|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Тамара Глишић

**...................................... ..........................................**

ДИПЛОМСКИ РАД

- Основне академске студије -

Нови Сад, 2021.

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист: |
| 1/1 |

*(Податке уноси предметни наставник - ментор)*

| Врста студија: | **Основне академске студије** |
| --- | --- |
| Студијски програм: | **Софтверско инжењерство и информационе технологије** |
| Руководилац студијског програма: | **ван. проф. др Мирослав Зарић** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Тамара Глишић** | Број индекса: | **SW38/2017** |
| Област: | **Системи базирани на знању** | | |
| Ментор: | **доц. др Синиша Николић** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА  ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:**

|  |
| --- |
|  |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за:  - Студента;  - Ментора |

Table of Contents

[1. УВОД 1](#_Toc80882613)

[2. СТАЊЕ У ОБЛАСТИ 2](#_Toc80882614)

[3. КОРИШЋЕНЕ ТЕХНИКЕ И ТЕХНОЛОГИЈЕ 3](#_Toc80882615)

[3.1. Angular 3](#_Toc80882616)

[3.2. Spring 4](#_Toc80882617)

[3.3. Drools 6](#_Toc80882618)

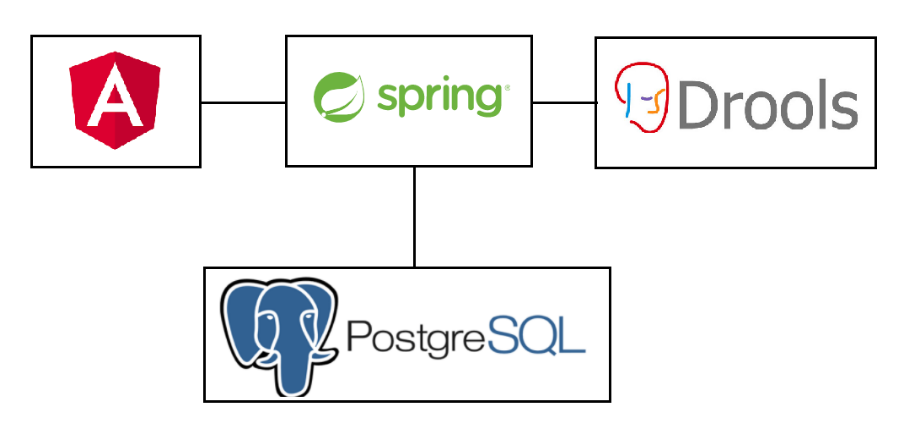
[3.4. REST сервис 7](#_Toc80882619)

# УВОД

# СТАЊЕ У ОБЛАСТИ

# КОРИШЋЕНЕ ТЕХНИКЕ И ТЕХНОЛОГИЈЕ

Решење је имплементирано као веб апликација чија архитектура обухвата клијентски и серверски део, базу података и систем базиран на правилима (Слика 3.1). Клијентски део имплементиран је употребом *Angular* радног оквира, за серверски део коришћен је *SpringBoot* заједно са *REST (Representational State Transfer)*  сервисима који за рамену података користе *JSON (JavaScript Object Notation)* формат. Размена података и комуникација обавља се путем *HTTP (Hyper Text Transfer Protocol)* протокола. Као база података коришћен је *PostgreSQL*, а систем базиран на правилима је *Drools.*



Слика 3.1 Архитектура веб апликације

## Angular

*Angular* је радни оквир за развој клијентских апликација употребом *HTML-a* (*Hyper Text Markup Language*) и *TypeScript-a.* Сам радни оквир написан је у *TypeScript-у,* а његов основни градивни блок су компоненте. Поред компонената, клијентска апликација писана у *Angular-у* састоји се још и од модула и сервиса, што чини три главна дела апликације.

***Компоненте***

Компоненте обједињују приказ, програмску логику која га контролише и податке. Програмска логика налази се у *TypeScript* класама које су повезане са *HTML* темплејтом који дефинише начин на који ће клијенту бити приказани подаци.

***Модули***

*Angular* апликације су модуларне и састоје се из бар једног (коренског) модула. Модули су *TypeScript* класе које обједињују различите компоненте, и омогућују да се компоенете из конкретног домена извезу како би могле бити коришћене у неком другом модулу или обрнуто, да могу да се увезу из неког другог модула и користе у конкретном модулу.

