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

Јелена Хрњак

***SECURItydsl* – наменски језик за подршку брзог успостављања конфигурације безбедносних аспеката у радном оквиру *spring***

Мастер рад

- Мастер академске студије -

Нови Сад, 2023.

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број, **РБР**: | |  | |
| Идентификациони број, **ИБР**: | |  | |
| Тип документације, **ТД**: | | Монографска документација | |
| Тип записа, **ТЗ**: | | Текстуални штампани материјал | |
| Врста рада, **ВР**: | | Мастер рад | |
| Аутор, **АУ**: | | Јелена Хрњак | |
| Ментор, **МН**: | | др Владимир Димитриески, доцент | |
| Наслов рада, **НР**: | | *SecurityDSL* – Наменски језик за подршку брзог успостављања конфигурације безбедносних аспеката у радном оквиру *Spring* | |
| Језик публикације, **ЈП**: | | Српски / ћирилица | |
| Језик извода, **ЈИ**: | | Српски | |
| Земља публиковања, **ЗП**: | | Република Србија | |
| Уже географско подручје, **УГП**: | | Војводина | |
| Година, **ГО**: | | 2023 | |
| Издавач, **ИЗ**: | | Ауторски репринт / Факултет техничких наука | |
| Место и адреса, **МА**: | | Нови Сад, Трг Доситеја Обрадовића 6 | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | | TODO | |
| Научна област, **НО**: | | Електротехничко и рачунарско инжењерство | |
| Научна дисциплина, **НД**: | | Примењене рачунарске науке и информатика | |
| Предметна одредница/Кqучне речи, **ПО**: | | Доменски оријентисано моделовање и језици | |
| **УДК** | |  | |
| Чува се, **ЧУ**: | | Библиотека Факултета техничких наука, Нови Сад | |
| Важна напомена, **ВН**: | |  | |
| Извод, **ИЗ**: | | TODO | |
| Датум прихватања теме, **ДП**: | |  | |
| Датум одбране, **ДО**: | |  | |
| Чланови комисије, **КО**: | Председник: |  |
|  | Члан: |  | Потпис ментора |
|  | Члан, ментор: | др Владимир Димитриески, доцент |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | Monographic publication | |
| Type of record, **TR**: | | Textual printed material | |
| Contents code, **CC**: | | Graduate-master Thesis | |
| Author, **AU**: | | Jelena Hrnjak | |
| Mentor, **MN**: | | Vladimir Dimitrieski, PhD, Assistant Professor | |
| Title, **TI**: | | SecurityDSL – A Domain-Specific Language for Supporting Rapid Configuration of Security Aspects in the Spring Framework | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | Serbian | |
| Country of publication, **CP**: | | Republic of Serbia | |
| Locality of publication, **LP**: | | Vojvodina | |
| Publication year, **PY**: | | 2023 | |
| Publisher, **PB**: | | Author’s reprint/Faculty of Technical Sciences | |
| Publication place, **PP**: | | Novi Sad, Faculty of Technical Sciences, Dositeja Obradovica sq. 6 | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | TODO | |
| Scientific field, **SF**: | | Electrical and computer engineering | |
| Scientific discipline, **SD**: | | Applied computer science and informatics | |
| Subject/Key words, **S**/**KW**: | | Domain-Specific Modeling Languages | |
| **UC** | |  | |
| Holding data, **HD**: | | The Library of Faculty of Technical Sciences, Novi Sad, Serbia | |
| Note, **N**: | |  | |
| Abstract, **AB**: | | TODO | |
| Accepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: |  |
|  | Member: |  | Menthor's sign |
|  | Member, Mentor: | Vladimir Dimitrieski, PhD, Assistant Professor |  |

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ДИПЛОМСКИ РАД** | Лист/Листова: |
|  |

| Студијски програм: | **Рачунарство и аутоматика** |
| --- | --- |
| Руководилац стидијског програма: | др Мирна Капетина, ванредни професор |

*(Податке уноси предметни наставник - ментор)*

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Јелена Хрњак** | Број индекса: | **Е2 64/2022** |
| Област: | **Примењене рачунарске науке и информатика** | | |
| Ментор: | **др Владимир Димитриески, доцент** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ДИПЛОМСКОГ РАДА:**

|  |
| --- |
| **Наменски језик за генерисање...** |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| * ... * ... |

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

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

**Садржај**

[1. Увод 1](#_Toc142355233)

[1.1 Структура рада 2](#_Toc142355234)

[2. Теоријске основе моделима вођеног развоја, наменских језика и безбедносних аспеката у радном оквиру *Spring* 3](#_Toc142355235)

