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

Тамара Глишић

**СИСТЕМ ЗА ПРОДАЈУ КАРАТА НА СКИЈАЛИШТУ И ПРЕПОРУКУ ОСИГУРАЊА**

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)

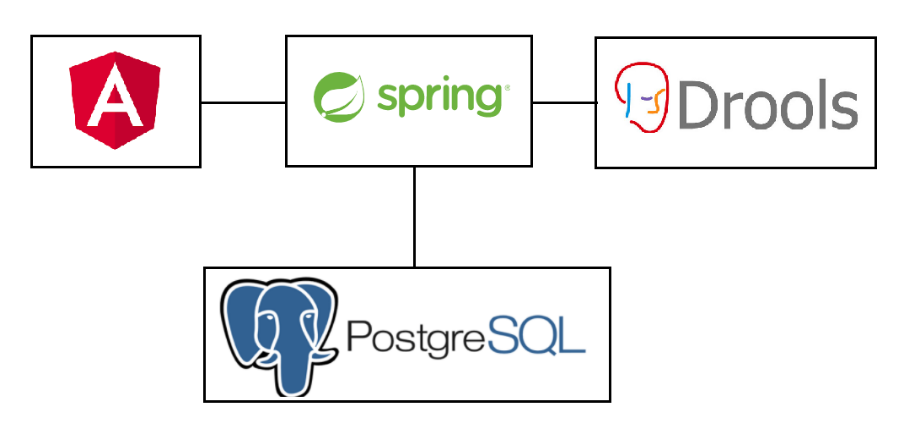
[3.4. REST сервис 7](#_Toc80882619)

# 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 . 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.

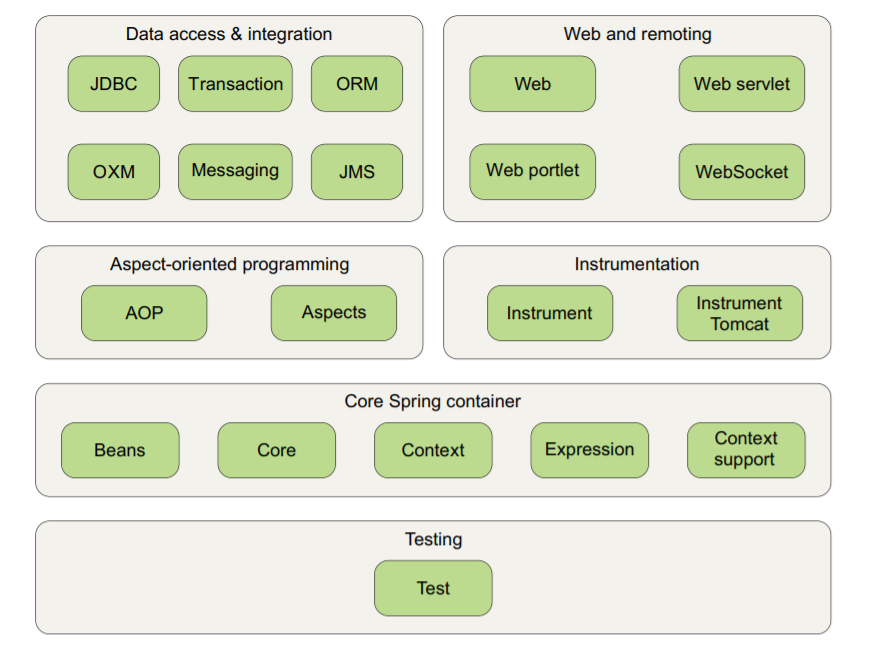
## Spring

*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 . 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.