***Сервиси***

Да би клијентска апликација комуницирала са серверским делом, користе се сервиси. Сервиси су *TypeScript* класе у којима се налазе подаци и логика која није карактеристична за само једну компоненту већ може бити дељена са више компоненти.

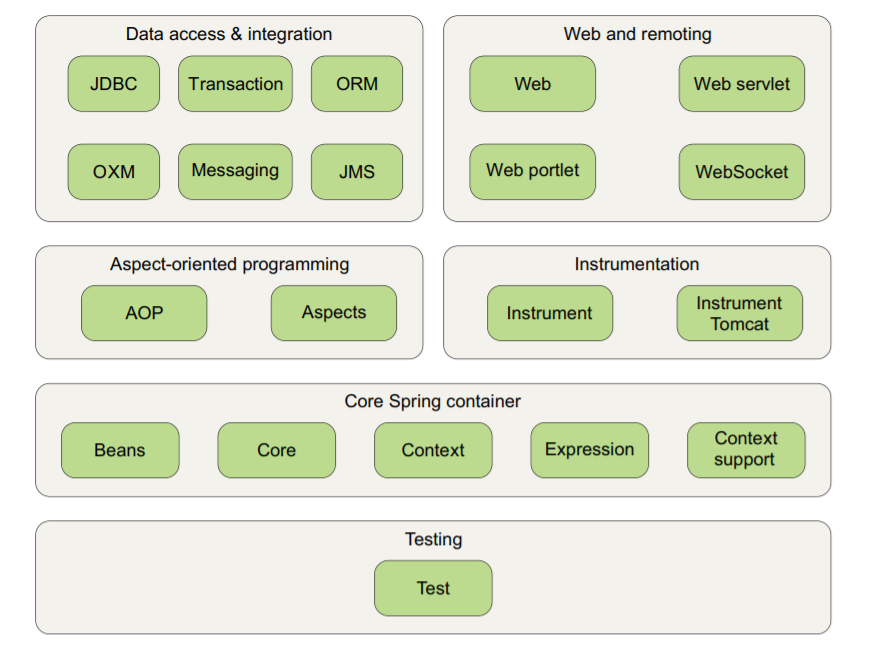
## Spring

*Spring* је *open-source* радни оквир написан у *Java* програмском језику. Настао је са циљем да омогући једноставнији развој *Java* пословних апликација. (референца на кнјигу СПринг ин ацтион). Главни концепти *Spring* радног оквира су *Dependency injection* (DI) и *Aspect-oriented programming* (AOP).

DI је механизам за повезивање објеката уз помоћ којег објекат добија своје зависности уместо да их сам добавља.

АОР омогућује поновну искоришћење компоненти. На овај начин функционалности се независно дефинишу, а затим се аспекти примењују у потребним деловима апликације.

*Spring* радни оквир се састоји из шест јасно дефинисаних модула (Слика 3.2).



Слика 3.2 Приказ модула у Spring-у [ ref na knjigu ]

***Core Spring container***

Главни део *Spring* радног оквира је Контејнер енгл. *Container* који управља животним циклусом објеката, тј начином на који ће они бити креирани, конфигурисани и коришћени. Овај модул садржи фабрику објеката која омогућава DI механизам при коришћењу објеката.

***Aspect – oriented programming***

Један од главних концепата, овај модул пружа основе за разовј личних аспеката у *Spring* апликацији. Са АОР проблеми везани за апликацију, као што су трансакције и безбедност, су одвојени од објеката на које се примењују.

***Data access and integration***

Модул за управљање подацима и интеграцију апстрахује везу са базом података. Рад са *Java Database Connectivity (JDBC)* и *Data – access objects (DAO)* модулима омогућују краћи и једоставнији код одвојен од базе и на тај начин омогућује експесивније поруке о SQL грешкама независно од система за управљање базом података. *Spring* има подршку за објектно – релационо мапирање (ОRM) али овај модул нема своју ORM имплементацију већ користи неку од постојећих као што су *Hibernate, OpenJPA* итд.