[2.1 – поглавље пре овог 3](#_Toc142355236)

[2.2 Преглед безбедносних механизама у *Spring Boot* апликацијама – поглавље пре овог 3](#_Toc142355237)

[2.2.1 *Java* 3](#_Toc142355238)

[2.2.2 *Spring Boot* 3](#_Toc142355239)

[2.2.3 *Basic Authentication* 3](#_Toc142355240)

[2.2.4 Аутентификација помоћу *JSON Web* токена 3](#_Toc142355241)

[2.2.5 *OAuth2.0* 3](#_Toc142355242)

[2.2.6 *PostgreSQL* 3](#_Toc142355243)

[2.2.7 *MySQl* 3](#_Toc142355244)

[2.2.8 *Oracle* 3](#_Toc142355245)

[3. Преглед постојећег стања у области 4](#_Toc142355246)

[3.1 Преглед постојећих начина за моделовање веб апликација 4](#_Toc142355247)

[4. Наменски језик за подршку брзог успостављања конфигурације безбедносних аспеката у радном оквиру *Spring* 5](#_Toc142355248)

[4.1 Апстрактна синтакса 5](#_Toc142355249)

[4.1.1 Концепт *Application* 7](#_Toc142355250)

[4.1.2 Концепт Database 8](#_Toc142355251)

[4.1.3 Концепт *Attribute* 9](#_Toc142355252)

[4.1.4 Концепт *Entity* 10](#_Toc142355253)

[4.1.5 Концепт *User* 11](#_Toc142355254)

[4.1.6 Концепт *Role* 12](#_Toc142355255)

[4.1.7 Концепт *RoleInstance* 12](#_Toc142355256)

[4.1.8 Концепт *Endpoint* 13](#_Toc142355257)

[4.1.9 Концепт *Controller* 14](#_Toc142355258)

[4.1.10 Концепт *Authentication* 16](#_Toc142355259)

[4.1.11 Концепт *Security* 17](#_Toc142355260)

[4.1.12 Концепт *BasicAuthentication* 17](#_Toc142355261)

[4.1.13 Концепт *JWT* 17](#_Toc142355262)

[4.1.14 Концепт *Claim* 18](#_Toc142355263)

[4.1.15 Концепт *OAuth2* 20](#_Toc142355264)

[4.1.16 Концепт *Provider* 21](#_Toc142355265)

[4.2 Конкретна синтакса 22](#_Toc142355266)

[4.2.1 Граматика наменског језика *securityDSL* 22](#_Toc142355267)

[4.3 Примери модела описаних наменским језиком *securityDSL* 25](#_Toc142355268)

[5. Генерисање веб апликација са безбедносном конфигурацијом 29](#_Toc142355269)

[6. Закључак 30](#_Toc142355270)

[Скраћенице 31](#_Toc142355271)

[Литература 32](#_Toc142355272)

[Биографија 33](#_Toc142355273)

# Увод

Коришћење веб апликација представља неизоставни део свакодневног живота за велики број корисника, али корисници често нису свесни колико личних података те апликације прикупљају, обрађују и складиште и колико безбедносни пропусти могу да утичу на њих. С обзиром на осетљивост и важност података којима апликације неретко рукују, неовлашћен приступ би могао да доведе до безбедносних ризика, укључујући злоупотребу и нарушавање приватности корисника. Стога, обезбеђивање високог нивоа заштите корисника и њихових података представља кључан циљ приликом развоја сигурних веб апликација.

Имплементација жељеног нивоа аутентификације и ауторизације, представља основу ефикасне заштите података од потенцијалних напада, крађе и злоупотребе. Међутим, обезбеђивање одговарајуће заштите за апликације представља сложен и временски захтеван процес, те је самим тим подложан грешкама. С обзиром да се безбедоносни апекти апликације изнова конфигуришу при почетној имплементацији сваке апликације, поред тога што је сложен и дуготрајан, овакав посао постаје и репетативан.

Са циљем уклањања наведених недостатака, тежи се оптимизацији и аутоматизацији развоја безбедних веб апликација како би се елиминисали безбедносни пропусти и грешке, али и уштедело време потребно при имплементацији. Једно од могућих решења представља аутоматско генерисање почетне веб апликације са конфигурисаним безбедносним механизмима који задовољавају потребе и захтеве корисника. Аутоматизација значајно олакшава рад програмерима због уштеде времена, доприноси квалитету софтвера и минимизира шансе за грешку.

