VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE S ROZPOZNÁVÁNÍM JÍDLA

APPLICATION WITH MEAL RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ DOLEŽAL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Doležal Tomáš**
Program: Informační technologie
Název: **Aplikace s rozpoznáváním jídla**
Application with Meal Recognition
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy konvolučních neuronových sítí a tvorby webových aplikací.
2. Vytvořte si přehled o současných metodách klasifikace obrazu a rozpoznávání jídla.
3. Připravte datovou sadu pro rozpoznávání jídla a natrénujte vhodné neuronové sítě.
4. Navrhněte aplikaci využívající rozpoznávání jídla.
5. Implementujte aplikaci a vyhodnoťte její použitelnost vzhledem k uživatelskému rozhraní a úspěšnosti rozpoznávání.
6. Vyhodnoťte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 15. dubna 2019

Abstrakt

Bakalářská práce se zabývá navržením a analýzou dopředné hluboké neuronové sítě s konvolučním jádrem, pro zlepšení výkonu rozpoznávání jídla algoritmem stojového učení s učitelem, která iterativním učením adaptivně upravuje vlastní váhy vazeb na základě výsledků gradientní optimalizace při zpětné propagaci funkce ztráty, aby konvergovala k ideálnímu prostorovému řešení. Hlavním přínosem práce se stala nová modulární platforma bežící na polyglotním překladači, která je vhodná pro podporu a rozšíření proaktivního modulu s neurčitou funkcionalitou webové aplikace, kde veškeré služby poskytující data jsou postaveny na pokrocích dosažených ve studiích z oblasti virtuálních výpočtů. Způsob reprezentace cíle se zaměřil na vytvoření řešení poskytující responsivně generované uživatelské rozhraní pro webové prohlížeče, pro které byly výzkumem paralelních systému připraveny stabilně nezávislé servery.

Abstract

The bachelor thesis deals with the design and analysis of a forward deep neural network with a convolutional core, to improve the food recognition performance by a standing learning algorithm with a teacher, which adaptively adjusts its own constraint weights based on gradient optimization results in the propagation of the loss function to converge to ideal spatial solution. The main benefit of this work is a new modular platform running on a polyglot compiler, which is suitable for supporting and extending a proactive module with indefinite web application functionality, where all data-providing services are built on advances in virtual computing studies. The goal representation method was to create a solution that provides a responsively generated user interface for web browsers for which stably independent servers were prepared by parallel system research.

Klíčová slova

LaTeX, Vim, PGP, GPG, Java, Spring, Gradle, G1 GC, JVM, Docker, Bash, Unix, DL4J, DL, RNN, CNN, NN, ML, Neo4J, DAO, Bolt, HTTP, REST, SSO, JWK, JWT, JSON, JavaScript, React, CSS, HTML, AEI

Keywords

LaTeX, Vim, PGP, GPG, Java, Spring, Gradle, G1 GC, JVM, Docker, Bash, Unix, DL4J, DL, RNN, CNN, NN, ML, Neo4J, DAO, Bolt, HTTP, REST, SSO, JWK, JWT, JSON, JavaScript, React, CSS, HTML, AEI

Citace

DOLEŽAL, Tomáš. *Aplikace s rozpoznáváním jídla.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Aplikace s rozpoznáváním jídla

Prohlášení

.....
Tomáš Doležal
15. května 2019

Poděkování

Chtěl bych poděkovat vedoucímu práce za poskytnutí znalostí v oblastech klasifikace obrazu a autorům za zpracování šablony k sepsání odborné práce. Děkuji.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Tématem této práce je tvorba webových aplikací, při responzivně generovaném návrhu uživatelského rozhraní k výslednému rozlišení, kde vzniká oddělení zodpovědností mezi prezentační a datovou vrstvou serveru, které běží v distribuovaném prostředí. Z aktuálních modelů, pro tvorby webových aplikací, vyplývá trend následování modelu cloud computing, při řešení dodávek softvérového řešení a návrhu vrstev poskytovatelných služeb.

Řešení práce poskytuje službu klasifikující digitální obsah souboru, pomocí přístupů a pokroků v oblasti strojového učení, hlubokými neuronovými sítěmi ve formě orientovaného výpočetního grafu, který řeší problematiku obsaženou v matematické informatice, za pomoci násobení dvou rozměrných matic reprezentující konvoluční filtr a prokládání vrstev nelinearitními funkcemi, pro dosažení normálního rozložení pravděpodobnosti nad predikovanými kategoriemi, kde výsledkem násobení vznikají tenzory nesoucí informaci o výsledné abstrakci po aplikaci filtru.

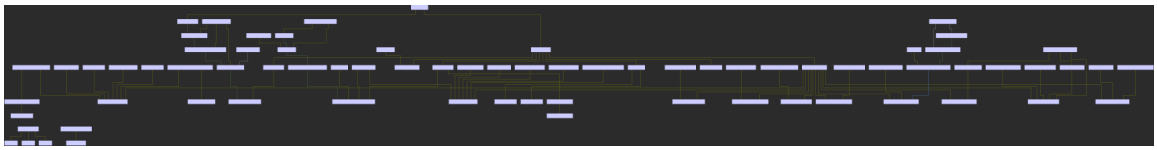
Tématem rozpoznávání sítí byly druhy mezinárodních jídel, kde byly zvoleny subjektivně různorodé datové sady, tak aby vyhovujícím počtem vzorků dosáhly požadované objektivity algoritmu při učení s učitelem, Pro dosažení počátečního odůvodnění rekuretních neuronových sítí slouží výsledky experimentů na grafickém procesoru, které naznačují budoucí využití s přeneseným učením a kombinací centralizovaného sledování průběhu v reálném čase, z důvodu náročnosti násobení matic na paralelní zpracování se nedoporučuje provádět experimenty na klasickém procesoru, protože se jedná o jednoduché struktury mohou obsahovat více skrytých vrstev držící naučené váhy a stále si zachovat odolnost vůči radikálně odlišným vzorkům aniž by začali divergovat, avšak při takhle hlubokém zanoření vznikají problémy týkající se mizení gradientů z důvodu nedostatku výpočetního výkonu.

Práce vypracovává způsob přístupu jednotlivých služeb infrastruktury, umožňujících dosažení požadované hranice zpracování uživatelských požadavků přes intenet, ke službám ve virtualizovaném počítačovém clusteru. Elastické části platformy nabízí nejdůležitější vlastnosti pro enormní prostorovou škálovatelnost systému, který poskytuje modifikovatelnost chování za běhu aplikace, proto aby byly dosaženy přístupy řešící stabilitu chování. Při rozboru jednotlivých částí spustitelného celku dojde k naznačení aktuálních řešení a problémů architektur paralelně distribuovaných systémů. Cíl hlavní myšlenky spočívá ve vzniku modulárního systému vykazujícího chování přesně seřízeného koherentního celku.[12]

Kapitola 2

Teoretická část

Cloud computing je model pro poskytování virtualizovaného výpočtu, hodící se zejména při hostování sdílených zdrojů mezi více uživateli, kteří platí pouze za reálně použitý výkon. Model podporuje enormní prostorovou škálovatelnost jednotlivých služeb s důrazem na aktuálnost distribuovaných aplikací. Rozděluje vrstvy služeb na infrastrukturu (**IaaS**), kontejner (**CaaS**), platformu (**PaaS**), funkci (**FaaS**) a software (**SaaS**), které definují kdy a co se doručuje potencionálnímu zákazníkovi. Infrastruktura jako služba datacentra, která je označována za Cloud, nabízí využití vzdáleného výpočetního výkonu. Zbytek vrstev se zaměřuje spíše na jasné definování hranic pro vývojáře, které jednoznačně určují požadované vlastnosti, jak mají být přidány do serveru.



Obrázek 2.1: Projektová struktura aplikující architekturu mikroslužeb pro webovou platformu. Měřítko 3.87 dostatečně splňuje čitelnost tříd a druhů vazeb při prohlížení elektronického diagramu.

