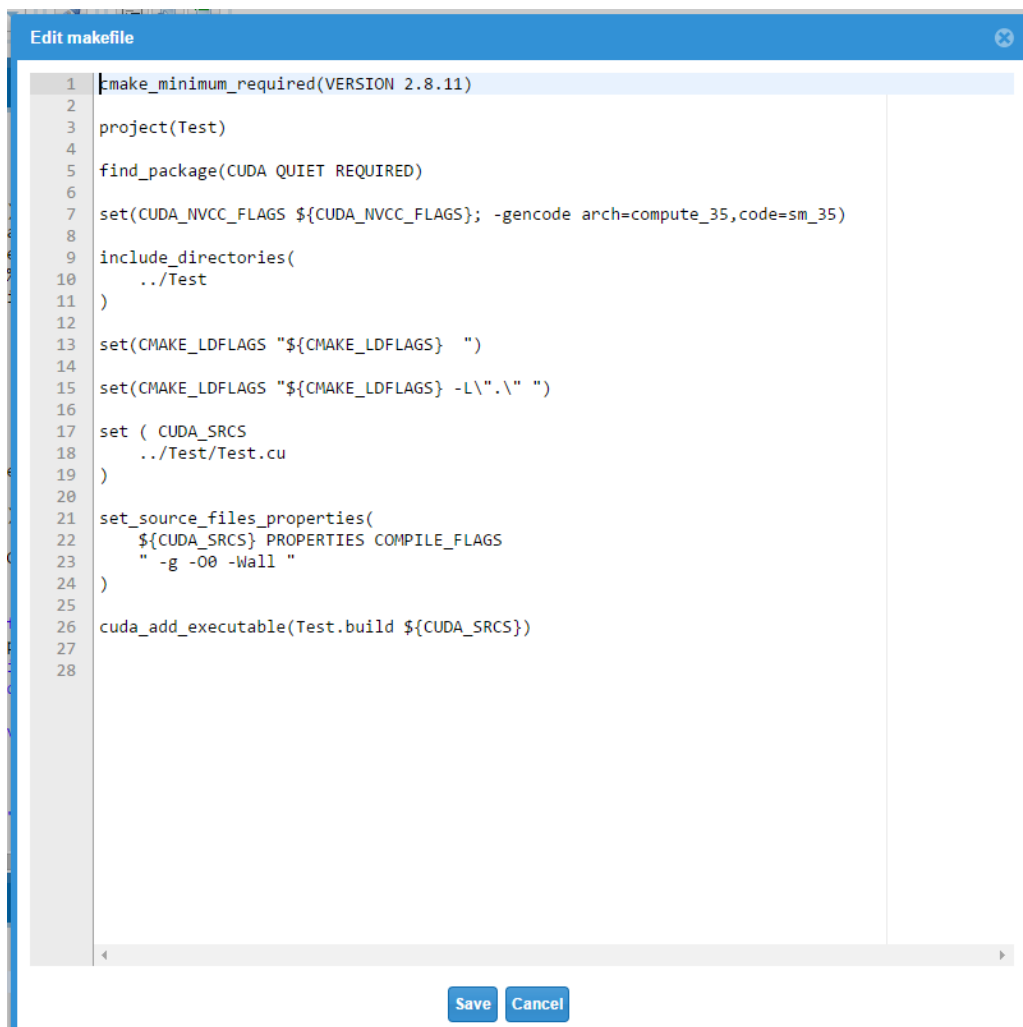
Vytvorenie súboru obsahuje povinné polia pre názov súboru a jeho príponu.



Okno pre nahranie ZIP súboru

Preklad a spustenie projektu

Po vytvorení projektu so zdrojovými kódmi je možné pristúpiť ku generácii konfiguračného súboru pre preklad, prekladu a spustení projektu. Generácia konfiguračného súboru je voliteľná položka. Je tiež automaticky vykonávaná pri preklade projektu. Možnosť jeho generácie je preto, že užívateľ obsah tohto súbor môže meniť.