Циљ, ово је средство, начин на који постижем циљ овог рада је креирање модела сигурних веб апликација и генерисање кода на основу тог модела. Посебна пажња усмерена је ка убрзању конфигурисања безбедносних аспеката, али и развоја осталих целина веб апликације. За постизање овог циља креиран је наменски језик *Security Domain-Specific Language (securityDSL)* за моделовање апликација уз генераторе који модел трансформишу у извршиви код*.*

Како би моделовање веб апликација било могуће, неопходно је да *securityDSL* садржи концепте за моделовање свих елемената апликације. Основни елементи се односе на базу података, слој за репрезентацију података, обраду захтева корисника и безбедносну конфигурацију. Наменски језик *securityDSL* омогућава једноставну спецификацију свих параметара апликације помоћу синтаксе која је програмерима лако читљива и позната. На основу модела и наведених параметара, генератори генеришу извршиви код написан у програмском језику *Java [1]*, коришћењем развојног оквира *Spring [2]*. Овим имплементација сигурних веб апликација постаје једноставнија и ефикасниј,а уједно се смањује могућносг грешака у процесу развоја.

## Структура рада

Након уводног поглавља следи поглавње ,,Преглед постојећег стања у области” где је направљен осврт на постојећа решења и извршена анализа сличности и разлика у односу на наменски језик *securityDSL* и пратеће генераторе. У овом поглављу дате су теоријске основе моделом вођеног развоја и наменских језика, пружајући увид у постојеће стандарде.

Треће поглавље ,,Наменски језик за моделовање веб апликација“ обухвата детаљан опис делова наменског језика и концепата које овај језик садржи.

У поглављу ,,Генерисање веб апликација са безбедносном конфигурацијом“ описана је имплементација генератора који трансформишу модел у извршиви код. Детаљно су описани кораци генерисања апликације и конфигурације безбедносних аспеката.

Поглавље ,,Закључак“ садржи резултате истраживања и осврт на постигнућа у раду. Дате су препоруке за будућа унапређења и развој, као и могућност примене.

# Теоријске основе моделима вођеног развоја, наменских језика и безбедносних аспеката у радном оквиру *Spring*

## – поглавље пре овог

## Преглед безбедносних механизама у *Spring Boot* апликацијама – поглавље пре овог

### *Java*

### *Spring Boot*

### *Basic Authentication*

### Аутентификација помоћу *JSON Web* токена

### *OAuth2.0*

### *PostgreSQL*

### *MySQl*

### *Oracle*

# Преглед постојећег стања у области

## Преглед постојећих начина за моделовање веб апликација

# Наменски језик за подршку брзог успостављања конфигурације безбедносних аспеката у радном оквиру *Spring*

Да би наменски језик омогућио брзо и ефикасно успостављање конфигурације безбедносних аспеката у радном оквиру *Spring,* неопходно је да омогући моделовање свих неопходних концепата за иницијализацију веб апликације. Ови концепти сврставају се у пет главних целина. На почетку, параметри који се односне на саму апликацију, као што су њен артефакт и опис, што омогућава брзу конфигурацију апликације. Слој за репрезентацију података представља другу целину, где је могуће дефинисање ентитета који су од интереса за домен апликације, док следећа целина представља параметре базе података који омогућавају складиштење и руковање подацима везаних за ентитете. Слој за обраду захтева корисника је целина за себе и омогућава дефинисање контролера. Последња целина представља сигурносни слог који се односи на аутентификацију и контролу приступа корисника. Наменски језик *securityDSL* подржава конфигурацију три безбедностна механизма у радном оквиру *Spring*:

* основну аутентификацију (*Basic Authentication),*
* аутентификација помоћу *JSON Web* токена (*JWT*) и
* *Open Authorization* (*OAuth2.0*)*.*