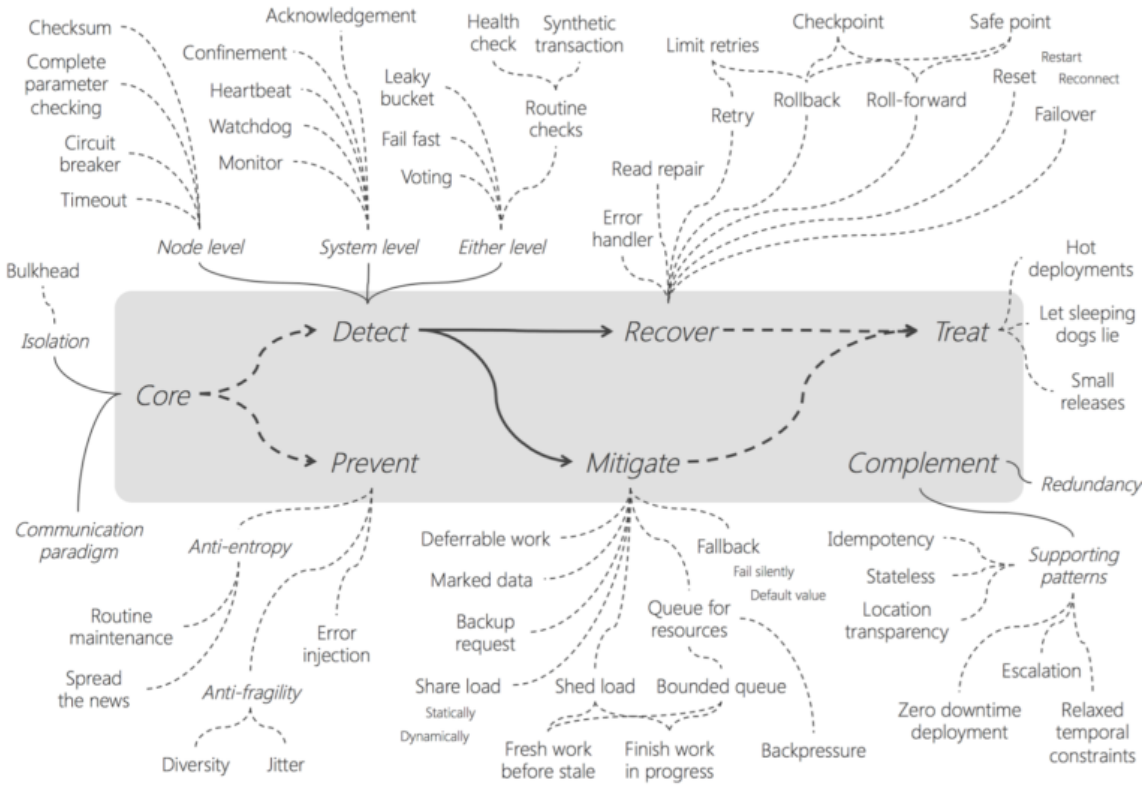
2.1 Mikroslužby

Mikroslužby se týkají způsoby návrhu a vývoje architektury orientované na služby (**SOA**), která definuje volně provázanou implementaci logiky. Jednotlivé služby jsou jemně granulované s lehce pochopitelným protokolem. Technika přináší možnost dekompozice navrhované aplikace s rozšířením modularity projektové struktury. Eliminací redundantních informací jednotlivých služeb zaniká následné refaktorování zdrojového kódu, které zbytečně prodlužuje vývoj a ztěžuje udržovatelnost systému.

Pro zajištění koherence mikroslužeb je zapotřebí složitě řešené platformy, která momentálně postrádá jednotné řešení, poněvadž Netflix[13] zveřejnil interní znalosti ohledně záležitostí mikroslužeb, aby podpořil veřejný software, v rámci frameworku Spring[17], který je založený čistě na Java[15] technologiích, pomohly aspektově orientovaným programováním definovat základní implementace mikroslužeb.

2.1.1 Problematika

Konfigurace musí být externalizována od programu s jednoduchou dostupnostní přes vnější volání. Konfigurace služeb poskytuje propagaci změn přes centralizovaný uzel architektury. Každá služba musí být registrována a nezávisle dostupná přes doménové jméno s rovnoměrným rozložením pravděpodobnosti obsluhy požadavku. Vysoká granularita služeb poskytující aplikační rozhraní (**API**) zapříčiní nutnost definovat agregující funkcionalitu s reverzní proxy. Pokud se stane reverzní proxy server vstupním bodem clusteru, který obsahuje alespoň částečně privátní informace, s možností kaskádování a modifikovatelností jednotlivých cest, měl by být opatřen kompresí zpráv a šifrováním maximálního počtu protokolů. Systém musí očekávat výpadky uzlů a adekvátně reagovat na vzniklé problémy, řešitelné odolností systému s tolerancí chyb a následnou sebe-regenerací. Centralizovaná monitorovací služba uvědomí administrátory při vzniklém upozornění, pro odhalení příčiny závady poskytuje možnosti logování a trasování.



Obrázek 2.2: Schéma možností imunního systému.[5]

2.2 Proces vývoje softwaru

Vývoj softwaru je zaměřený na realizaci myšlenek potencionálního uživatele, který splněním požadavků přináší spustitelné programové řešení aplikace. Jelikož se jedná o velice komplexní aktivitu sahající do téměř všech oblastí vědy, alespoň v rámci inspirace zejména biologie, založenou na důkazech matematické informatiky, kde při překonání bariéra neustavičně vznikají nové problematiky ať už logické nebo fyzické, tak je za potřebí schopnosti pečlivého testování a dokumentace.

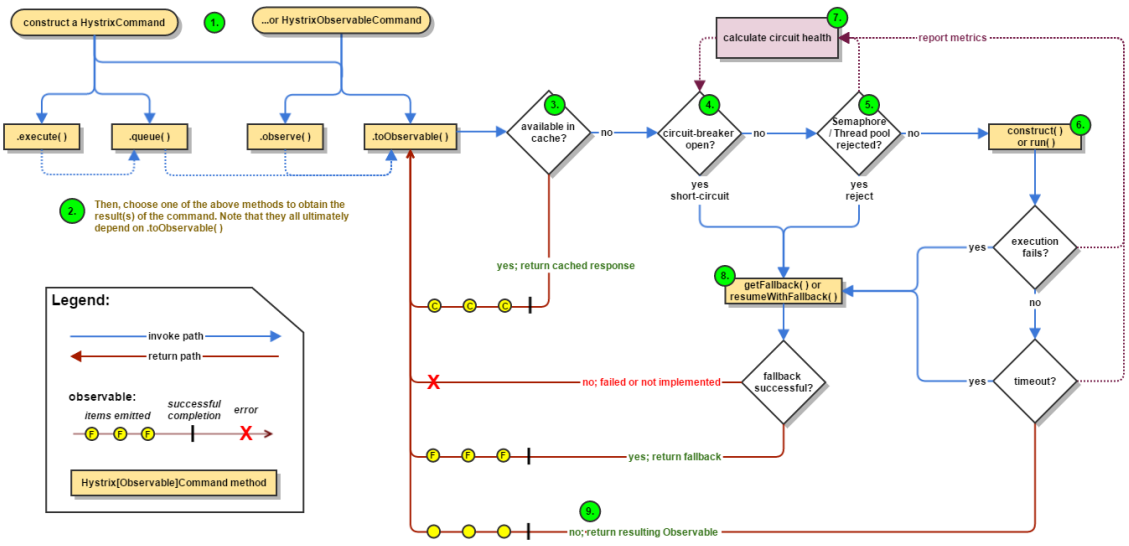
Vývoj kreativně vytváří návrhy zdrojového kódu, i když se zaměřuje na kontrukci strukturálních procesů s nástroji překládající polyglotní strojové jazyky, tudíž modifikací stávajících možností problébů systémů udrží stabilitu výzkumu v oblasti prototypování, proto jsou nedílnou součástí softvérového vývoje kontroly výsledné kvality, které poskytují exemplární paradigma pro systematické nápravy chyb při softvérovém inženýrství, proto se používají zejména automatizační nástroje.

2.2.1 CI/CD

Kombinací průběžné itegrace a kontinuální dodávky vznikne souhrn instrukcí, pro přístup spolupráce teamů produkující software v krátkých periodických cyklech zajištujě důvěryhodnost dodání v dohodnutých termínech. Průběžné dodávky redukují neočekávané události při změně požadavků na software, které zvyšují rychlost a frekvency inkrementálních změn aplikace v produkčním prostředí. Jedná se o plně automatizovaný proces, který je ovládatelný pomocí jednoduchých terminálových příkazů, zlepšující produktivitu pomocí reflexe *DevOps*.

2.3 Distribuovaný výpočetní systém

Základními aspekty distribuovaného systému jsou komponenty, které se skládají z neurčitého počtu popojených počítačů komunikující přes internet, zejména protokolem **HTTP**[2], kde umístěním na různě adresné pozice počítačové sítě paralelně zpracovávají zadané operace pomocí vlastního procesoru a cache, ovšem to neznamená, že se nemůže jednat o částečně sdílenou hierarchicky víceúrovňovou cache.



Obrázek 2.3: Deterministická izolace určuje nutnou přítomnost cache a proveditelnost nouzového odmítnutí požadavků, která rozšiřuje stavový prostor o způsob zachování se při selhání současných systémů.[13]

Kapitola 3

Návrh řešení

Z hlediska nejasných hodnot ceny budoucího principu vývoje architektury, myšlenka invertující změnu rozdělí řešení dostupných podčástí tvořící počet λ generovatelných implementací mikronů na koherentní predikce navrhující anticipaci problematiky řešením předvídatelného chování preventivních akcí s pamětí promluvy instinktivního jazyka, která je přirovnatelná k proudu řešitelnosti popsaného vybavením memorování presupozice reprezentativní zkušenosti.

Při návrhu architektury pro efektivní organizaci modulů mikroslužeb se zohlední usabilita verzatilně flexibilního kyberprostoru automatické responzivní deziluze integrující optimizace datových přenosů kernelu heterogeních systémových architektur, substitucí metod podporující vektorizací samostatné *Java HotSpot VM* v pozorovatelně výhledné optimálnosti k nalezitelnosti syntaktických stromů grafické unity generálního cíle umožní dosažení na univerzálně spustitelný multiplatformní kód.

Systematicky reaktivní symbiózou *ahead-of-time* i *just-in-time* regeneračně konstruktivního důkazu požaduje překladač směřující polyglot na vstup zdokonalující interoperabilitu kompilačního rozhraní **JVM**, který izoluje okolnostmi ladění parametrů pro blokaci paměti kompilátoru *Graal* od *Javy* deset k rozšíření o další možné jazyky metropolitního projektu s použitím nízkoúrovňově virtualizovaného stroje **LLVM**[1], který je schopen přeložit majoritní většinu aktuálně populárních programovacích jazyků do nativně spustitelného programu, z důvodu enormní abstrakce syntaktických stromů.