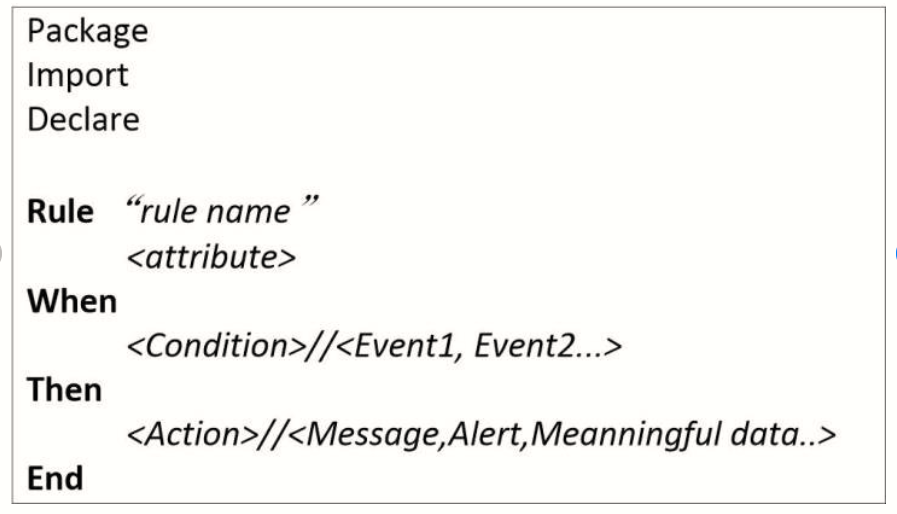
***Testing***

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.



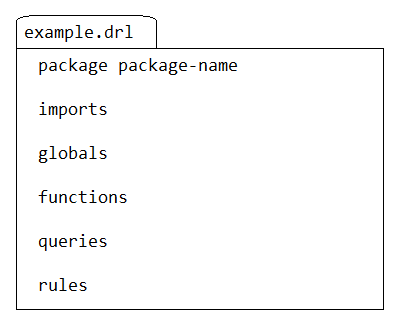
Слика . Структура правила [1]

*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 . 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.
* Преглед купљених полиса уз конкретну карту.
* 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.
* Корисник бира полису из листе предложених полиса осигурања. Изабрану полису може да купи само једном или да исту полису одмах купи и за све остале кориснике карата.
* 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

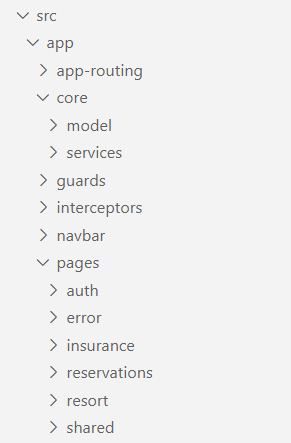
ЈДКСАЈДЧ

# 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

Клијентска апликација имплементирана је уз помоћ *Angular* радног оквира. Састоји се од коренског модула, модула за рутирање и још шест модула у који обухватају логички сродне компоненте. Структура апликације приказана је на слици 5.1.



Слика . Структура Angular апликације

*app – routing –* овде се налази модул за рутирање који импортује руте. Руте на које су намапиране компоненте, налазе се у фајлу *routes.ts.*

*core –* садржи модел и сервисе за комуникацију клијентске апликације са серверским делом.

*guards –* пакет у коме се налази *loginGuard* и *roleGuard* за проверу права приступа рутама. Поједине руте су доступне само улогованим корисницима, док су неке доступне само за конкретне улоге корисника, админа или регистрованог корисника.

*interceptors –* сервис за пресретање и измену захтева. Дешифрује токен који се налази у *localStorage* и шаље га заједно са захтевом као *Authorization* параметар.

*navbar –* у овом пакету груписани су навигацион менији. Саджи навигациони мени за неулогованог корисника, улогованог корисника и админа.

*auth –* модул саджи компоненту за логовање и компоненту за регистрацију.

*error –* модул у коме се налазе компоненте за приказ грешака уколико, на пример, корисник покуша да приступи страници за коју нема приступ.

*insurance –* модул који обједињује компонете за приказ, претрагу и измену полиса осигурања као и за попуњавање упитика за предлог полиса.

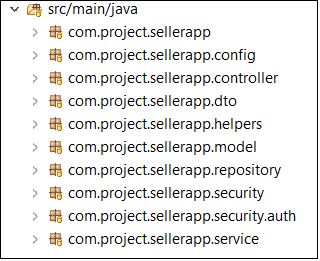
*reservations –* модул чине компонента за приказ резервација конкретног корисника и компонента за добављање неопходних информација за креирање нове резервације.

*resort –* у овом модулу груписане су компоненте за приказ, едитовање, додавање нових скијалишта, као и компонента за приказ капацитета скијалишта.