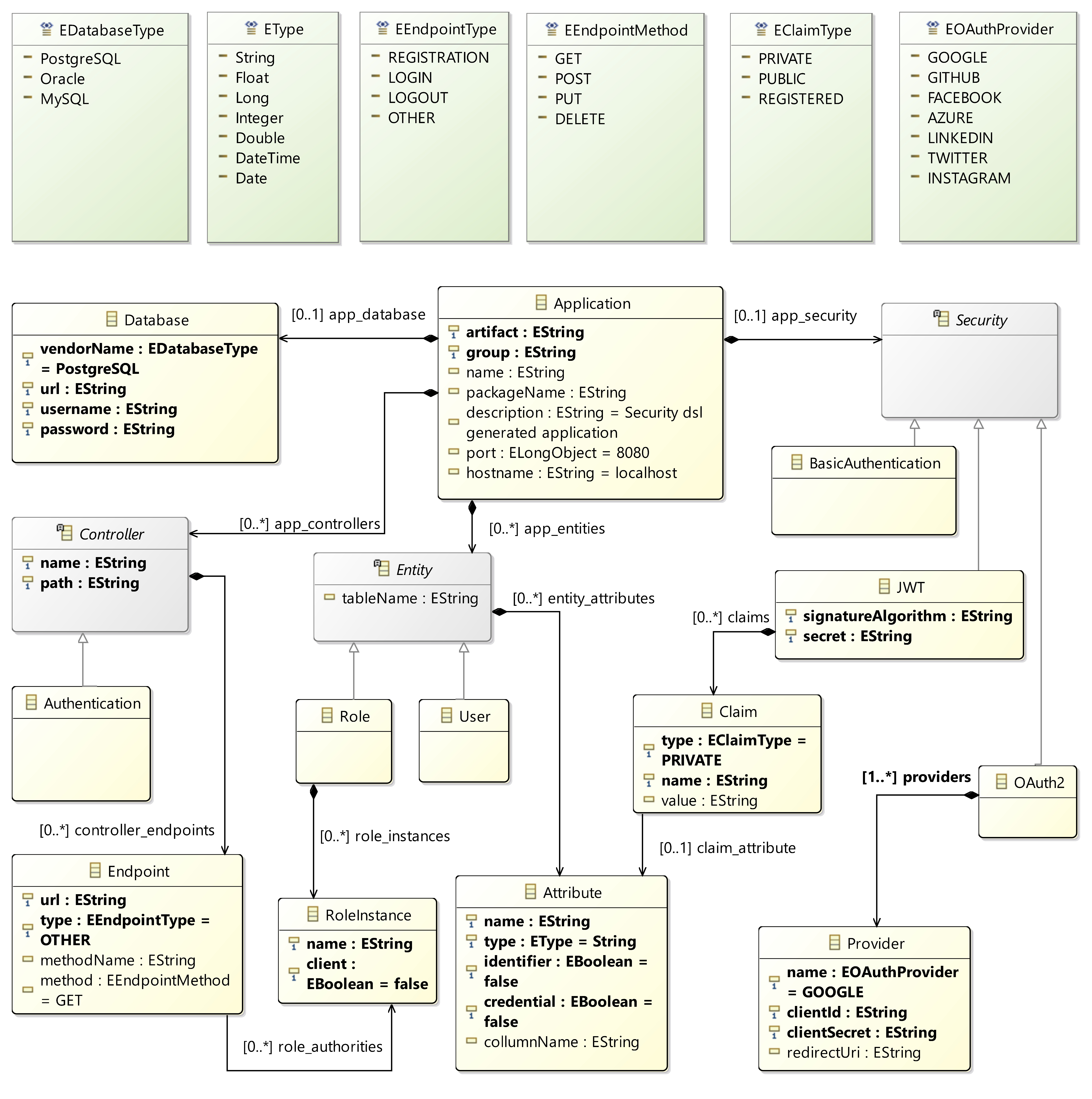
Овим се обезбеђује флексибилност и могућност одабира оптималног безбедносног механизма у зависности од потреба и захтева система.

Иако је примарна намена наменског језика *securityDSL* конфигурација безбедносних аспеката, корисницима је омогућена и конфигурација веб апликације без безбедносног слоја. Претходно наведене целине је могуће комбиновати уз ограничења која ће бити наведена у наставку поглавља. Коришћење *securityDSL* доменским експертима знатно убрзава и олакшава процес развоја сигурних веб апликација у радном оквиру *Spring.*

## Апстрактна синтакса

Апстрактна синтаска омогућава опис структуре наменског језика *securityDSL* и представљена је помоћу мета-модела (слика 4.1).

У даљем тексту дат је опис концепата апстрактне синтаксе где су енглески називи концепата који су приказани на слици наведени курзивом.



Слика 4.1 – Апстракнта синтакса наменског језика *securityDSL*

Коренски концепт апстрактне синтаксе је апликација (*Application*) и садржи податке о параметрима неоходним за иницијализацију апликације. Кориснику је остављена могућност проширења апликације додатним концептима. Могуће је подесити параметре за повезивање са базом података *(Database)*, а поред тога, могу се дефинисати ентитети (*Entity*) који се односе на кориснике (*User*) и роле (*Role*). При дефинисању ентитета, неопходно је навести обележја (*Attribute*) за сваки ентитет. Концепт који се односи на обраду захтева корисника (*Controller*) је повезан са концептом *Endpoint* где је остављена могућност додавања метода које се односе на различите функционалности апликације. Навођењем инстанци рола (*RoleInstance*) које ће се налазити у систему и повезивањем са одређеним *endpoint-*овима омогућена је контрола приступа. Контролер за аутентификацију (*Authentication*) може да садржи методе за регистрацију, пријаву на систем и одјаву са система.