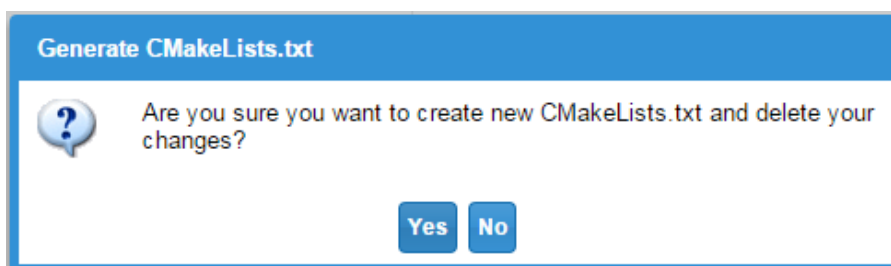


```
1 cmake_minimum_required(VERSION 2.8.11)
2
3 project(Test)
4
5 find_package(CUDA QUIET REQUIRED)
6
7 set(CUDA_NVCC_FLAGS ${CUDA_NVCC_FLAGS}; -gencode arch=compute_35,code=sm_35)
8
9 include_directories(
10     ../Test
11 )
12
13 set(CMAKE_LDFLAGS "${CMAKE_LDFLAGS} ")
14
15 set(CMAKE_LDFLAGS "${CMAKE_LDFLAGS} -L\".\" ")
16
17 set ( CUDA_SRCS
18     ../Test/Test.cu
19 )
20
21 set_source_files_properties(
22     ${CUDA_SRCS} PROPERTIES COMPILE_FLAGS
23     " -g -O0 -Wall "
24 )
25
26 cuda_add_executable(Test.build ${CUDA_SRCS})
27
28
```

Save Cancel

Editácia konfiguračného súboru pre preklad

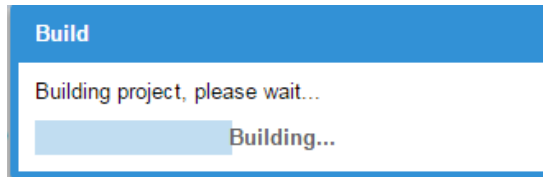
Pri vykonaní zmeny užívateľom je následne pri preklade aplikácie zobrazená možnosť ponechania tejto zmeny alebo automatickej generácie súboru na základe štruktúry projektu.



Zobrazenie možnosti ponechania zmeny konfiguračného súboru pre preklad

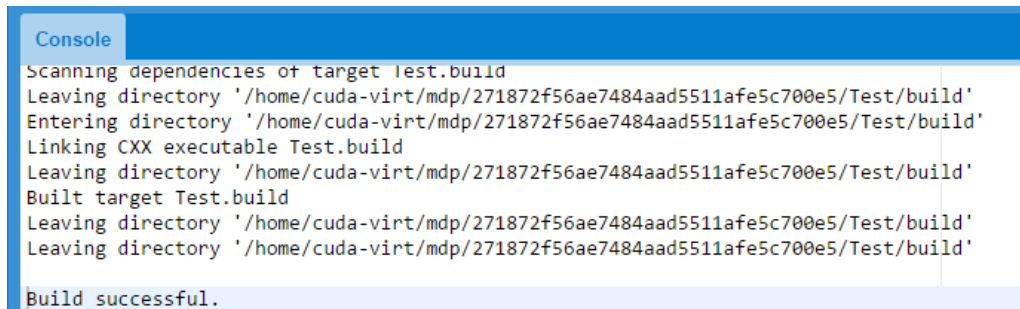
Pri potvrdení je automaticky vygenerovaný konfiguračný súbor, užívateľove zmeny sa zahodia. Pri zamietnutí zostanú ponechané užívateľove zmeny. Po obidvoch možnostiach pokračuje preklad aplikácie.