*shared –* овде се налази дијалог компонента за потврду акција. Компонента је прилагодљива различитим акцијама.

## Серверска апликација

Серверска апликација имплементирана је у *Java* програмском језику коришћењем *SpringBoot* радног оквира. Структура апликације (Слика 5.2) прати препоручене конвенције. Пакети групишу логичке целине.



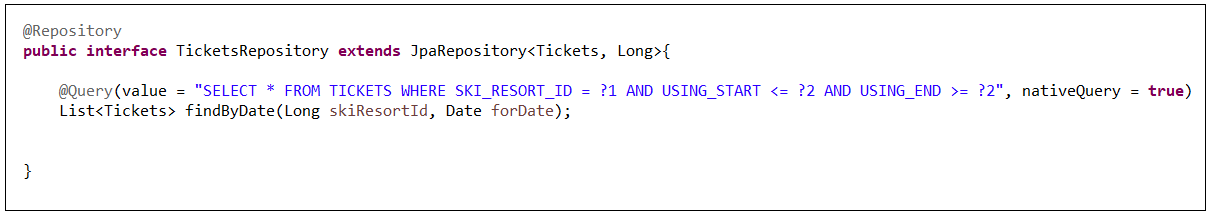
Слика . Структура серверске апликације

*controller –* обједињује класе контролера. Котролери су задужени за комуникацију између клијента и сервера. Сваки захтев који је упућен серверу пролази кроз контролер и кроз проверу права приступа. Контролери инјектују потребне сервисе и након пристизања захтева позивају потребне методе из сервиса. Након обрађеног захтева клијенту се шаље статус о успешности обрађеног захтава, и резултати обраде захтева. Овај пакет садржи следеће контролере:

* *AuthenticationController –* обрађује захтеве за логовање, регистрацију, и одјаву са система.
* *InsuranceCompanyController –* обрађује захтеве који се тичу осигуравајућих компанија.
* *PolicyController –* обрађује захтев за процесуирање података из упитника.
* *PurchasedPolicyController –* за обраду захтева који се извршују над купљеним полисама.
* *SkiResortController –* за обраду захтева за добављање, претрагу, измену, додавање нових скијалишта.
* *TicketsController –* обрађује захтеве везане за карте.

*service –* у овом пакету се налазе класе сервиса. Класе сервиса имплементирају пословну логику и инјектују потребне интерфејсе репозиторијума преко који се врши комуникација са базом. Поред тих сервиса, овај пакет садржи и сервис који активира и комуницира са правилима у *Drools* апликацији. То је сервис *KieService.* У њему се налазе методе за позивање правила везаних за куповину карата и препорку осигурања, као и за позивње *CEP (Complex Event Processing)* правила.

*repository –* садржи интерфејсе који директно комуницирају са базом. Ови интерфејси наслеђују *JpaRepository* [2] из *Spring* радног оквира. *JpaRepository* поред стандардних *CRUD (Create Update Delete)* операција, пружа подршку за додавање и нових, нестандардних операција над подацима. Специфичне операције не морају се имплемнтирати, довољно је у интерфејсу написати декларације метода пратећи конвенције у именовању. Нестандардни упити који се не могу на овај начин изразити, задају се у *JPQL* језику који је део *JPA* имплементације (Слика 5.3).



Слика . Пример нестандардног упита задатог у JPQL језику, проналази карте тако да задати датум упада у период коришћења карата

*config –* садржи класе неопходне за конфигурисање безбедности апликације. Конфигурација безбедности се врши редефинисањем метода класе *WebSecurityConfigurerAdapter.* Дефинишемо начин утврђивања идентитета корисника при аутентикацији, дефинишемо права приступа за захтеве ка одређеним *URL* – овима *(енгл. Uniform Resource Locator)* и дефинишемо генералну безбедност веб апликације, као што је игнорисање аутентикације за одређене захтеве идт.

*dto –* пакет који саджи *DTO (Data Transfer Object)* класе које се користе за пренос података између клијента и сервера. Ове класе моделују податке из базе.