Посебан део апстрактне синтаксе односи се на сигурносни слој (*Security*), где су подржана три безбедносна механизма: основна аутентификација (*BasicAuthentication*), аутентификација заснована на *JWT* токенима (*JWT*) и *OAuth2.0* аутентификација *(OAuth2).* У зависности од жељеног механизма могу се дефинисати додатни параметри описани адекватним концептима.

Сваки од наведених концепата могу се додати или изоставити у зависности од потреба корисника. Оваква апстрактна синтакса омогућава кориснику флексибилност и једноставно проширење генерисане апликације у складу са потребама.

У наставку поглавља описани су сви концепти апстрактне синтаксе наменског језика *securityDSL* уз опис обележја и асоцијација концепата. Поред тога, наведена су и појашњена ограничења имплементирана помоћу декларативног језика *OCL.*

### Концепт *Application*

Коренски концепт апстрактне синтаксе *Application* садржи обележја која се односе на основне параметре неопходне за иницијализацију апликације.

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *artifact* | *EString* | да | Назив артефакта |
| *group* | *EString* | да | Назив групе који представља организацију, компанију или тим који развија апликацију |
| *name* | *EString* | не | Назив апликације |
| *packageName* | *EString* | не | Назив пакета (*namespace)* у ком се налази изворни код апликације |
| *description* | *EString* | не | Опис апликације |
| *port* | *ELongObject* | не | Порт на ком апликација слуша захтеве |
| *hostname* | *EString* | не | Адреса рачунара или сервера на ком се извршава апликација |

Табела 4.1 – Обележја концепта *Application*

Портови у опсегу од 1024 до 49151 представљају регистроване портове који се могу доделити апликацији. Ограничење *validRegisteredPort* (листинг 4.1) гарантује да апликација користи валидан и регистрован порт.

**invariant** validRegisteredPort('Port must be in the valid range of

1024 to 49151!'):

**self**.*port* >= 1024 **and** **self**.*port* <= 49151;

Листинг 4.1 – Порт апликације мора бити унутар опсега регистрованих портова

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *app\_database* | *Database* | 0..1 | База података апликације |
| *app\_entitites* | *Entity* | 0..\* | Eнтитети у апликацији |
| *app\_controllers* | *Controller* | 0..\* | Контролери у апликацији |
| *app\_security* | *Security* | 0..1 | Безбедносна подешавања апликације |

Табела 4.2 – Асоцијације концепта *Application*

Апликација може садржати ентитете, контролере, базу података и аспекте безбедности. Кориснику је остављен простор да наведе или изостави одређене целине. Међутим, навођење неких целина повлачи одређена ограничења. Уколико се наведу ентитет, неопходно је да апликација има дефинисану базу података која складишти податке о тим ентитетима. Ово је омогућено *OCL* ограничењем *hasDatabawseForEntity* (листинг 4.2).

Листинг 4.2 – Уколико апликација има дефинисане ентитете, мора да има дефинисану базу података

**invariant** hasDatabaseForEntity('An application must have a database defined if it has entities!'):

**self**.*app\_entities* -> *isEmpty*() **or** **not** **self**.*app\_database*->*isEmpty*();

### Концепт Database

Апстрактни концепт *Database* представља параметре за повезивање апликације са базом података. Тип података *EDatabaseType* представља енумерацију заназив компаније која је развила жељени систем за управљање базама података. Корисник је дужан да унесе валидне креденцијале и конекциони стринг како би апликација успешно успоставила везу са базом података. Ово омогућава да апликација чита, додаје и ажурира податке сачуване у бази података.

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *vendorName* | *EDatabaseType* | да | Систем за управљање базама података, при чему су могуће вредности *PostgreSQL, MySQL* и *Oracle* |
| *url* | *EString* | да | Конекциони стринг за повезивање са базом података |
| *username* | *EString* | да | Корисничко име за приступ бази података |
| *password* | *EString* | да | Лозинка за приступ бази података |

Табела 4.3 – Обележја концепта *Database*

### Концепт *Attribute*

Концепт *Attribute* моделује обележја која описују различите карактеристике ентитета. *EType* представља енумерацију за тип податка обележја моделованог концептом *Attribute.* Атрибут представља идентификатор ентитета уколико је вредност обележја *identifier* једнака *true,* док је атрибут креденцијал уколико је вредност обележја *credential* једнака *true.*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *name* | *EString* | да | Назив обележја |
| *type* | *EType* | да | Тип обележја, при чему су могуће вредности *String, Float, Long, Integer, Double, DateTime* и *Date* |
| *identifier* | *EBoolean* | да | Да ли је обележје идентификатор |
| *credential* | *EBoolean* | да | Да ли је обележје креденцијал |
| *collumnName* | *EString* | не | Назив колоне у табели базе података који се односи на обележје |