***Web and remoting***

Овај модул садржи радни оквир по *Model – View – Controller (MVC)* парадигми која је општеприхваћен приступ за изградњу и развој комплетних веб апликација. MVC гарантује да је *User interface (UI)* одвојен од апликативне логике. Поред веб апликација, овај модул пружа подршку за позиве удањених метода и на тај начин омогућују комуникацију са другим апликацијама.

***Instrumentation***

Омогућује прилагођавање рада *Java Virtual Machine (JVM)*, конкретно, садржи измењене начине за учитавање класа тј. трансформише их уз помоћ специфичне имплементације *ClassLoader* објеката.

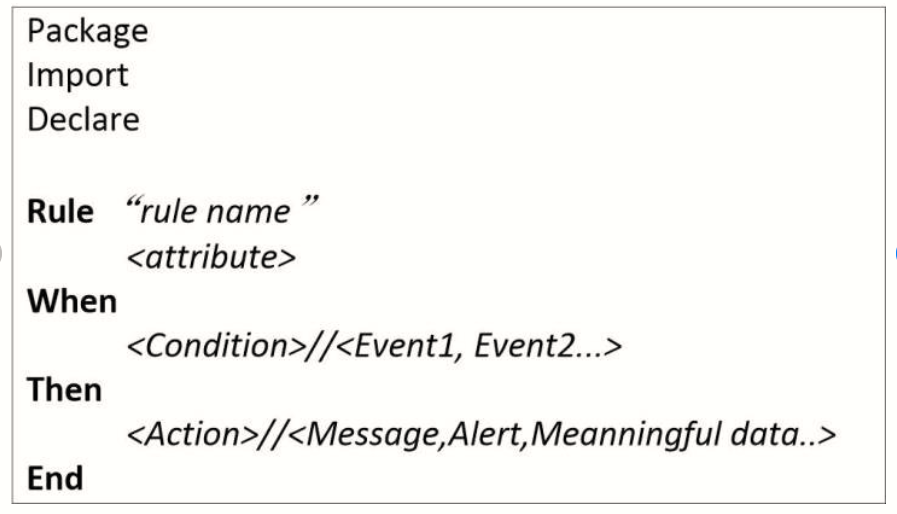
***Testing***

Овај модул омогућује јединично и интеграционо тестирање засновано на *Junit* и *TestNG* библиотекама. Садржи и имплементацију *mock* објеката који могу бити коришћени за јединичне тестове изоловано од других делова апликације. *Mock* објекат је објекат двојник који симулира понашање стварног објекта.

## Drools

*Drools* је систем за управљање пословним правилима и имплементацију експертских система који опонашају доношење одлука људског стручњака из неког домена. Заснован је на закључивању унапред и уназад и подржава *Complex event processing (CEP)* за откривање и обраду значајних догађаја у систему.

*Drools* има своју синтаксу за писање правила која прати структуру приказану на слици 3.3.



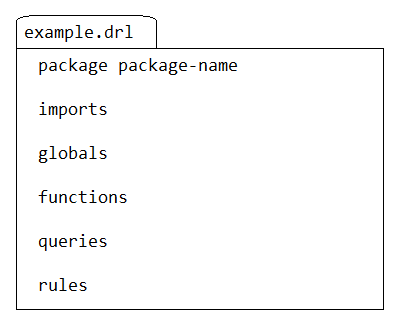
Слика 3.3 Структура правила [1]

*Attribute –* атрибути су опциони и означавају како правило треба да се понаша.

*Condition –* услови се постављају према правилима и служе за верификацију правила. Ако су дефинисани услови испуњени, правило се примењује, тј. реализују се акције дефинисане у *then* делу правила.

*Action –* пише се у стандардном *Java* коду и садржи акције које треба извршити уколико су испуњени услови.

Правила се пишу у фајлу који има .drl екстензију и структуру приказану на слици 3.4. Сви елементи су опциони.



Слика 3.4 Структура .drl фајла

## REST сервис

*Representational State Transfer (REST)* је стил софтверске арфитектуре заснован на постојању ресурса и униформног управљања њима преко скупа предефинисаих операција којег чине методе:

* *CREATE –* метода за креирање ресурса.
* *READ –* метода за читање ресурса.
* *UPDATE –* метода за измену ресурса.
* *DELETE –* метода за брисање ресурса.