*model –* садржи класе које моделују систем. Како су ове класе повезане између себе представњено је у поглављу.............\*\*\*

*helpers –* пакет који садржи классе за мапирање података из ентитета у *DTO* класе. Такође садржи и класу *Utility* са методама које манипулишу датумима, конвертујућих у *String,* или за рачунање нових датума неопходних за извршавање неких задатака.

*security –* класе за рад са токенима. У овом случају *JSON* веб токенима. Ту се налазе класе са методама за генерисање, преузимање токена из захтева и читање података из токена, класа која дефинише филтер који ће пресретати сваки захтев клијента упућен серверу и проверавати валидност токена.

## *Drools* апликација

*Drools* апликација чини засебну апликацију базирану на знању. У њој се налазе правила на основу којих систем опонаша експерта специфичне области и доноси одлуке у зависности од тих правила. Правила су подељена у логичке целине и налазе се у три пакета. Први пакет садржи правила за израчунавање цене карата, други пакет је пакет са правилима за препоруку осигурања и трећи пакет садржи *CEP* правила која служе за детектовање сумњивих активности или праћења догађаја од значаја, више о тим догађајима у наставку.

### Правила за израчунавање цене карте

Након уношења корисникових захтева на клијентској страни, подаци се обрађују, пролаазе кроз правила, и рачуна се најповољнија цена за његове карте. Клијентов захтев се састоји из:

* одабира скијалишта,
* одабира типа транспорта (може бити ски лифт, гондола или комбиновано),
* одабира периода (дневна карта, полудневна или ноћна),
* почетак / крај коришћења карте,
* уношење појединачно броја корисника карата. Корисници карата могу бити деца, одрасли или сениори.
* Навођење одређених привилегија за попусте (студентски попуст или *loyalty)*.
* вако сника.

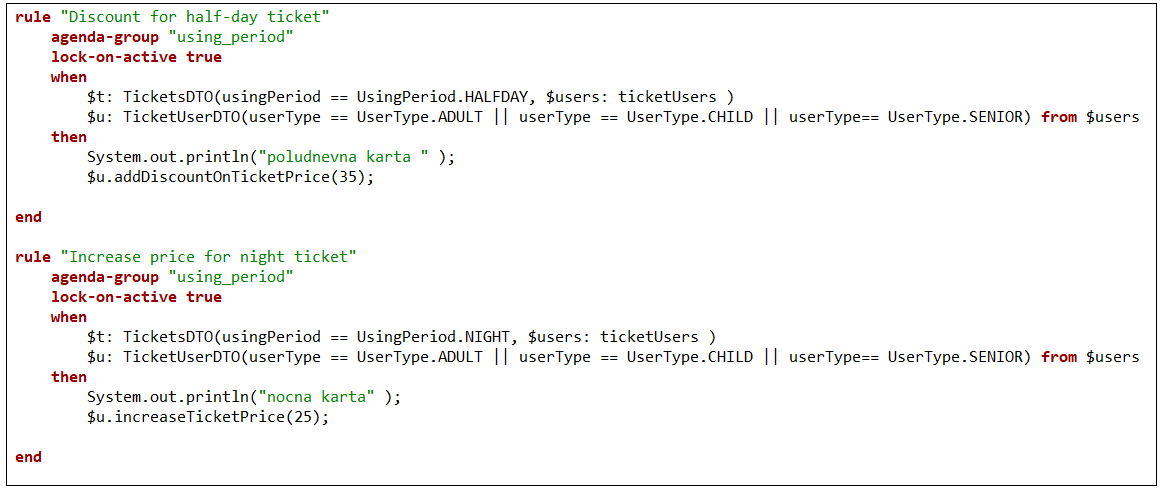
Правила су груписана у агенда групе*,* што нам омогућава да правила сврстамо у логички сродне групе и на тај начин ставимо фокус на извршавање одређене групе правила. Водећи се тиме, правила за израчунавање цене груписана су у следеће агенда групе:

* *transport\_type* (Слика 5.4):како свако скијалиште поседује иницијалну цену карте за коришћење ски лифта или гондоле, ова група правила на основу одабраног типа транспорта, сваком кориснику карата поставља почетну цену карте која ће се даље прилагођавати унетим захтевима.