Табела 4.4 – Обележја концепта *Attribute*

### Концепт *Entity*

Ентитети апликације моделовани су помоћу концепта *Entity.* Асоцијација ентитета са концептом *Attribute* омогућава да ентитети садрже сва релевантна обележја за домен апликације*.*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *tableName* | *EString* | не | Назив табеле у бази података која се односи на ентитет |

Табела 4.5 – Обележја концепта *Entity*

Ограничење *uniqueTableName* (листинг 4.3) гарантује да сви ентиети имају јединствен назив табеле у бази података што спречава потенцијалне проблеме приликом рада са подацима. За ентитете за које није наведен, назив табеле биће изједначен са називом ентитета у множини (*users* или *roles*).

**invariant** uniqueTableName('Table names must be unique!'):

*Entity*.*allInstances*() ->

*select*(e | *e*.*tableName* <> **null**) -> *isUnique*(e | *e*.*tableName*.*toLower*());

Листинг 4.3 – Сви ентитети имају јединствен назив табеле у бази података

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *entity\_attributes* | *Attribute* | 0..\* | Обележја ентитета |

Табела 4.6 – Асоцијације концепта *Entity*

Асоцијација *entity\_attributes* моделује придруживање одређених атрибута ентитету. Јединственост назива атрибута у оквиру ентитета омогућена је ограничењем  *uniqueAttributeName* (листинг 4.4), док јединственост назива колона у табели, гарантује ограничењне *uniqueCollumnName* (листинг 4.5). Неопходно је ентитети да поседују тачно један идентификатор, што је обезбеђено ограничењем onlyOneIdentifier (листинг 4.6).

**invariant** uniqueAttributeName('Attribute names within an entity must be unique!'):

**self**.*entity\_attributes* -> *isUnique*(a | *a*.*name*);

Листинг 4.4 – Сви атрибути у окциру ентитета имају јединствене називе

**invariant** uniqueCollumnName('Column names must be unique if defined!'):

**self**.*entity\_attributes* -> *exists*(a | *a*.*collumnName* <> **null**) **implies**

**self**.*entity\_attributes* -> *isUnique*(a | *a*.*collumnName*);

Листинг 4.5 – Сви атрибути имају јениствене називе колона унутар табеле у бази података

Листинг 4.6 – За ентитет мора да постоји тачно један атрибут који је идентификатор

**invariant** onlyOneIdentifier('Entity must have exactly one identifier attribute!'):

**self**.*entity\_attributes* -> *size*() > 0 **implies**

**self**.*entity\_attributes* -> *select*(a | *a*.*identifier*) -> *size*() = 1;

### Концепт *User*

Приликом генерисања кода, концепт *User* биће искоришћен за генерисање класе које представља ентитет корисника апликације. Наслеђује концепт *Entity,* те садржи обележја која ближе описују корисника. Могуће је постојање највише једне инстанце концепта *User* за исправну конфигурацију апликације (листинг 4.7)

**invariant** uniqueUserEntity('There can be at most one entity of type "User" in the model!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*User*)) -> *size*() <= 1;

Листинг 4.7 – Могуће је постојање највише једне инстанце концепта *User*

Атрибут који представља лозинку је подразумеван и биће генерисан са остатком кода, те не постоји потреба за експлицитним навођењем истог (листинг 4.8). Како би аутентификација била омогућена неопходно је да класа *User*, поред постојеће лозинке, поседује још тачно један атрибут који представља креденцијал (нпр. корисничко име). Ради једноставније провере креденцијала он мора бити типа *String.* Ово је гарантовано ограничењем *oneStringTypeCredentialForUser* (листинг 4.9). Увођењем креденцијала, потребно је онемогућити их у класама које не представљају кориснике система (листинг 4.10).