Pri preklade je zobrazené modálne okno s informáciou o prebiehajúcom preklade.



Prebiehajúci preklad

Po preklade aplikácie je výstup zobrazený v konzole.

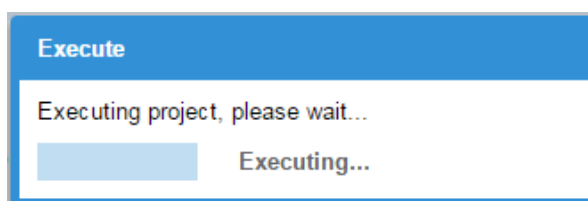


Úspešný preklad projektu



Neúspešný preklad projektu

Po úspešnom preklade je dovolené vykonanie spustenia aplikácie. Pri získavaní výstupu zo spustenia je zobrazené modálne okno s informáciou o aktuálnom vykonávaní projektu.



Vykonávanie behu aplikácie

Po vykonaní aplikácie je výstup zobrazený v konzole.

```
Console
----- RUNNING -----

--- General Information for device 0 ---
Name: Tesla K40c
Compute capability: 3.5
Clock rate: 745000
Device copy overlap: Enabled
Kernel execution timeout: Disabled

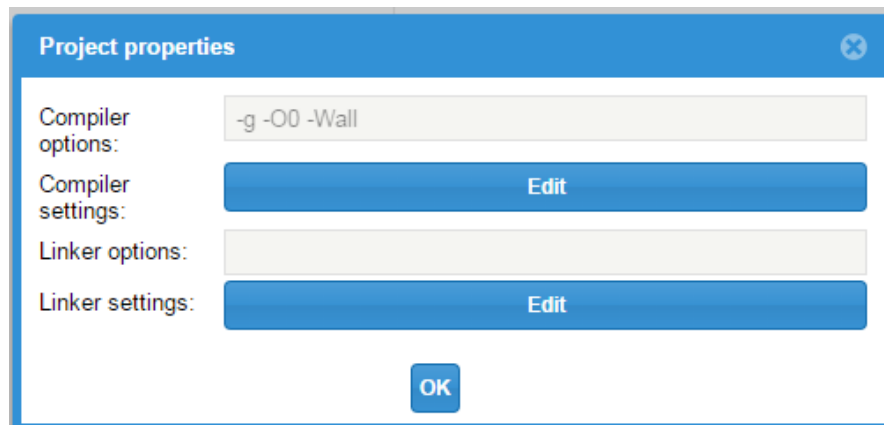
--- Memory Information for device 0 ---
Total global memory: 12079136768
Total constant memory: 65536
Max memory pitch: 2147483647
Texture alignment: 512

--- MP Information for device 0 ---
Multiprocessor count: 15
Shared memory per multiprocessor: 49152
Registers per multiprocessor: 65536
Threads in warp: 32
Max threads per block: 1024
Max thread dimensions: (1024, 1024, 64)
Max grid dimensions: (2147483647, 65535, 65535)
```