Слика . Правила за одабран тип транспорта

* *using\_period:* цена карте је иницијално за дневне вожње. Уколико је корисник ипак одабрао полудневне или ноћне вожње, његова цена карте се умањује или повећава респективно. За полудневну карту цена је мања за 35%, док је за ноћну вожњу она скупља 25%.



Слика . Правила за одабрани период коришћења

* *type\_ticket* (Слика 5.6):систем генерише тип карте на основу броја корисника. Свака карта је иницјално појединачна. Да би карта била групна, мора да има бар 20 корисника, а да би била породична мора да задовољи услов да има бар два одрасла корисника и једно дете и да нема сениор корисника.



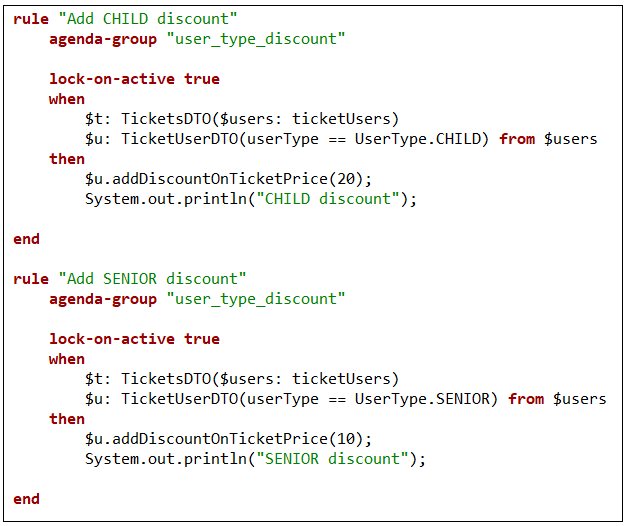
Слика . Правила која генеришу чињеницу типа карте

* *type\_ticket\_discount* (Слика 5.7): различити типови карата доносе одређене погодности, па тако на основу изгенерисане чињенице типа карте додељује се попуст. Уколко је карта групна, њена вредност се умањује 10%, а ако је породична 15%. Додатни попуст се остварује уколико породична карта има корисника са *loyalty* погодностима. Онда се одобрава још 15% попуста за сваког корисника.



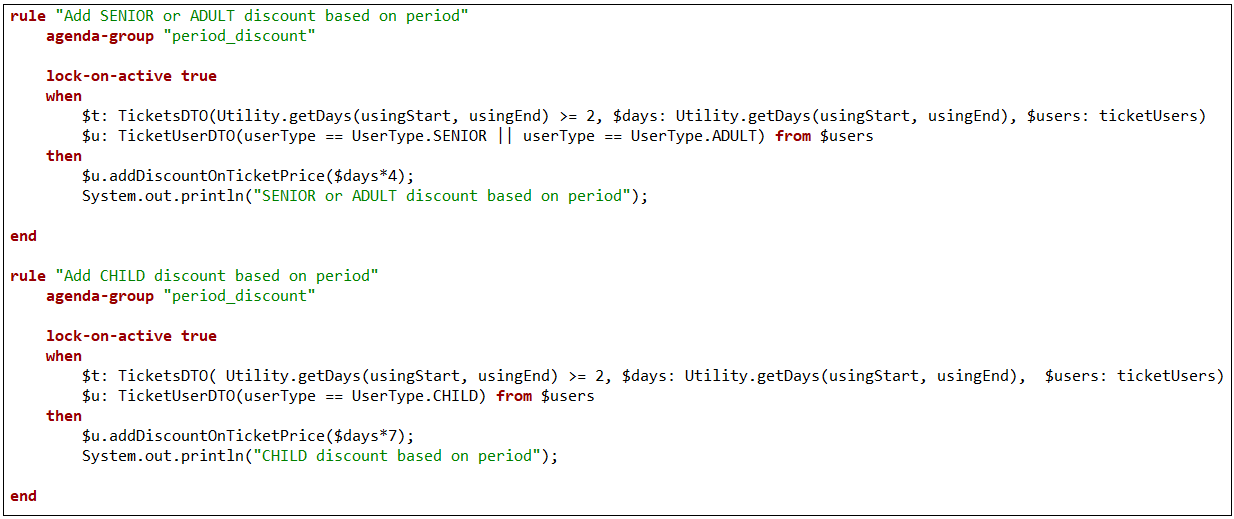
Слика . Правила за одобравање попуста на основу типа карте

* *user\_type\_discount* (Слика 5.8): иницијална цена карте односи се на корисника карте типа одрасли (енгл. *Adult*), с тога ова група правила умањује цену карте за 20% ако је корисник карте дете, или 10% ако је корисник сениор.