**invariant** noAttributeNamedPassword('User entity cannot have an attribute

named "password"!'):

**self**.*entity\_attributes* -> *forAll*(a | *a*.*name*.*toLower*() <> 'password');

Листинг 4.8 – За концепт *User* не сме да постоји атрибут који представља лозинку

**invariant** oneStringTypeCredentialForUser('User entity must have exactly one attribute of type String marked as a credential!'):

**self**.*entity\_attributes* -> *select*(a | *a*.*credential*) -> *size*() = 1

**and**

**self**.*entity\_attributes* ->

*select*(a | *a*.*credential*) -> *forAll*(a | *a*.*type* = *EType*::*String*);

Листинг 4.9 – За концепт *User* мора да постоји тачно један атрибут типа *String* који представља креденцијал

Листинг 4.10 – Само концепти *User* могу да поседују атрибут који представља креденцијал

**invariant** otherEntitiesDoesntHaveCredential('Entities other than User

cannot have a credential attribute!'):

**not** **self**.*oclIsTypeOf*(*User*) **implies** **self**.*entity\_attributes* -> *select*(a | *a*.*credential*) -> *isEmpty*();

### Концепт *Role*

Концепт *Role* наслеђује концепт *Entity* и моделује ентитет који се односи на роле апликације. Роле могу бити имплементиране као класе са различитим обележјима или као енумерација, што зависи од изабране безбедносне конфигурације у оквиру апликације. Могуће је постојање највише једне инстанце концепта *Role* (листинг 4.11).

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *role\_instances* | *RoleInstance* | 0..\* | Инстанце роли у апликацији |

Табела 4.7 – Асоцијације концепта *Role*

**invariant** uniqueRoleEntity('There can be at most one entity of type "Role" in the model!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*Role*)) -> *size*() <= 1;

Листинг 4.11 – Могуће је постојање највише једне инстанце концепта *Role*

Дефинисано је ограничење *uniqueRoleInstanceName* (листинг 4.12), које осигурава да свака инстанца роле има уникатан назив. Ово ограничење омогућава избегавање конфликата приликом додавања нових рола.

**invariant** uniqueRoleInstanceName('Role instance names must be unique!'):

**self**.*role\_instances* -> *isUnique*(r | *r*.*name*);

Листинг 4.12 – Инстанце рола морају имати јединствен назив

### Концепт *RoleInstance*

Инстанце рола, било да је у питању енумерација или ентитет са обележјима, моделоване су помоћу концепта *RoleInstance*. Представљају роле које је могуће доделити корисницима система како би била омогућена контрола приступа ресурсима апликације.

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *name* | *EString* | да | Назив конретне роле |
| *client* | *EBoolean* | да | Ознака да ли је рола клијент. Уколико је вредност обележја *false* рола је администраторска |

Табела 4.8 – Обележја концепта *RoleInstance*

### Концепт *Endpoint*

*Endpoint* је концепт који представља тачку комуникације између корисника и апликације, што га чини битним елементом за контролу приступа, односно саму безбедност апликације. Корисници комуницирају са сервером путем *endpointa* шаљући захтеве за извршавање одређених функционалности. Асоцијација *role\_authorities* описује роле којима је дозвољен приступ *endpoint*-у, односно роле које имају овлашћење да приступе одређеној функционалности систмеа.

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *url* | *EString* | да | Путања, односно *URL* адреса *endpoint-*а, |
| *type* | *EEndpointType* | да | Тип *endpoint*-a, где су могуће вредности регистрација (*REGISTRATION*), пријављивање (*LOGIN*), одјављивање (*LOGOUT*) и друго (*OTHER*) |
| *methodName* | *EString* | да | Назив методе на коју се односи *endpoint* |
| *method* | *EEndpointMethod* | не | *HTTP* метод који означава каква је врста захтева, при чему су могуће вредности *GET, POST, PUT* и *DELETE* |

Табела 4.9 – Обележја концепта *Endpoint*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *role\_authorities* | *RoleInstance* | 0..\* | Роле које имају право приступа *endpoint*-у |

Табела 4.10 – Асоцијације концепта *Endpoint*

Ограничење *urlStartsWithForwardSlash* (листинг 4.13) гарантује да путање *endpoint*-а започињу карактером ‘/’ што доприноси конзистентности апликације. Поред овог ограничења, битно је да роле којима је дозвољен приступ буду јединствене у оквиру *endpoint*-a (листинг 4.14).

**invariant** urlStartsWithForwardSlash('Endpoint URL should start with a forward

slash!'): **self**.*url*.*at*(1) = '/';

Листинг 4.13 – Путање *endpoint*-а започињу карактером ‘/’

Листинг 4.14 – Роле којима је додељен приступ *endpoint*-у не могу да се дуплирају унутар истог

**invariant** uniqueRoleAuthorities('Role authorities must be unique for each endpoint!'):

**self**.*role\_authorities* -> *isUnique*(r | *r*.*name*);

### Концепт *Controller*

Концепт *Controller* описује контролере апликације. Садржи информације о називу и путањи контролера, при чему је неопходно да обе вредности буду јединствене унутар апликације (листинг 4.15 и листинг 4.16). Уобичајено је да се називи контролера разликују од путања како би се избегли конфликти приликом рутирања захтева што је гарантовано ограничењем *uniqueControllerPath*. Путања контролера представља апсолутну путању у оквиру апликације, те је неопходно да почиње карактером '/'. Ово такође омогућава конзистентност генерисаног кода (листинг 4.17). Ентитети који се односе на кориснике (*User*) и роле (*Role*) доводе до постојања класа са истим називима, па самим тим постоји ограничење назива контролера (листинг 4.18).