Výstup aplikácie po jej spustení

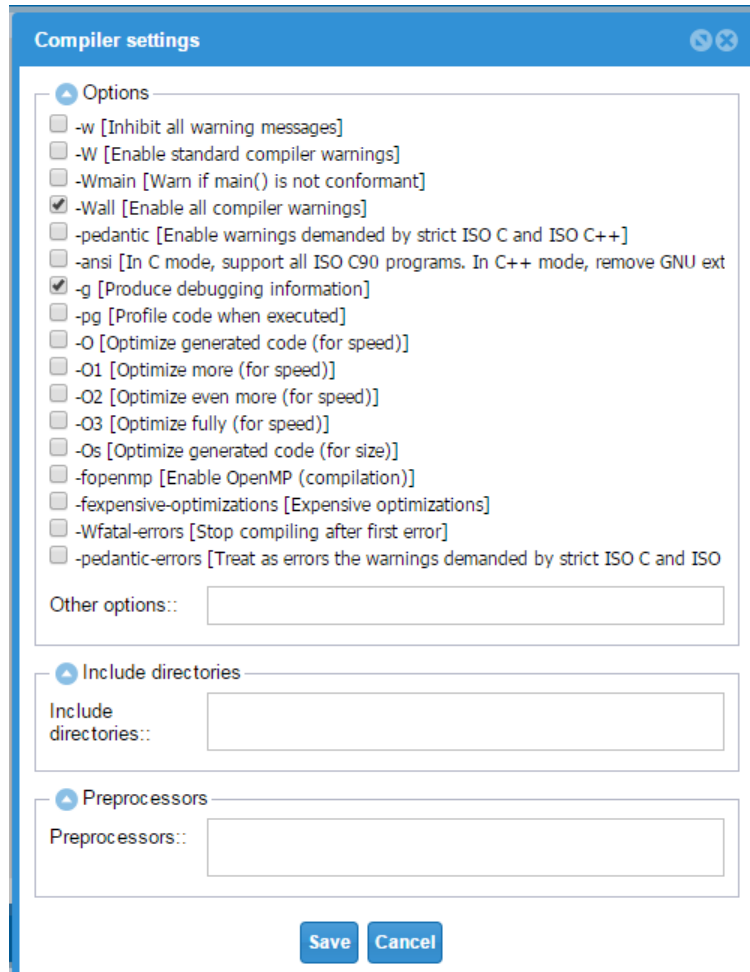
Nastavenie vlastností projektu

Nastavenie vlastností aktívneho projektu je vyvolané po kliknutí na Properties.

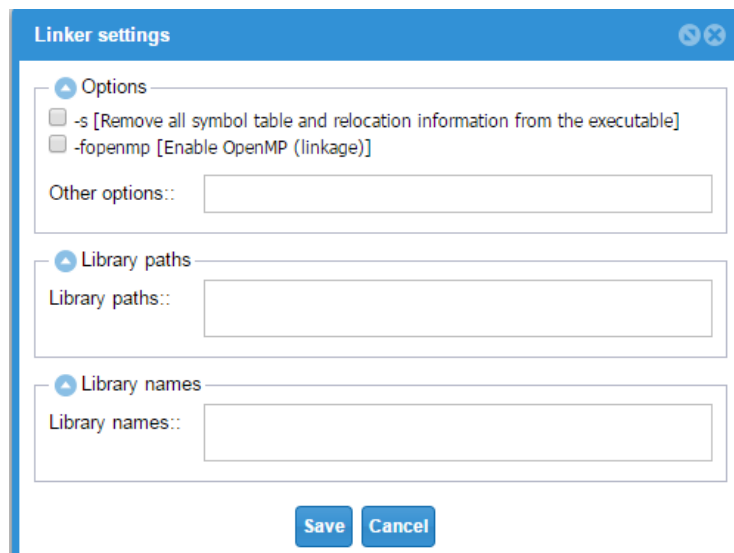


Nastavenie vlastností projektu

Toto nastavenie je rozdelené na dve časti, nastavenie kompilátora a linkera. Každá z týchto možností poskytuje špecifické vlastnosti pre nastavenie.



Nastavenie vlastností kompilátora



Nastavenie vlastností linkera

PRÍLOHA P II: POPIS PRILOŽENÝCH SÚBOROV NA DISKU CD

Text diplomovej práce:	<i>fulltext.pdf</i>
Zdrojové kódy aplikácie:	<i>Prílohy/Zdrojové kódy</i>
Používateľská dokumentácia:	<i>Prílohy/CudaIDE_manual.pdf</i>
Programová dokumentácia:	<i>Prílohy/Dokumentácia</i>
Web archív:	<i>Prílohy/CudaOnlineIDE.war</i>