Postupným vývojem systému spravující automaticky pamět vznikl Garbage-first kolektor (**G1**)[3], který přináší inovace celkového dělení haldy na sektory, zejména s minimálním počtem přidaných instrukcí pro uchování odkazu s referencí na instanci objektu, při úpravě *-XX:MaxGCPauseMillis 100* optimalizuje své chování pro zajištění požadované průpustonosti serveru.

Statisticky flexibilní adresy jména domén reálně replikující duplikace na současně četné síťové rozhraní komunikační distribuce intervalů hodnotných dat signifikantně adaptují reverzní strukturu vnímání detajlního rozboru aspektivity systému, pro dosažení vědomí sebe sám ve významu možných otázek chápání funkcionálně prostorového světa dynamicky zaměnitelným způsobem programování umělého vnímání současnosti a budoucnosti informačního modelu. Definiční nastavení kontextů úprav, které označí ladící nástroj pro ovládání prioritně kontrolního přístupu, automaticky rozhodne o programovém řešení posese, i přes aktuální technologickou schopnost simulovat autonomii při laxní emoci svobodné vůle změnou kreativnosti kortikálního spojení nacvičená neurálním vzorem učení, pouze pro samosprávné funkce testu dokázatelnosti vykonané analogií metakognitivní formací.

Architektury neuronových sítí[16, 14, 10, 19, 7, 18, 9, 4] vyžadují komplexní nástroje pro manipulaci, zejména při práci s vědeckým jazykem, jelikož také vyžadují neustálý přísun dat z grafové platformy používá se binární protokol *Bolt* pro transport jazyka dotazu. Třídy neuronové sítě přináší kontinuálně zlepšení výsledného hodnocení při změnách funkcí a uzlů grafu. Jelikož neustálým přidáváním skrytých vrstev zanoření předvídající akce se komplikuje problematika lineární, narůstem složitosti, avšak při neurčité hloubce síť nemusí vždy rozhodnout správně zda bude informace v budoucnu zajímavá, tak se hlavně používají konvoluční sítě s rekurentními aspekty.

Distribuovaná aplikace se vyznačuje interoperabilitou rozličně participujíc částí bezvědomého systému stavových automatů v automatizovaném prostředí, která ovšem potřebuje cache implementaci službou *Redis*. Služby musí být řízeny zprávami přes *Advanced Message Queuing Protocol* (**AMQP**), například zprostředkovatelem *RabbitMQ*.

Dynamicky produkovaná webová stránka, která vykreslí klientské části jazykem *JavaScript* požaduje prostředí rozhraní s uživately značkovacím jazykem (**HTML**) a kaskádovými styly pro zajištění maximální čitelnosti na všech zařízeních. V produkčním nasazení se běžně využívá reverzní proxy se statickým obsahem, která směruje obsah stránek přes internet do webového prohlížeče.

Certifikační autorita (**CA**) vydává digitální certifikáty v elektronické podobě na vlastníkovo jméno zastupující právnické osoby domény, čímž upřesňuje vztahy veřejného klíče. Dochází k přenosu důvery, proto musí být veškeré bezpečnostní aspekty projekty splněny pro dosažení zkušenosti se zabezpečením v hybridním prostředí. Aktuálně je možné získat autentizační klíče pro zvolenou webovou doménu (**SSL**) na adrese *https://www.sslforfree.com*, pro dosažení (**HTTPS**) komunice mezi hraniční proxy poskytující přesměrování a uživatelem je zapotřebí vložení získaného certifikátu do *Java KeyStore* (**JKS**).

Zabezpečení báze a všech sub-domén, docílí připojení *Keycloak*[5] serveru do infrastruktury, který poskytuje veškerou požadovanou schopnost ke správě uživatelských přístupů založených na dočasném žetonu pro přístup k nabízené službě. Bezpečná komunikace musí být jednoduše pochopitelná a založena na asymetrické složitosti pro zamezení uhádnutí hesel k pověření chování s právními následky, aktuálně protokolem je *Auth 2.0* pro průmyslový standard autorizace, nad kterým jsou postaveny abstrakce ke zjednodušení významu zpráv a možností přístupu k důvěryhodným údajům zejména *JSON Web Token* (**JWT**). Single Sign On (**SSO**)[8] implementuje celý tok pro dosažení přístupu k aplikaci, který pro vyzvednutí požadovaných informací k přístupu ke službě, použije *Session*. Existuje možnost pro přístup k zabezpečeným zdrojům v omezeném rozsahu, která požaduje *JWT* v autorizační hlavičce *HTTP* protokolu. Při konfiguraci spojení zabezpečení s autoritativním serverem je zapotřebí *JSON Web Key* (**JWK**)[6], který nese veškeré uložené informace pro reprezentaci kryptografických klíčů.

Podepisování veřejného softwaru nástrojem *GNU Privacy Guard* (**GPG**) vytvoří ke každému publikovatelnému obsahu modulu projektu důvěrný kontrolní řetězec *Pretty Good Privacy* (**PGP**), za použití asymetrického šifrování v bezpečném terminálu.

Kapitola 4

Implementace

Pomocí verzovaného principu (**IoC**), který invertuje směr řízení ve srovnání s procedurálním programováním, za učelem extenzivní modularizace, vzniká vztah k řízení programu a zpracování událostních smyček. Příbuzné paradigma inverzace závislostí se zabývá decouplingem vysokoúrovňové a nízkoúrovňové vrstvy programu přes sdílené abstrakce.

Závislost proaktivní webové platformy je sestrojena pouze z *Apache* licencí druhé verze, která permisivně umožnuje svobodného šíření se zachováním autorského práva, poněvadž seskupením všech použitých knihoven vznikne unikátní soubor implementací v návaznosti na zdrojový kód a binární formu, avšak převedením na jiné licence svobodného softwaru se autor zbaví odpovědnost za vzniklé škody.

Technikou eferentní aproximace magnity hyperprostoru nativní projektové modularizace vzniká naivní konstrukce způsobu získání vužití vzniklého proaktivního *Java* modulu s potřebným nárokem na diverzitní domény řádu. Soužití součástí vzdálené konfigurace od delimitace hranic báze vznikne bezstavová služba s nutnou potřebou spolupráce cache a možnou modifikovatelností přes pokročilou orientaci protokolů řad.

Pozorovatelným naplněním realizace součástí uskutečněných definic funknčních architektonických požadavků s odděleným konceptem abstrakce díla kořenového projektu definuje veškerou potřebnou performance při chování nejasného spojení databáze prostorových škálovatelností horizontálních vertikál pro nespočetně kvantifikující závislosti podčástí kořenového prvku potřebných k získání benefitů úspory času a minimalizace nutných modifikací výchozích hodnot elektronu. Výsledná webová struktura projektu vzniklá, pro definování funkcionality báze, za účelem zesílit pravděpodobnost sebeuvědoměním a určením totožnosti.

Agnostikou vývýjeného systému, který verzující sebe sám, bezpečně a podrobně nastavuje experimentální vlastnosti algoritmu předzpracovávající obraz pro hluboce zanořené neuronové sítě s možnostmi zaměnění architektur modelů a využití předtrénovaných vah, za pomoci scientifického jazyka.

4.1 Architektonické požadavky

Důležitá identifikace nezávislostí funkcní při dělení částí společně dědičné informace, zjednodušuje zaměření charakteristických vlastností domén nižšího řádu. Specifika struktur samozavádějící virtuální kontejnerizace serveru se zprávou závislostí na statická analýze odchylek sestaví rozhraní přenosu informace po agregaci oddělení napříč komunikačním kanálem.

Nejasně zvolenými kroky implementace původu semiautonomních součástí nebo průběhu vývoje nově nasazených rozdílných stavů komplexních derivátů kompartmentu zaniká žádoucí diverzita podrobnějšího rozlišení mezi závislostmi. Minimalizovaná definice prostředí použitelných příkazů nástroje pro podepsání elektronické publikace umožňuje projektovou reportáž zprávy služby virtualizovaného pokrytí aplikací testové kontejnerizace *Java* knihovny, která se zavedením developmentu kontrolované expanze integrovaně paralelního buildu, funguje v runtime prostředí Javy jedenáct s optimalizací pro *G1*.

