SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-52598

AUTOMATIZOVANÉ GENEROVANIE APLIKAČNÉHO ROZHRANIAVPROCESNEORIENTOVANÝCH SYSTÉMOCH

DIPLOMOVÁ PRÁCA

2019 **Juraj Kubričan**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-52598

AUTOMATIZOVANÉ GENEROVANIE APLIKAČNÉHO ROZHRANIAVPROCESNEORIENTOVANÝCH SYSTÉMOCH

DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Číslo študijného odboru:	2511
Názov študijného odboru:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	prof. RNDr. Gabriel Juhás, PhD.
Konzultant:	Ing. Milan Mladoniczky

Bratislava 2019 **Juraj Kubričan**

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Juraj Kubričan
Diplomová práca:	Automatizované generovanie aplikačného rozhraniavprocesneorientovaných systémoch
Vedúci záverečnej práce:	prof. RNDr. Gabriel Juhás, PhD.
Konzultant:	Ing. Milan Mladoniczky
Miesto a rok predloženia práce:	Bratislava 2019

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean et est a dui semper facilisis. Pellentesque placerat elit a nunc. Nullam tortor odio, rutrum quis, egestas ut, posuere sed, felis. Vestibulum placerat feugiat nisl. Suspendisse lacinia, odio non feugiat vestibulum, sem erat blandit metus, ac nonummy magna odio pharetra felis. Vivamus vehicula velit non metus faucibus auctor. Nam sed augue. Donec orci. Cras eget diam et dolor dapibus sollicitudin. In lacinia, tellus vitae laoreet ultrices, lectus ligula dictum dui, eget condimentum velit dui vitae ante. Nulla nonummy augue nec pede. Pellentesque ut nulla. Donec at libero. Pellentesque at nisl ac nisi fermentum viverra. Praesent odio. Phasellus tincidunt diam ut ipsum. Donec eget est. A skúška mäkčeňov a dĺžnov.

Kľúčové slová: kľúčové slovo1, kľúčové slovo2, kľúčové slovo3

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Juraj Kubričan
Master's thesis:	Automaticgenerationofapplicationinterface inprocessorientedsys-tems
Supervisor:	prof. RNDr. Gabriel Juhás, PhD.
Consultant:	Ing. Milan Mladoniczky
Place and year of submission:	Bratislava 2019

On the other hand, we denounce with righteous indignation and dislike men who are so beguiled and demoralized by the charms of pleasure of the moment, so blinded by desire, that they cannot foresee the pain and trouble that are bound to ensue; and equal blame belongs to those who fail in their duty through weakness of will, which is the same as saying through shrinking from toil and pain. These cases are perfectly simple and easy to distinguish. In a free hour, when our power of choice is untrammelled and when nothing prevents our being able to do what we like best, every pleasure is to be welcomed and every pain avoided. But in certain circumstances and owing to the claims of duty or the obligations of business it will frequently occur that pleasures have to be repudiated and annoyances accepted. The wise man therefore always holds in these matters to this principle of selection: he rejects pleasures to secure other greater pleasures, or else he endures pains to avoid worse pains.

Keywords: keyword1, keyword2, keyword3

Poďakovanie

I would like to express a gratitude to my thesis supervisor.

Obsah

Zoznam obrázkov a tabuliek

Zoznam algoritmov

Zoznam výpisov

Úvod

Tu bude krasny uvod s diakritikou atd.
A mozno aj viac riadkovy uvod.

1 Analýza problému

1.1 Úvod do petriho sietí

Petriho siete sú modelovací nástroj určený na modelovanie udalostných systémov.

1.2 Procesný server

Majme procesný server tento server v sebe obsahuje informáciu o rôznych procesoch. Tieto procesy majú rôznu štruktúru a sú v rôznych stavoch. Štruktúru týchto procesov môžeme reprezentovať pomocou Petriho sietí a ich stav pomocou rôznych značkovaní v týchto sieťach.

1.3 Aplikačné rozhranie

Na to aby sme mohli k dátam pristupovať zvonka uzavretej domény procesného servera potrebujeme vytvoriť aplikačné rozhranie ktoré bude riešiť autentifikáciu, autorizáciu a dokumentáciu.

2 Špecifikácia

V tejto kapitole najprv stručne opíšeme hlavnú funkcionalitu navrhovanej aplikácie, potom zadefinujeme funkcionálne a nefunkcionálne požiadavky na aplikáciu.

Softvér, ktorý sme sa rozhodli implementovať bude slúžiť ako rozhranie medzi procesným serverom a internetom. Bude umožňovať klientovi pripojiť sa na procesný server cez internet, získať informácie o dátach v prechodoch petriho siete a bude umožňovať modifikovať stav siete (procesu) spúšťaním prechodov. [**t01**] sdkjjsad

2.1 Funkcionálne požiadavky

1. Rozhranie bude umožňovať registrovať používateľov a priraďovať im roly
2. Umožní prihlásenie používateľa pomocou štandardného autentifikačného protokolu.
3. Autentifikovaným používateľom umožní prístup k dátam z tých prechodov ktoré majú právo čítať podľa ich roly.
4. Autentifikovaným používateľom umožní spúšťať prechody ktoré majú právo spúšťať podľa ich roly.
5. Pri spúšťaní prechodu prebehne validácia vstupných dát. V prípade nevalidných alebo nekompletných dát nepovolí spustenie prechodu.
6. Rozhranie poskytne online dokumentáciu prechodov v sieti, táto dokumentácia bude zahŕňať URL prechodu, potrebné dátové polia na spustenie prechodu a roly, ktoré sú oprávnené prechody spúšťať.
7. Rozhranie poskytne aplikačné rozhranie viacerým sieťam s rôznou štruktúrou.