Слика . Правила за умањење цене на основу типа корисника

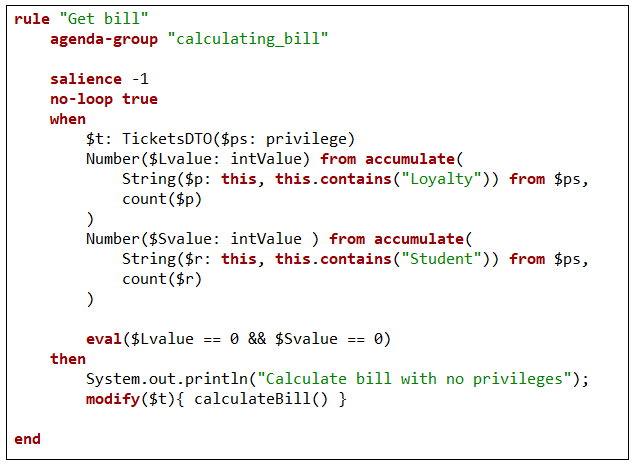
* *period\_discount* (Слика 5.8): уколико се карте купују за минимум два дана коришћења, њихова цена се умањује. Проценат умањења цене за сениоре и одрасле се рачуна по формули бр\_дана \* 4, а за децу бр\_дана \* 7.



Слика . Правила за попуст на дужину периода коришћења карте

* *calculating\_bill:* након што је формирана коначна цена карте која се односи на један дан, систем рачуна укупну цену за све дане коришћења. У овој групи правила утврђујемо да ли корисници карата имају одређене привилегије, тј. да ли су студенти или поседују *loyalty* погодности.

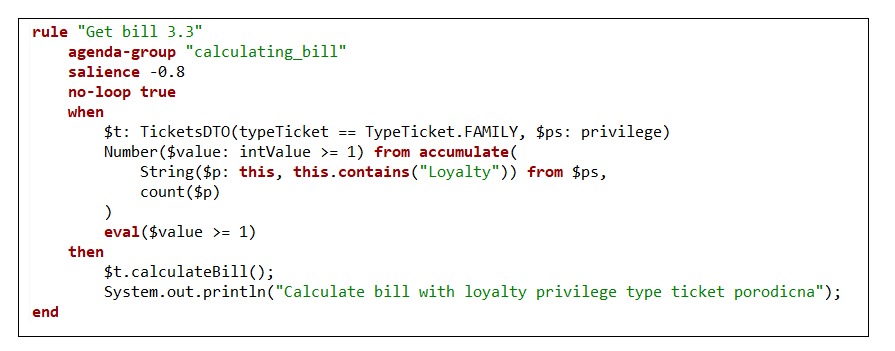
Уколико нема никаквих привилегија (Слика 5.10) укупна цена се рачуна на једноставан начин, позивањем методе *calculateBill()* која иницијалну цену карте за сваког корисника, која је претходно утврђена проласком кроз правила, помножи са бројем дана.