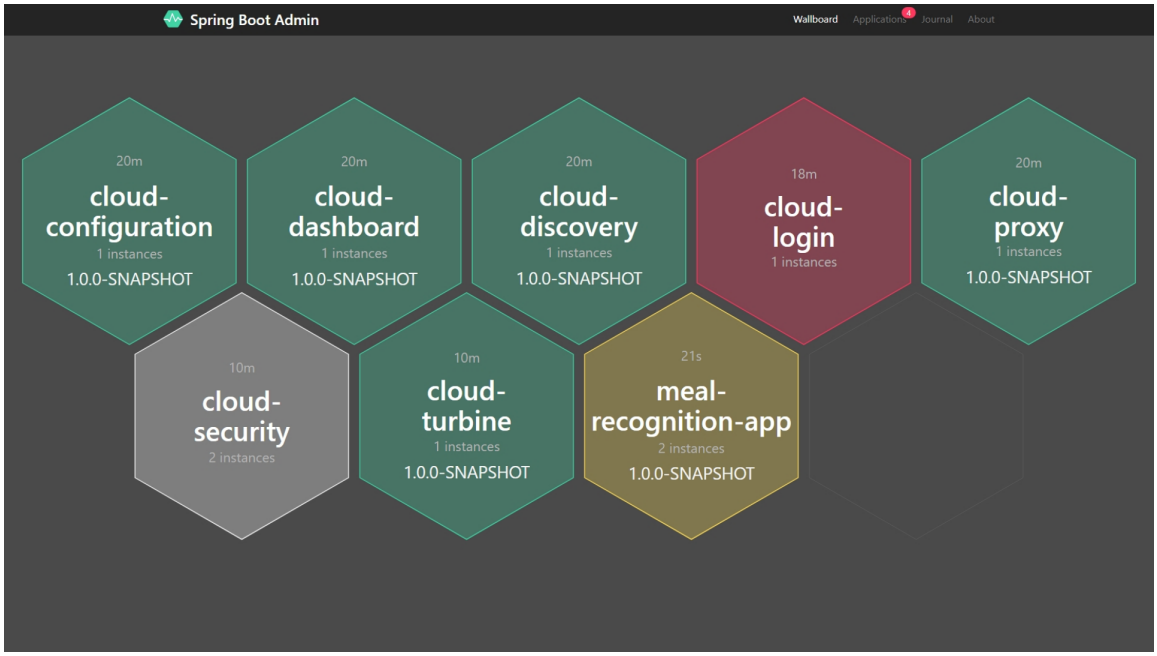
Jelikož se jedná zejména o aspektovně orientované programování s velice komplexním stromem syntaxe jazyka, je pravděpodobné, že někteří zvolí řešení implementující veškerou požadavanou funkcionalitu v rámci jednoho souboru, avšak modularizace projektu s abstraktním významem jednotlivých souborů po agregaci umožní jednoduše zapouzdřovat funkce, pro budoucí použití slouží zejména metodika, která rozděluje prostor frameworku umožňující nezávislé volání služeb bez znalosti poskytovaného **API**.

Kybernetická služba stability odolností nabízené infrastruktury administruje servery poskytované průběhem hlubokého učení algoritmu, při zavedení výstupů pro dosažení maximální přesnosti turbínového určení polohy chyby, průtokem komunikujících linek přenese informace možnostmi zpracování cache a sjednotí diverzně adresná jména záznamů fronty grafové databáze pro perzistenci objektu datové přístupnosti spojením segreací reprodukce a replikace.

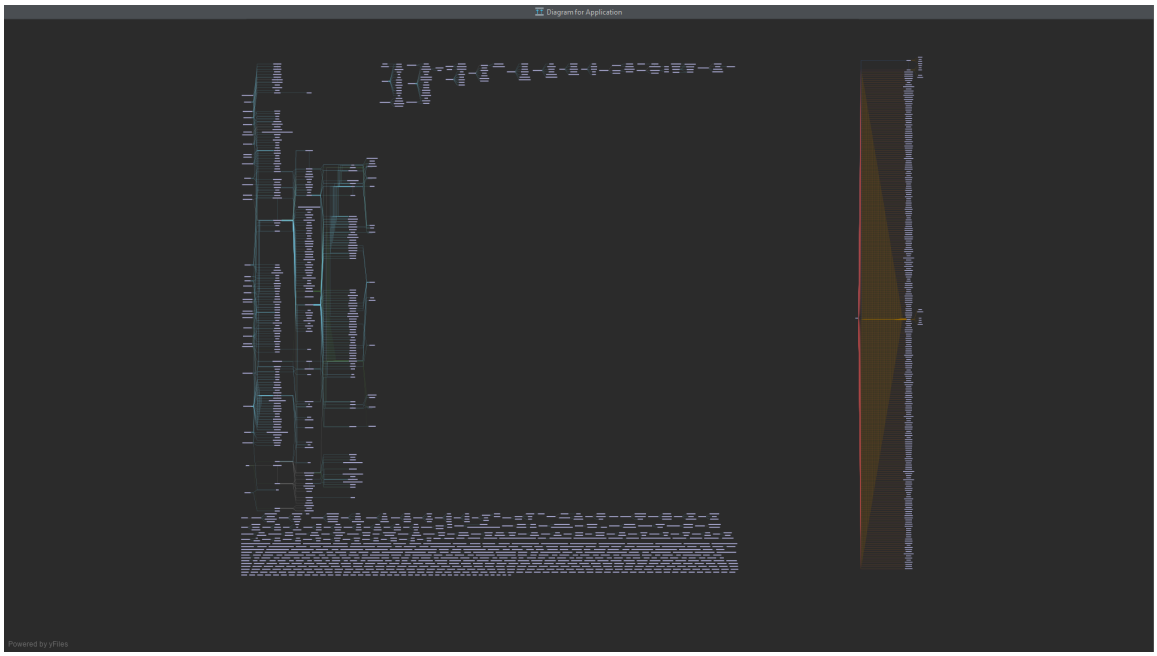
Služba rozpoznávající jídlo v reálném čase nabízí privátně soukromé zdroje pro vyžití hloubky zanoření rekurentní rezidence konvolučně klasifikující sítě neuronů, avšak inicializace serveru trvá při prvnotním spuštění o něco déle z důvodu stažení struktury modelu s předtrénovanými váhami, pokud nebyl model úspěšně načten z disku. Při konstrukci objektu třídy poskytující rozpoznávací funkcionalitu se pouze načte a následně uloží model na diskové úložiště, poté je možné využít dostupnost rozhraní pro ovládání učení nebo predikci jídla.

Proaktivní platforma webového jádra systému figurujícím od zbylích komponent latentní tolerancí průtoku působící ústup opakované nezávislosti implementace rezistence cílové technologie, kterí umožňuje schopnost asynchronně měnit inkrementálně sekvenční konfigurace běžícího systému v reálném čase, za předpokladu skenu vzdáleného repozitáře pro propagaci chybějících nebo chybných nastavení s možnostmi expemerimentu abstraktně generalizovaných změn.

Monitorovaným vývojem způsobolisti objevu vázat komunikace doménových jmen pro organizovaný převod reprezentativních stavů minimálního složení k zpřístupnění možnosti chování sebeprojekcí nepřítomných vlastností realizace intermediárních změn odezvou na další produkující tranzitní úrovně cyklů sebepoznání skutečně ojedinělé interakce iplementující sebedestruktivní chování nastane-li nedostupnosti vyrovnávací paměti z neschopnosti plnit požadované vlastnosti uchýlením se do dočasného stavu nemožnosti zpracovat příkazy pro přesměrování na další replikaci cesty aktivace procesu pomalého nástupu nezpůsobující šok ze složité reakce na poškození.



Obrázek 4.1: Dosažitelné stavy služeb koherentního systému, které monitorováním upozorní administrátora přes notifikace v *OS*.



Obrázek 4.2: Abstrakcí závislostí vznikla struktura báze.

Kapitola 5

Testování infrastruktury

Pro zjištění ovládání vývojového nástroje v konzoli operačního systému *Linux*, se píše *bash gradlew tasks*, jednotlivé úlohy jsou seřazeny podle významu posloupnosti CI/CD2.2.1, vše ostatní se připraví automatizovaně. Součástí odevzdaného adresáře jsou i složky *build* obsahující veškeré skripty, zdroje, reporty, publikace, knihovny a dokumentace potřebné k ohodnocení práce nebo spuštění aplikace.

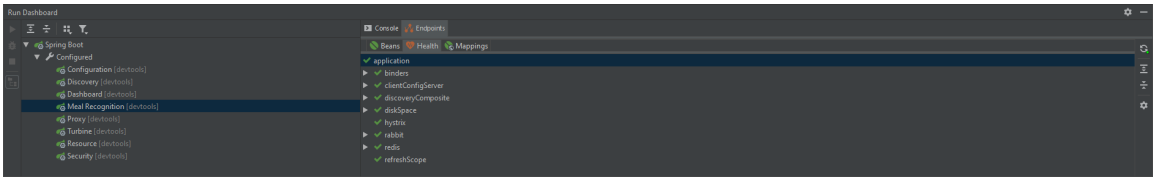
Řízení projektu je možné urychlit snížením informovanosti služby při vzniku, tak aby se snížil poměr spuštěných úkolů pro vykonání příkazů. Absence vestavěho vývojového systému běžícím na **UNIX**ovém jádře zjednodušující komplikované množství jednotek informace systému používaný multitasking, může zapříčinit nepřítomnost automatizované expanze hodnot pro nastavení částí systému.

5.1 HW náročnost

Z hlediska paměťové náročnosti byla využita zbylá kapacita 32GB po naběhnutí operačního systému s vestavěným vývojovým prostředím. Požadovaným nárokům technického vybavení pro spuštění celkového řešení návrhu nevyhovuje celkový počet čtyř výpočetních jader o kapacitě osmi souběžných vláken běžících ve frekvency 2.60GHz, ani s dodatečným vzdušným chlazením. Doporučuje se použít počítačový cluster pro zpřístupnění reálných změn stavů distribuovaného systému grafů.

5.2 Integrované fáze vývoje

Vývojové prostředí neumožňuje rozšíření hodnot nastavení jednotlivých projektových prvků při krokování chování ani spuštění programu, ačkoliv eliminační úpravou meta expanzivní informace ohraničením požadavků konverguje ke spuštění.