2.2 Nefunkcionálne požiadavky

1. Rozhranie bude škálovateľné
2. Rozhranie bude napísané vo frameworku Spring Boot
3. Rozhranie bude zabezpečené štandartnými bezpečnosntými prvkami

3 Návrh

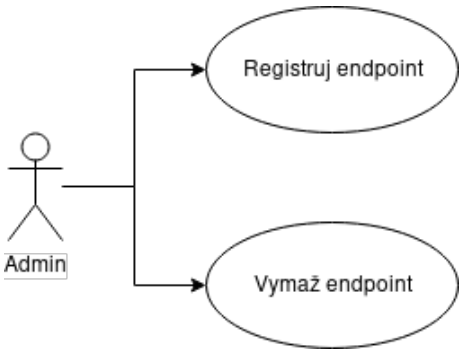
3.1 Prípady použitia

Zo špecifikácie vyplývajú nasledovné interakcie používateľa s naším systémom.

3.1.1 Registrácia koncového bodu

Prvý prípad použitia 1 je registrácia koncového bodu. Aktér ktorý môže registrovať koncové body je len administrátor. Pri tejto akcií administrátor poskytne nášmu systému informácie o štruktúre procesu vo formáte petriflow, zoznam používateľov im prislúchajúcich rolí v XML a unikátny identifikátor siete. Pokial che administrátor upraviť spustí proces registrácie nanovo s upravenými údaji o sieti.

Administrátor taktiež môže sieť vymazať. 1



Obr. 1: Prípad použitia - Registrácia koncového bodu

3.1.2 Získanie informácie o prechode

Keď už sú sieť aj používateľia úspešne zaregistrovaný, si môžu používatelia, vyžiadať 2 informácie o prechode. Tieto informácie budú poskytnuté len používateľovi s rolou oprávnenou na čítanie dát z daného prechodu.

3.1.3 Spustenie prechodu

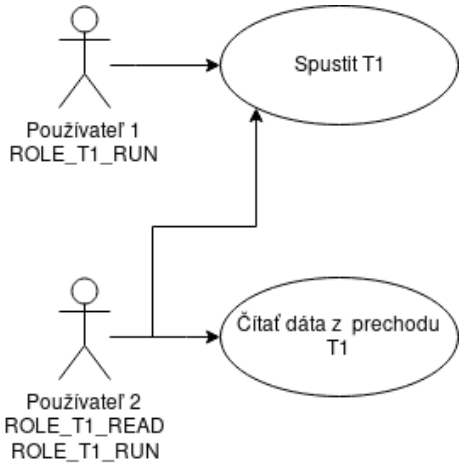
Používateľia s príslušnými rolami môžu taktie spúšťať 2 prechody. Pri spustení prechodu poskytne používateľ dátové polia potrebné na spustenie prechodu.

3.1.4 Autentifikácia

Z požiadavky na bezpečnosť aplikácie vyplýva ešte prípad použitia, kedy sa neautentifikovaný používateľ autentifikuje, aby nadobudol identitu rozpoznanú našim systémom.

3.2 Architektúra

Aby sme splnili požiadavku na jednoduché škálovanie aplikácie a pre sprehľadnenie architektúry zvolili sme si architektúru mikroservisov. Táto architektúra pozostáva z via-



Obr. 2: Prípad použitia - Získanie informácií o prechode

cerých oddelených častí, každá z týchto častí má svoju jasne definovanú funkciu. Takéto mikroservisy sú jednoducho testovateľné, dajú sa nasadzovať postupne a nezávisle od seba a softvér navrhnutý v tejto architektúre býva spravidla robustný a vysoko škálovateľný.

4 Implementácia

4.1 Použité technológie

4.1.1 Kotlin

Kotlin je relatívne nový programovací jazyk, projekt Kotlin bol po prvý krát zverejnený v roku 2011 spoločnosťou JetBrains(Andrey Breslav). Bol vyvinutý ako moderný staticky typovaný jazyk, ktorý podporuje rýchlu kompiláciu do javy. V roku 2017 vyhlásil Google podporu pre Kotlin v operačnom systéme Android.

Medzi jeho hlavné výhody patrí menší boilerplate (menej zbytočného kódu),

Vylepšený systém typovania premenných. premenné môžu byť null a kompilátor vie odvodiť v ktorých prípadoch premenná null obsahovať môže a v ktorých nie,

Jednoduchosť prechodu z Javy na kotlin štruktúra kódu je podobná jave, Jetbrains dokonca poskytuje transpiler, ktorý dokáže kód z Javy vo väčšine prípadov trasnpilovať do Kotlinu

Kotlin beží na JVM, takže sa dá používať na všetkých bežných platformách.