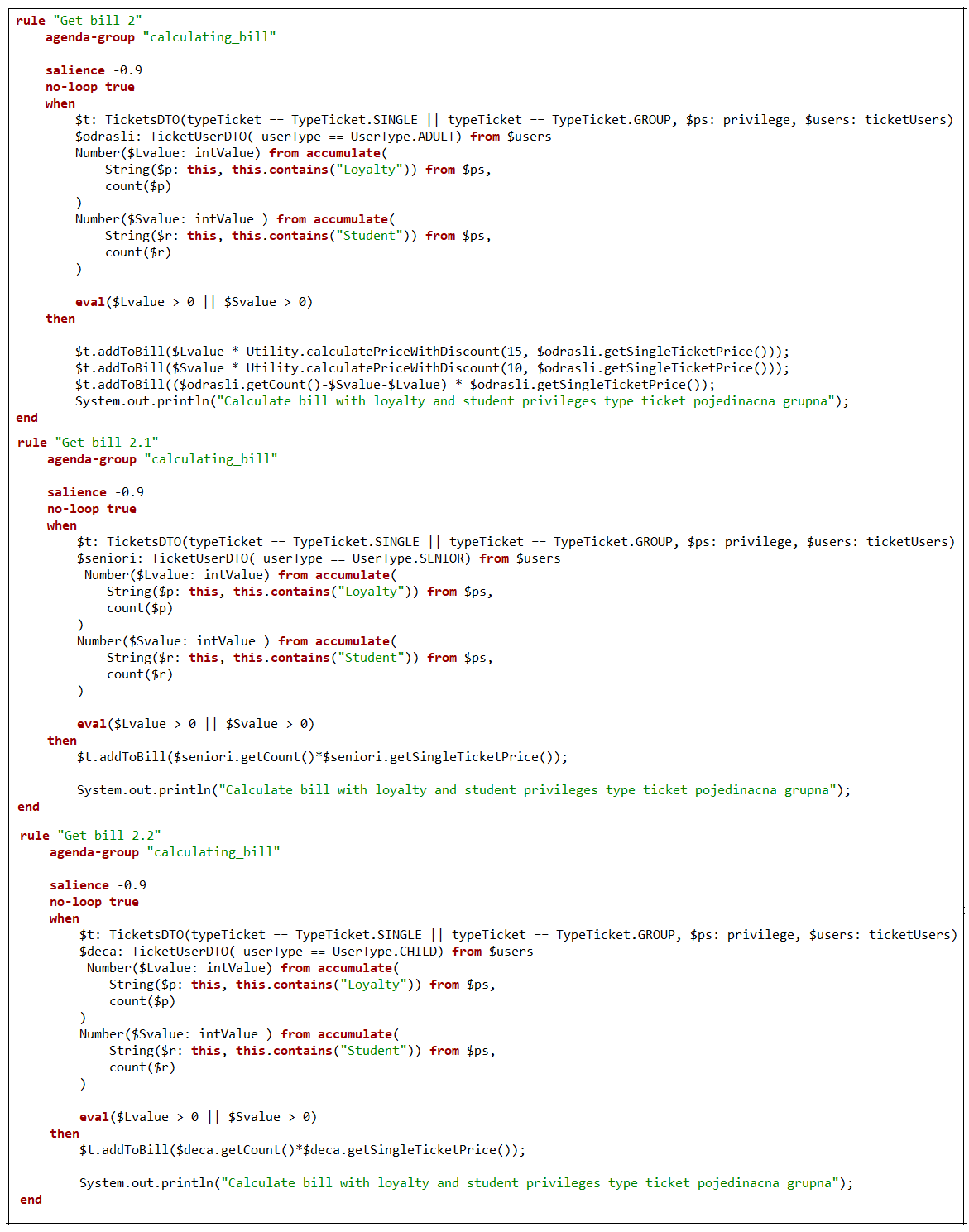
Слика . Рачунање укупне цене за кориснике без привилегија

Ако неко од корисника има одређене погодности, и ако је у питању појединачна или групна карта (Слика 5.12), то значи да ће само одређени корисници добити право на те погодности, тј. да ће попуст бити примењен на онај број карата колико постоји привилегија. Слично важи и уколико је тип карте породична, а неко од корисника има право на студентски попуст (Слика 5.13). Тада се умањује онолико иницијалних цена карата, чији су корисници типа одрали, колико има права на студентски попуст. Сениори и деца немају права на студентски попуст.