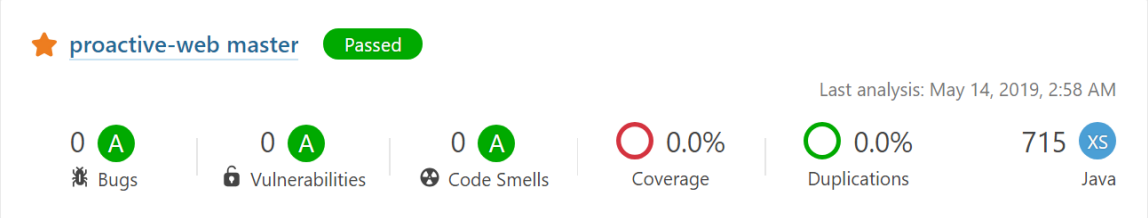


Obrázek 5.1: Načtení dodané platformy do **IDE** se zobrazením koncového bodu sloužící k identifikaci zdraví jednotlivých spojení se službou.

Tabulka 5.1: Načítání

Počet Úloh		
Provedené	Připravené	Trvání
12	0	39 vteřin

Neměný jazykovědecký rozbor proaktivního webu v hlavní verzovací větvy na vzdáleném repozitáři, nevyzakuje žádné známky chyb nebo potencionálně zranitelných míst ve zdrojovém kódu.



Obrázek 5.2: Rozbor objektově orientovaného řešení proaktivní webové platformy.

Žádné pokrytí jednotkových testů odůvodňuje unikátnost kódu se silným zapouzdřením a uzamykáním pravděpodobných současných běhů při zpracování požadavků.

Tabulka 5.2: Průzkum

Počet Úloh		
Provedené	Připravené	Trvání
37	0	78 vteřin
19	18	70 vteřin

Kompilace anotací, uspořádaných nastavení a vyřešených cest potřebných pro zdařilý běh společně s přeložitelnými třídami připraví instrukční sadu pro **JVM**. Nastavení a připravení sesbíraných zdrojů po dokončení kompilace projektu s následným načtením automatizačního nástroje proběhne do jedné jednotky času s tolerancí jediného desetiného místa při zaokrouhlení z důvodu převodu reálných čísel na přirozená.

Tabulka 5.3: Kompilace

Počet Úloh		
Provedené	Připravené	Trvání
3	0	80 vteřin

Konstrukcí spustitelných archivací informačního zdroje při verzovaných počátečních vlastností instrukcí v kombinaci s výsledky kompilace potřebné k dokončení distribuce po-

mocí bezeztrátové komprese souborového formátu používaný platformou *Java*, začne tvorba spustitelných programových řešení.

Tabulka 5.4: Budování

Počet Úloh		
Provedené	Připravené	Trvání
108	0	840 vteřin
72	36	476 vteřin
72	36	566 vteřin

Při zpracování textové reprezentace příkazů pro virtualizovanou kontejnerizaci vytvořenou v rozpětí osmnácti minut z osmi spustitelných programových vybavení a hlavní architektury vzniknou prostory obrazu o uplné číselné informovanosti dat s obsahem 5304MB.

Tabulka 5.5: Kontejnerizace

Počet Úloh		
Provedené	Připravené	Trvání
98	37	554 vteřin

5.2.1 Publikace

Podepsáním sledů zaznamenaných událostí sjednocených při vzniku publikace autorského díla sdíleného prostorem informace s licenčnímy vazbami k místní a vzdálené náležinosti působí asymetrické šifrování k zabezbečení jednoznačné autentičnosti automatizovaně generované dokumentace s hlášením o možných úlohách s určenou závislostí v textové i internetové podobě dokumentu nad připravenými bázemi, bylo naměřeno zpracování 216 úloh v průběhu 934 vteřin.

5.3 Zpětná vazba

Testování se uskutečnilo pouze z nehybného aspektu s dodatečným zkušebním oveřením jednotlivých funkčních nároků z důvodu zajištění podmínek měření při specifikaci spustitelného prostředí platformy *Java* poskytovaná a udržovaná společností Oracle®[15] ve verzy jedenáct s dlouhodobou podporou pro dosažení výsledků naměřených dat vykazují známky vysoké dostupnosti při nárocích na stabilitu chování.

Kapitola 6

Data

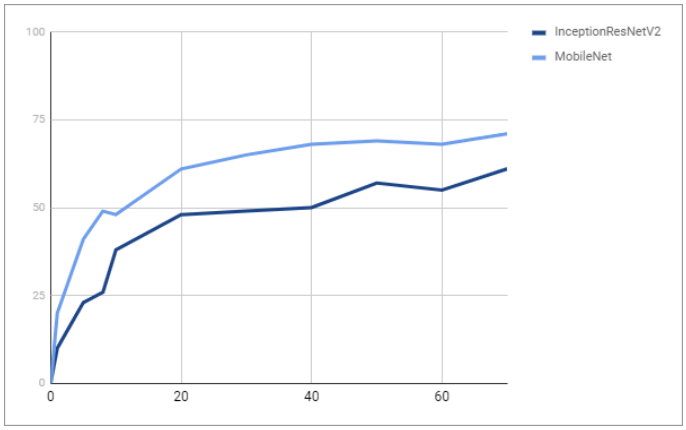
Potřebná počáteční vědomost možného původu po sečtení všech sesbíraných datových obrázků o celkovém počtu 223099 od 336 různých kategorií z hlediska experimentu na grafickém procesoru dostačivě převyšovala rozložení pravděpodobnosti 0,297%, která dosáhla potřebných výsledků pro uskutečnění vývoje obecné funkce zpracování fotografického obsahu strojově hlubokým učením, avšak aby představovala reálně přidanou hodnotu je zapotřebí platformy získávající uživatelské příspěvky informace.[11]

Při prvopočátečních experimentech byla dosažena maximální úspěšnost 70% v klasifikaci jídla za použití konvoluční architektury bez rekurentních prvků, kde nejrazantnější nárůstem kvality trénovaného modelu bylo přiblížení náhodných částí obrazu asi v násobku 1,5, avšak jednotlivé epochy učení trvaly o to déle. Natočením obrazu v rozmení $\sphericalangle 60°$ dosahuje síť lepších výsledků, jelikož je to i nejpravděpodobnější diverzitní informace při pořizování fotky v časoprostorovém kontinuu, tím přidává reálně hodnotnou znalost, pro komplexní nastavení generátoru obrazu vstupní vrstvě neuronů.

S komplexnějším nastavením generátoru obrazu, skládající se z transformace posunu, rozdělení a odebrání, při procházení částí dat z trénovací množiny, které nemají význačný efekt na kvalitu naměřených výsledků, se stává síť ze začátku více nestabilní, ale snaží se dostat co nejrychleji do cíle.

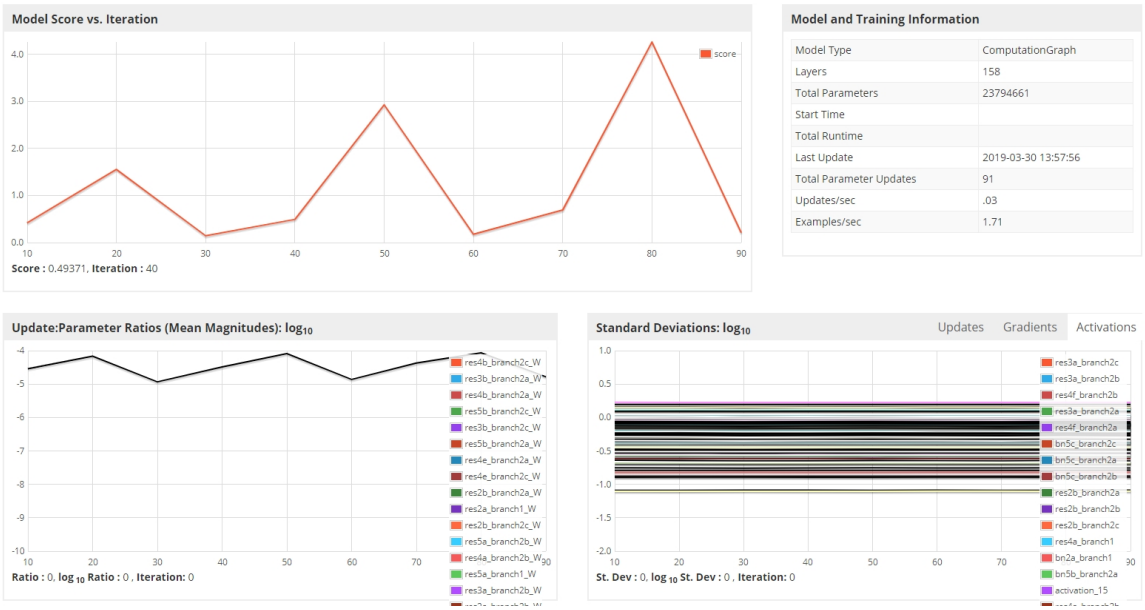
Aplikováním transformace trénovacích dat, při generování vstupů vzniká experiment ze kterého vyplývá, že konfigurace s vyšší mírou volatilních dat pro trénink je pomalejší z hlediska trvání učení, ale také výrazně eliminuje přetrénování výsledného grafu.

Při současném učení na více grafických kartách se trénování skoro lineárně zlepšovalo, avšak získat takhle výkonný stroj je velice obtížná úloha, ať už z hlediska nákladů nebo férového využití sdíleného výpočetního výkonu v rámci vedecké sféry.



Obrázek 6.1: Experimentální vyladění atributů pro učení klasifikace hluboce reziduálního grafu, zvyšuje přesnost při odhadu kategorie. Vertikálně vyznačená přesnost použitých modelů v závislosti na počtu iterací výpočetní techniky.