REST сервис користи ресурсе чија је репрезентација у виртуалној форми у облику JSON објекта или неке друге репрезентације која одговара кориснику тих ресурса. Ти ресурси се трансферују из једне у другу апликацију.

REST дефинише шест архитектонских ограничења која треба да испуни сваки сервис да би био RESTful.

* Униформни интерфејс – како би било задовољено ово ограничење неопходно је испунити захтеве као што су: идентификација ресурса, манипулација ресурсима кроз њихову репрезентацију, самоописна порука као информација клијенту и хипертекст што значи да ће клијент након приступа ресурсу моћи да користи хиперлинкове како би могао да реализује све доступне акције карактеристичне за тај ресурс.
* Клијент – сервер као две независне апликације побољшавамо преносивост корисничког интерфејса на више платформи и на тај начин поједностављујемо компоненте сервера.
* *Stateless –* овај вид комуникације захтева да сваки захтев који долази од клијента на сервер мора да садржи све потребне информације за његову обраду. Стање сесије се у потпуности чува на страни клијента.
* Могућност кеширања – подаци у одговору или захтеву се имплицитно или експлицитно означавају као подаци који могу или не могу да се кеширају. То значи да је могућа поновна искористивост једног те истог одговора уколико корисник шаље еквивалентне захтеве.
* Слојевитост – начин организације компонената у хијерархије слојева, при чему свака компонента не може да интерагује са компонентама које су изван њеног непосредног слоја.
* Код на захтев (опционо) – REST омогућава проширење функционалности преузимањем и извршавањем кода у облику аплета или скрипти.

# СПЕЦИФИКАЦИЈА

Систем за куповину карата за скијалиште и препоруку осигурања на стази ради тако што обрачунава цену карата за унете критеријуме. Након што је корисник купио карте, он може да купи и осигурање које ће му бити предложено на основу његових захтева и здравствеог стања које је од значаја за скијаше. Његове функционалности биће представљене кроз различите *Unified Modeling Language (UML)* дијаграме.

## Дијаграм случајева коришћења

Дијаграми случајева коришћења илуструју на које све начине је могуће користити апликацију. На тај начин увиђамо сценарије у којима апликација интерагује са крајњим корисником, као и циљеве до којих ти сценарији могу да доведу.

Апликацију могу да користе три типа корсника, то су:

* Неулоговани корисник
* Улоговани корисник
* Администратор

Сва три типа корсника имају приступ странци за логовање и прегледу и претрази постојећих скијалишта и информацијама колика је попуњеност сваког од скијалишта за наредних 5 дана.

Права улогованог корисника су следећа:

* Куповина карата за конкретно скијалипте. Корисник уноси потребне информација као што су избор скијалишта, период коришћења, тип транспорта, дневни период корићења карте, уноси кориснике карата и да ли има права на неке погодности и попусте.
* Преглед свих купљених карата.
* Отказивање карата. Корисник може да откаже и уклони карте минимум два дана пре почетка важења карата.
* Куповина осигурања за кориснике карата. Корисник попуњава упитник у којем уноси податке неопходне за процену идеалне полисе која би задовољила његове критеријуме, способности и здравствено стање.
* Отказивање полиса осигурања. Корисник може да откаже купљене полисе минимум два дана пре почетка важења.

Права администратора су следећа:

* Додавање, измена, брисање и претрага скијалишта.
* Увид у посећеност скијалишта за конкретан датум.
* Претрага купљених полиса осигурања.
* Унос података везаних за искоришћење полисе осигурања.

## Дијаграм класа

## Дијаграм секвенци

ЈДКСАЈДЧ

# ИМПЛЕМЕНТАЦИЈА

Систем за куповину карата за скијалиште и препоруку осигурања састоји се из клијентске апликације имплементиране коришћењем *Angular* радног оквира, серверског дела уз помоћ *Spring-a* и система базираног на правилима који је имплеметиран употребом *Drools* алата. Код апликације налази се на линку: гитхуб.цом

## Клијентска апликација

## Серверска апликација

## *Drools* апликација

ЛИТЕРАТУРА

1. Drools rule structure - <https://www.researchgate.net/figure/Drools-Rule-Structure_fig1_330211452>, преузето августа 2021.