4.1.2 Gradle

Gradle je voľne šíriteľný nástroj na automatizáciu zostavovania softvéru. Je stavaný na to aby bol schopný zostaviť takmer ľubovoľný program. Podporuje jazyky ako java, C++ Python, a mnoho ďalších. V našom projekte sa gradle použijeme na manažment závislostí, kompiláciu kódu a spustenie samotného skompilovaného programu. Konfigurácia nástroja prebieha pomocou konfiguračného súboru napísaného v jazyku Groovy, tieto konfiguračné súbory sa v našom prípade použitia ukázali ako veľmi prehľadné a ľahké na použitie. Gradle taktiež používa pokročilú techniku memoizácie procesu zostavovania softvéru takže jeho výkon je pri opakovanej kompilácií vyšší.

4.1.3 Spring boot

Spring Boot je voľne šíriteľný framework založený na Jave. Je vyvíjaný a udržiavaný tímom Pivotal. Je určený na vytváranie nezávislých, produkčných aplikácií a mikro-služieb.

Pri práci so Spring boot budeme využívať návrhové vzory:

Dependency injection / Inversion of control (Tým že v jednoduchých triedach pridáme anotáciu, vieme z frameworku zdediť nielen funkcie, ale aj control flow)

Singleton (aplikácia vie zaručiť že z daného objektu sa v rámci jednej inštancie aplikácie vytvorí len jedna inštancia, ku ktorej sa dá pristupovať z celej aplikácie)

Factory (aplikácia používa na vytváranie objektov tzv Beanov návrhový vzor factory)

4.1.4 Spring cloud

Spring Cloud je framework, ktorý obsahuje bohatú sadu nástrojov na vytváranie mikroslužieb a cloudových riešení. Medzi nástroje Spring Clopudu patí:

- Cloud config - nástroj na distribúciu konfiguračných súborov medzi kontajnermi mikrosluzieb
- Service discovery - nástroj na registráciu a monitorovanie mikroservisov
- Gateway - Nástroj na routovanie a load balancing v rámci mikroservisov
- Cloud Authentication - Nástroj na riešenie komplexnej autentizácie a autorizácie v rámci

4.2 Inštalácia a konfigurácia kontajnerov

Všetky Spring Boot kontajnery sme inštalovali pomocou spring initializr

tento nástroj vygeneruje zip súbor so založeným projektom vo frameworku Spring Boot. Pri vytváraní projektu je možné si vybrať Jazyk v ktorom bude projekt založený a nástroj ktorý bude projekt zostavovať

4.2.1 Inštalácia a konfigurácia kontajnerov

5 Testovanie

Na testovanie sme použili softvér Insomnia REST Client[**insomnia**]. Tento program je určený na testovanie REST a GraphQL služieb. Je to voľne šíriteľná alternatíva známeho programu Postman, postavená na platforme Electron s použitím knižnice React. Insomnia nám dovolí vytvoriť a uložiť viacero testovacích dopytov, ktoré môžeme neskôr spustiť a overiť ich správne fungovanie. Insomnia taktiež podporuje autentifikačný protokol OAuth2, takže nám stačí iba zadať prístupové údaje a autorizačný token si stiahne sama. Taktiež v prípade vypršania autorizačného tokenu ho automaticky obnoví.

6 Záver

Táto architektúra však nie je vhodná na menšie projekty, lebo réžia vzniknutého softvéru býva spravidla vysoká, lebo si vyžaduje viacero spustených inštancií servisov. Tiež nie je vhodná v prípadoch, kde sa čakávajú väčšie zmeny biznis logiky aplikácie. Pri väčšej smene biznis logiky je často nutné prerábať viacero servisov a zmena protokolu, ktorým medzi sebou komunikujú.

Záver

Conclusion is going to be where?
Here.

Prílohy

A Štruktúra elektronického nosiča

/CHANGELOG.md

· file describing changes made to FEIstyle

/example.tex

· main example *.tex* file for diploma thesis

/example_paper.tex

· example *.tex* file for seminar paper

/Makefile

· simply Makefile – build system

/fei.sublime-project

· is project file with build in Build System for Sublime Text 3

/img

· folder with images

/includes

· files with content

/bibliography.bib

· bibliography file

/attachmentA.tex

· this very file

B Algoritmus

Algorithm B.1 Vypočítaj $y = x^n$

Require: $n \geq 0 \vee x \neq 0$

Ensure: $y = x^n$

```
y ⇐ 1
if n < 0 then
    X ⇐ 1/x
    N ⇐ −n
else
    X ⇐ x
    N ⇐ n
end if
while N ≠ 0 do
    if N is even then
        X ⇐ X × X
        N ⇐ N/2
    else {N is odd}
        y ⇐ y × X
        N ⇐ N − 1
    end if
end while
```

C Výpis sublime

```
../../ fei .sublime-project
```

Listing C.1: Ukážka sublime-project