Až po představení převedeného učení, s počátečně inicializovanými vahami předtrénované na klasifikační soutěži rozlišující schopnosti současných algoritmů na tisíci kategoriích, vznikla možnost počítat komplexně zanořené grafové modely, které mrazí 150 vrstev při cyklech modifikace snížilo radikálně náročnost výpočtů jednoho průchodu modelem. Obecně identifikující informace interaktivních skenů změn učení s uzamčenými vrstvami uzlů a vazeb pro eliminace aplikace zpětné propagace gradientu, omezí službu poskytující možnost interaktivního náhledu na konkrétních vrstvy sítě o zmrazené neurony, pokud se podaří načtení předtrénovaných modelů, avšak pro průzkum zvolené architektury grafu je zapotřebí měnit veškeré vrstvy.



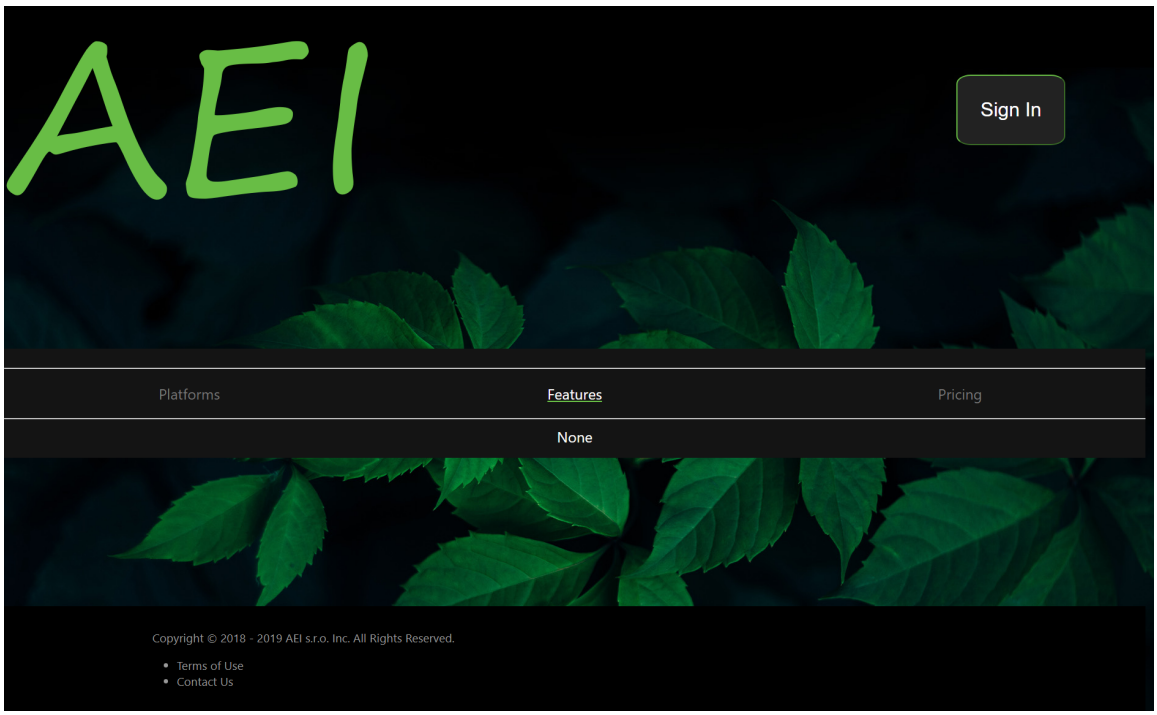
Obrázek 6.2: Pouze poslední plně propojená vrstva se trénuje, pro dosažení alespoň počátečních iterací složitého grafu na procesoru, napříč načteným datasetem v maticovém tvaru.

Obrázek 6.3: Náhodně vybraný vzorek fotografií jídla.

Kapitola 7

Uživatelské rozhraní

Aktuálně zvěřejněných stránek na internetu stále narůstá, proto je vhodné se zamyslet nad každým prvkem k čemu je vhodný a jaký má důvod, jelikož jinak uživatelé, kteří si kladou vysoké nároky na intuitivní ovládání s minimálním počtem rušící informace nevyužijí nabízené platformy, a i ostatní návštěvníci webové stránky přestanou mít po určitém čase pomalu zájem o celou aplikaci. Jako hlavní inspirace celého projektu o webovém rozhraní úvodního dokumentu pro nalákání uživatelů, byl momentálně Netflix, jakožto jeden z hlavních přispěvatelů digitálního obsahu na internetu, podporuje otevřený software zveřejněním zdrojových kódů vlastních architektornických řešení.



Obrázek 7.1: Úvodní webová stránka mající přilákat uživatele jednoduchostí.

Volbu ideálního množství veřejné informace a velikosti textu komplikuje věková rozmanitost možných uživatelů, proto s ohledem na seniornější uživatele byly zvoleny spíše větší styly písma. Funkce záhlaví spočívá ve vykreslesní zveřejněné informace určených částí, skládajících se z jednoduše zapamatovelného loga tvořícího povědomí s funkčním odkazem

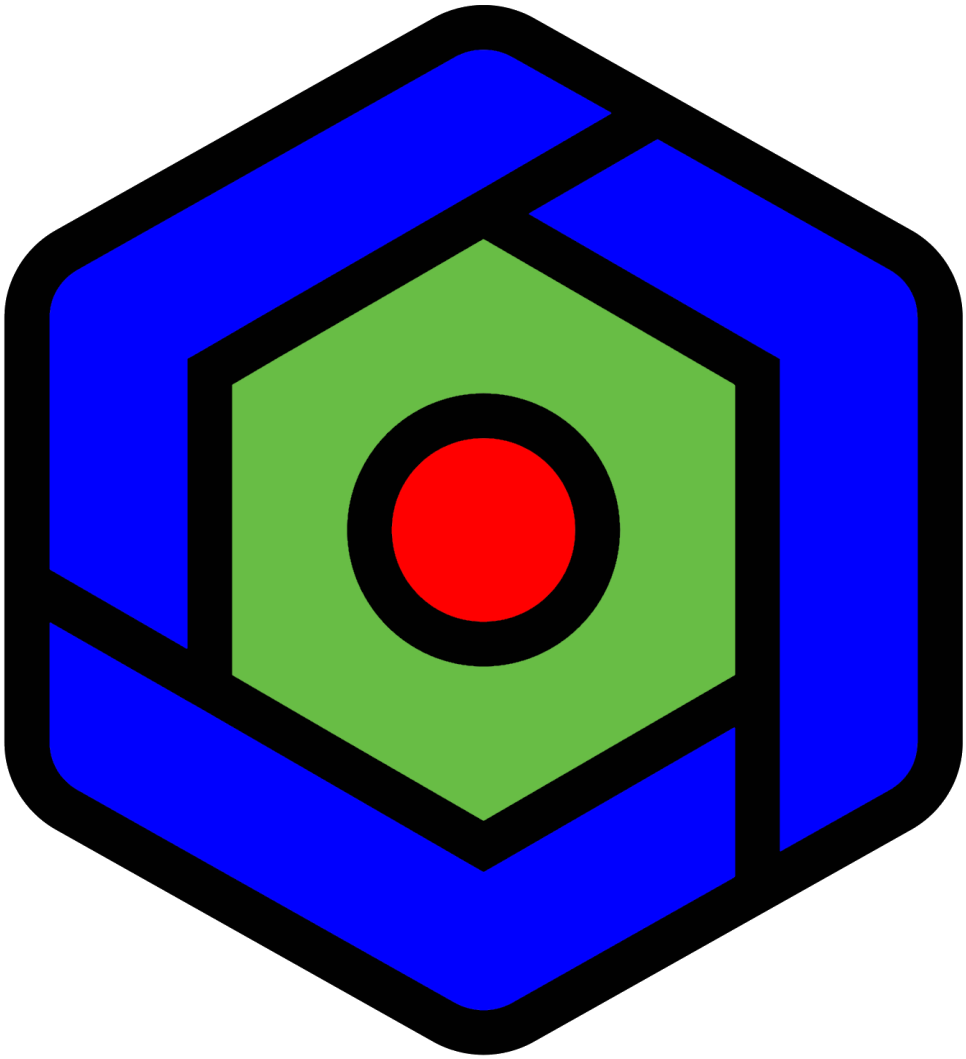
k hranici pro zpracování požadavků a definováním relevantně grafického pozadí v rámci poskytovaného obsahu. Počátečním zhodnocením unikátního aspektu nabízené důvěrnosti při zabezpečení přínosu s poplatky vznikne zřejmá důvěrnost s uživateli, která nabídne další možnost k získání pozitivní zkušenosti se systémem. V poslední řadě, by se měla v obsahu vyskytovat informace pro kontaktní spojení k právní ochraně vlastnictví odpovědné skupiny i odkaz pro přečtení a potvrzení podmínek použití.

Při vývoji byly postupně upravovaný styly částí webové stránky, aby responzivně vyhovovaly symetrickým požadavkům na jednoduchost a jednoznačnost, vzhledem k neurčitosti výsledného rozlišení obrazovky. Ideální rozpoložení bylo dosaženo pomocí číselné kombinace matematických sekvencí, z důvodu přirozeného vnímání tvarů a barev vyskytující se v přírody.

Nejtěžší volbou se stává barva tématu aplikace, aby byla ve výchozí hodnotě jasnější od zbytku využitých barev ve výsledném rozlišení, dnešní webové prohlížeče implementují i možnost dynamické změny zejména v mobilních zařízeních, s úrovněmi kontrastu pro dosažení maximální čitelnosti s ohledem na možné poruchy a vnímání zraku.