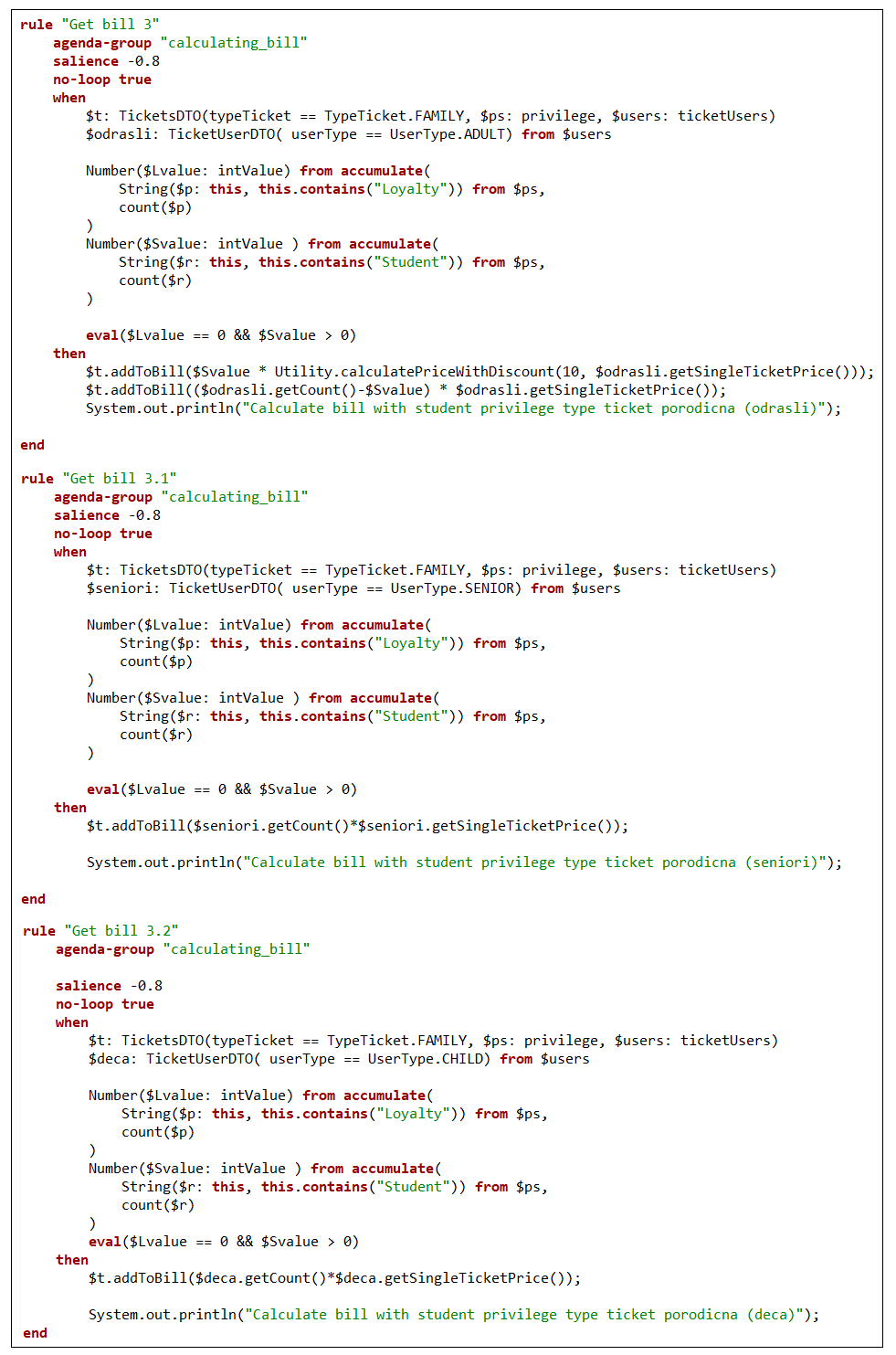
Изузетак је *loyalty* погодност када је породични тип карте. Тада се попуст обрачунава на крају и односи се на цео рачун. Правило за то приказано је на слици 5.11.



Слика . Правило за породични тип карте са loyalty привилегијама



Слика . Правила за рачунање укупне цене карата типа појединачна или групна са привилегијама



Слика . Правила за рачунање укупне цене карата типа породична са студентским попустом

* *regular\_guest:*
* *occupancy\_rate:*
* *bonus:*

### Правила за препоруку осигурања

### *CEP* правила

ЛИТЕРАТУРА

1. Drools rule structure - <https://www.researchgate.net/figure/Drools-Rule-Structure_fig1_330211452>, преузето августа 2021.
2. JpaRepository - <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.repositories>, преузето августа 2021.