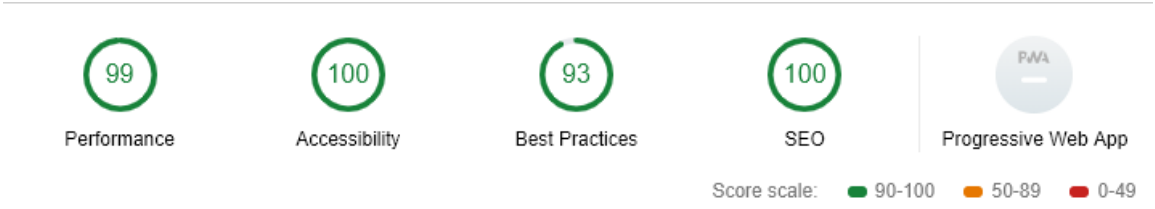
Samozřejmě koncová zařízení vybavena polohovací periferií, při provedení změn vyžadují dodatečné chování u zkušenosti s ovládáním, pro snadnou orientaci a zvýraznění aktuální pozice kurzoru na webové stránce pomocí nabízeného výstupního rozhraní s počítačem. Další možná kombinace vstupů přidá stisknutí znaků jako variantu ovládání, které webové prohlížeče implementují změnou zaostření a provedením akce.

Z důvodu nejednoznačnosti symbolu byl zvolen hexagon, kvůli nemožnosti převodu do vyššího dimenzionálního prostoru a významu počtu hran v možných pravidelných mnohoůhelníhových tvarů v cíleném rozměru, které mouhou být převedeny výše s kombinací aditivního modelu míchání barev pro vyzáření světla a zvoleného tématu reprezentativní barvy spektra.



Obrázek 7.2: Ikona zástupce přidružená ke konkrétní website.

Jelikož cílem celého návrhu se stává zaujmutí uživatele a udržením jej spokojeným při užívání, tak počáteční implementací vstupu poskytuje registraci s přihlášením. Výsledná aplikace splňuje aktuální požadavky pro dosažený výkon s možnou dostupností při dodržení osvědčených postupů u progresivních návrhů webových rozhraní.



Obrázek 7.3: Generovaná revize na mobilním zařízení s třetí generací mobilní telekomunikační technologie.

Kapitola 8

Závěr

Na základě provedených experimentů s neuronovými sítěmi při klasifikaci digitálního obrazu, předpokládám úspěšnost v rozpoznvání jídla 80%, po dokončení alespoň jedné trénovací iterace přes sesbíraný dataset, která se skládá z 2048 epoch virtuálně separovaných dat. Zhodnocením výsledků služby rozpoznávající jídlo přináší schopnost automatizovaně třídit obraz podle aktuální struktury adresáře. Na základě zjištěných údajů je možné použít moduly k obecné klasifikaci dvou rozměrně digitálního obrazu, a pro sestavení vlastního datacentra, který používá architekturu mikroslužeb k efektivnějšímu vývoji prototypů. Zvolené problémy jsem vyřešil pomocí jazyka běžícím na Java Virtual Machine, ve kterém se podařilo dosáhnout polyglotního základu proaktivních aplikací.

Další vývoj by se zaměřil na drobné doladění pojmů a přidání dynamicky nebo periodicky se opakujícího jevu, který mění optimalizace serverů. Přínosem technologie automatické správy projektu pokračují autonomní systémy, kde z pohledu užívání doporučuji nejprve pochopit problematiku pomocí dostupných odkazů a textu, před přidáním služby pro přiblížením fraktálního porovnání segmentace a regrese barev, která rozšíří vyhlazující funkce podobnosti.

Vytvořené řešení poskytuje responsivně generované uživatelské rozhraní pro webové prohlížeče, avšak pouze pro úvodní seznámení s nabízenou infrastrukturou, která představuje hlavní přínos zpracováním nové modulární platformy se stabilnějších chováním. Výsledky této práce zjednoduší výzkum paralelních systémů, z důvodu vzniklé nezávislosti souběžného spuštění komponent infrastruktury a možné rozšířitelnosti služeb při minimálním programování, s celkovým základem distribuované virtualizace.

Metakognitivním naplnění faktu, že už při jednoročním výzkumu paralelně distribuovaných systémů, jsem byl schopen rozpoznat slabiny řešení implementující stejná paradigma po nahlédnutí na zdrojový kód polyglotních jazyků, což se zdá být obtížné vzhledem na běžný počet jader v přenositelných zařízeních, usuzuji budoucí zaměření na teorie grafů z důvodu podobnosti problematiky k neuronovým sítím, který byl vybrán pro vlastní touhy měnit a zlepšit nezávislost účelu virtualizovaného prostředí nabízející výpočetní výkon a pamět, jednodochý avšak komplexně jednoznačným způsobem popisuje řešení grafů.

Jelikož se jedná o aplikaci nové technologie, která byla zvěřejněna teprve nedávno, je zapotřebí dát širší oblasti lidí čas na dokončení implementace veškerých možných nástrojů v rámci definic všech dostupných příloh frameworku, ačkoliv přináší řešení delimitace proaktivních od reaktivních aspektů systému, tak rozšířením mění schopnosti reakce, upozorňuji, že některé nevhodné operace zasahující do serializace obsahu pamětí cache zapříčiní momentálně nejednoznačný pád souvisejících částí systému.

Implementace koherence plně nezávislého systému pravděpodobně nebude nikdy kompletně dokončena, jelikož vžy existuje možnost rozšíření technologie o principy chaosu, proto je důležité, aby práce abstrahovaly a rešily co největší množství zvolené problematiky, v mém případě z oblasti matematické informatiky.

Literatura

[1] Aho, A. V.; Lam, M. S.; Sethi, R.; aj.: *Compilers: Principles, Techniques, and Tools 2nd Edition*. Addison Wesley, 2006, ISBN 978-0321486813.

[2] Belshe, M.; BitGo; Peon, R.; aj.: *Hypertext Transfer Protocol Version 2 (HTTP/2)*. [Online; navštíveno 15.05.2019].
URL https://tools.ietf.org/html/rfc7540

[3] Charlie Hunt, P. P., Monica Beckwith; Rutisson, B.: *Java Performance Companion*. Addison-Wesley Professional, 2016, ISBN 978-0133796827.

[4] Graves, A.: *Supervised Sequence Labelling with Recurrent Neural Networks*. [Online; navštíveno 22.04.2019].
URL https://www.tum.de

[5] Ibryam, B.: *It takes more than a Circuit Breaker to create a resilient application*. [Online; navštíveno 10.05.2019].
URL https://developers.redhat.com

[6] Jones, M.; Microsoft: *JSON Web Key (JWK)*. [Online; navštíveno 15.05.2019].
URL https://tools.ietf.org/html/rfc7517

[7] Joutou, T.; Yanai, K.: *A food image recognition system with multiple kernel learning*. [Online; navštíveno 22.04.2019].
URL https://www.uec.ac.jp

[8] K. LI, E.; Group, A.; Hunt, P.; aj.: *System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements*. [Online; navštíveno 15.05.2019].
URL https://tools.ietf.org/html/rfc7642

[9] Kaiming He, S. R., Xiangyu Zhang; Sun, J.: *Deep Residual Learning for Image Recognition*. [Online; navštíveno 22.04.2019].
URL https://arxiv.org/pdf/1512.03385.pdf

[10] Lu, Y.: *Food Image Recognition by Using Convolutional Neural Networks (CNNs)*. [Online; navštíveno 22.04.2019].
URL https://msu.edu

[11] Lukas Bossard, M. G.; Gool, L. V.: *Food-101 - Mining Discriminative Components with Random Forests*. [Online; navštíveno 22.04.2019].
URL https://www.vision.ee.ethz.ch

[12] László, E.: *Věda a ákašické pole - Integrální teorie všeho*. Pragma, 2005, ISBN 80-7205-216-0.

[13] Netflix: *Hystrix*. [Online; navštíveno 10.04.2019].
URL https://github.com/Netflix/Hystrix/wiki

[14] Niki Martinel, G. L. F.; Micheloni, C.: *Wide-Slice Residual Networks for Food Recognition*. [Online; navštíveno 22.04.2019].
URL https://www.uniud.it

[15] Oracle: *Java Platform Standard Edition 11 Documentation*. [Online; navštíveno 10.05.2019].
URL https://docs.oracle.com

[16] Qian Yu, D. M.; Wang, J.: *Deep Learning Based Food Recognition*. [Online; navštíveno 22.04.2019].
URL https://www.stanford.edu

[17] Spring: *Spring Framework*. [Online; navštíveno 13.05.2019].
URL https://spring.io

[18] Szegedy, C.; Wei Liu, Y. J.; Sermanet, P.; aj.: *Going deeper with convolutions*. [Online; navštíveno 22.04.2019].
URL https://arxiv.org/pdf/1409.4842.pdf

[19] Yuji Matsuda, H. H.; Yanai, K.: *Recognition of multiple-food images by detecting candidate regions*. [Online; navštíveno 22.04.2019].
URL https://www.uec.ac.jp

Přílohy

Seznam příloh

Příloha A

Obsah USB flash disku

Adresářová struktura	Význam obsahu struktury
company	Projekt renderovací úvodní uživatelské rozhraní.
configuration	Obsah veřejně dostupné vzdálené konfigurace.
proactive-web	Implementovaná architektura distribuovaného systému se všemi informacemi.
research	Experimentálně vědecké skripty pro práci s neuronovými sítěmi.
thesis	Zdrojové informace k sestavení Bakalářské práce.
xdolez71.pub	*PGP* klíč pro ověření asymetrické kryptografie autora práce.

Příloha B

Aplikace

```
------------------------------------------------------------
Tasks runnable from root project
------------------------------------------------------------

Application tasks
-----------------
bootRun - Runs this project as a Spring Boot application.
run - Runs this project as a JVM application

Build tasks
-----------
assemble - Assembles the outputs of this project.
bootBuildInfo - Generates a META-INF/build-info.properties file.
bootJar - Assembles an executable jar archive containing the main classes and their dependencies.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
generateGitProperties - Generate a git.properties file.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles test classes.

Build Setup tasks
-----------------
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Distribution tasks
------------------
assembleBootDist - Assembles the boot distributions
assembleDist - Assembles the main distributions
bootDistTar - Bundles the project as a distribution.
bootDistZip - Bundles the project as a distribution.
distTar - Bundles the project as a distribution.
distZip - Bundles the project as a distribution.
installBootDist - Installs the project as a distribution as-is.
installDist - Installs the project as a distribution as-is.

Docker tasks
------------
docker - Builds Docker image.
dockerClean - Cleans Docker build directory.
dockerfileZip - Bundles the configured Dockerfile in a zip file
dockerPrepare - Prepares Docker build directory.
dockerPush - Pushes named Docker image to configured Docker Hub.
dockerTag - Applies all tags to the Docker image.
dockerTagsPush - Pushes all tagged Docker images to configured Docker Hub.

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project 'proactive-web'.
components - Displays the components produced by root project 'proactive-web'. [incubating]
dependencies - Displays all dependencies declared in root project 'proactive-web'.
dependencyInsight - Displays the insight into a specific dependency in root project 'proactive-web'.
dependencyManagement - Displays the dependency management declared in root project 'proactive-web'.
dependentComponents - Displays the dependent components of components in root project 'proactive-web'. [incubating]
help - Displays a help message.
model - Displays the configuration model of root project 'proactive-web'. [incubating]
projects - Displays the sub-projects of root project 'proactive-web'.
properties - Displays the properties of root project 'proactive-web'.
tasks - Displays the tasks runnable from root project 'proactive-web' (some of the displayed tasks may belong to subprojects).
```

Obrázek B.1: Vývoj.

Příloha C

Podpora

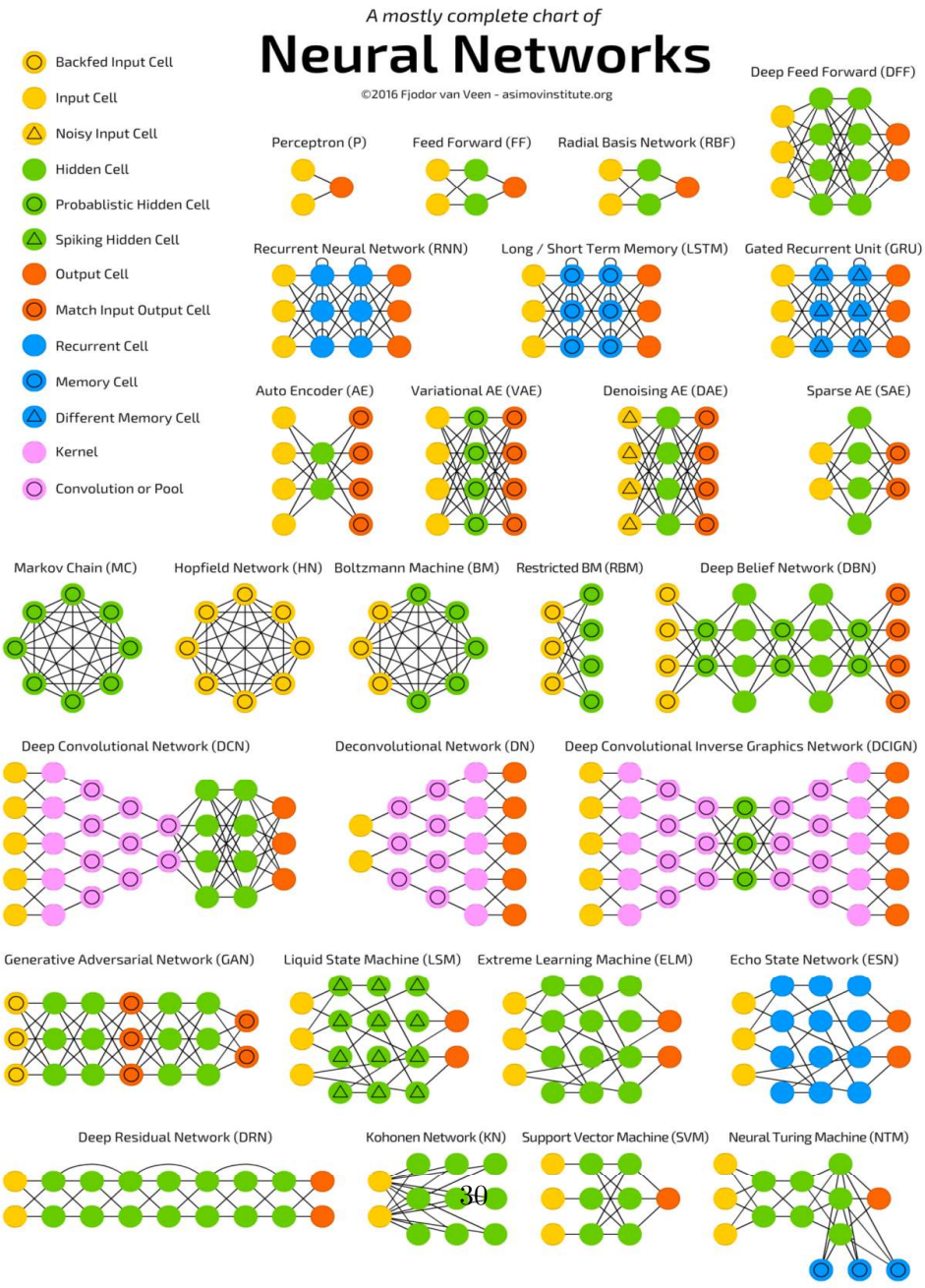
```
IDE tasks
---------
cleanEclipse - Cleans all Eclipse files.
cleanIdea - Cleans IDEA project files (IML, IPR)
eclipse - Generates all Eclipse files.
idea - Generates IDEA project files (IML, IPR, IWS)
openIdea - Opens the IDEA project

Management tasks
----------------
clear - Clears your workspace.
copyright - Updates all projects copyrights.
create - Creates new SaaS application. -Papp=example
permission - Updates all projects executable permissions.

Publishing tasks
----------------
generateMetadataFileForJavaProjectPublication - Generates the Gradle metadata file for publication 'javaProject'.
generatePomFileForJavaProjectPublication - Generates the Maven POM file for publication 'javaProject'.
publish - Publishes all publications produced by this project.
publishAllPublicationsToBuildRepository - Publishes all Maven publications produced by this project to the Build repository.
publishAllPublicationsToNexusRepository - Publishes all Maven publications produced by this project to the Nexus repository.
publishJavaProjectPublicationToBuildRepository - Publishes Maven publication 'javaProject' to Maven repository 'Build'.
publishJavaProjectPublicationToMavenLocal - Publishes Maven publication 'javaProject' to the local Maven repository.
publishJavaProjectPublicationToNexusRepository - Publishes Maven publication 'javaProject' to Maven repository 'Nexus'.
publishToMavenLocal - Publishes all Maven publications produced by this project to the local Maven cache.

Reporting tasks
---------------
analyze - Statically analyzes project source code.
projectReport - Generates a report about your project.
repository - GIT repository statistics.

Verification tasks
------------------
check - Runs all checks.
jacocoTestCoverageVerification - Verifies code coverage metrics based on specified rules for the test task.
jacocoTestReport - Generates code coverage report for the test task.
test - Runs the unit tests.

Version tasks
-------------
major - Next major project source code version.
minor - Next minor project source code version.
release - Releases project source code.
snapshot - Snapshots project source code.
version - Project source code version.
```

Obrázek C.1: Řízení.

Příloha D

Síť

Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data



Příloha E

Web

Micro service platform Infrastructure

A software development technique a variant of the service oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a micro services architecture, services are fine grained and the protocols are lightweight.

- Properly scoped functionality
- Presentation an API
- Traffic management
- Data offloading
- Monitoring

Effective ***implementation*** and *code* ***architecture*** are two major aspects to be considered during software development.

Micro service provides definition for simple communication between scalable applications.

About

Single option of platform for the development of a Spring Boot or a pure Java application. There is no need to resolve unnecessary problems as a connection between applications. The concept is clean Micro Service Architecture in my view. That must be properly defined otherwise it is not a micro service if data monolith is created. Comparison of the effectiveness of the current system and the development process.

Cloud

Connection with cloud service is established through the cloud scaled proxy cluster. Load balanced request is redirected based on cloud discovery record. Requests and messages on the cloud bus are possible to trace. Every request have to be authenticated and authorized as *SSO*. During the update or dump component cloud will remain stable. Contains self discovery, realtime configuration, load balancing, measurements, resilience, messaging, logging and turbine aggregation functionality.