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *name* | *EString* | да | Назив контролера |
| *path* | *EString* | да | Путања контролера |

Табела 4.11 – Обележја концепта *Controller*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *controller\_endpoints* | *Endpoint* | 0..\* | *Endpoint-*ови контролера |

Табела 4.12 – Асоцијације концепта *Controller*

**invariant** uniqueControllerName('Controllers should have unique names!'):

*Controller*.*allInstances*() -> *isUnique*(c | *c*.*name*);

Листинг 4.15 – Називи контролера унутар апликацјие морају да будујединствени

**invriant** uniqueControllerPath('Controller paths should be unique and

different from names!'):

*Controller*.*allInstances*() -> *isUnique*(c | *c*.*path*)

**and**

*Controller*.*allInstances*() ->

*forAll*(c | '/' + *c*.*name*.*toLower*() <> *c*.*path*.*toLower*());

Листинг 4.16 - Путање контролера унутар апликације морају да буду јединствени и да се разлику од назива контролера

**invariant** controllerPath('Controller path should start with \'/\'!'):

**self**.*path*.*at*(1) = '/';

Листинг 4.17 – Путања контролера мора да почиње караткером ‘/’

Листинг 4.18 – Ограничење назива за контролере

**invariant** controllerNotNamedUserRole('Controller names cannot be "User" or "Role"!'):

**not** *Controller*.*allInstances*() ->

*exists*(c | *c*.*name*.*toLower*() = 'user' **or** *c*.*name*.*toLower*() = 'role');

За правилно рутирање захтева и рад апликације неопходна је јединственост назива метода на које се односи *endpoint* (листинг 4.19)и путање *endpoint*-а(листинг 4.20) унутар контролера. *Endpoint*-овиза регистрацију, пријаву и одјаву са система имају препоручене, унапред дефинисане *HTTP* методе: *POST, POST* и *GET* редом. За *endpoint-*ове типа *OTHER* неопходно је навести метод (листинг 4.21). Регистрација, пријава и одјава са система су функционалности контролера за аутентификацију, те се ове врсте *endpoint-*ова не могу наћи у друдим конторлерима (листинг 4.22).

**invariant** uniqueEndpointMethodNames('Endpoints within a controller must have unique method names!'):

**self**.*controller\_endpoints* -> *isUnique*(e | *e*.*methodName*);

Листинг 4.19 – *Endpoint-*ови унутар контролера морају имају јединствене називе метода

**invariant** uniqueEndpointURLs('Endpoints within a controller must have unique URLs!'):

**self**.*controller\_endpoints* -> *isUnique*(e | *e*.*url*);

Листинг 4.20 – *Endpoint-*ови унутар контролера морају имају јединствене путање

**invariant** methodRequiredForOtherType('Endpoints of type "OTHER" must have a defined method!'):

**self**.*controller\_endpoints* -> *select*(e | *e*.*type* = *EEndpointType*::*OTHER*) -> *forAll*(e | *e*.*method* <> **null**);

Листинг 4.21 – *Endpoint-*ови типа *OTHER* морају да имају дефинисан *HTTP* метод

**invariant** endpointLimits('Controllers of type other than "Authentication" should not have registration, login, or logout endpoints!'):

**not** **self**.*oclIsTypeOf*(*Authentication*) **implies** (

**self**.*controller\_endpoints* ->

*select*(e | *e*.*type* = *EEndpointType*::*REGISTRATION*) -> *isEmpty*()

**and**

**self**.*controller\_endpoints* ->

*select*(e | *e*.*type* = *EEndpointType*::*LOGIN*) -> *isEmpty*()

**and**

**self**.*controller\_endpoints* ->

*select*(e | *e*.*type* = *EEndpointType*::*LOGOUT*) -> *isEmpty*());

Листинг 4.22 – *Endpoint-*ови за регистрацију, пријаву и одјаву са система могу да се налазе само у котнролеру за аутентификацију

### Концепт *Authentication*

Концепт *Authenticatiion* наслеђује *Controller.* Контролер за аутентификацију обрађује захтеве који се односе на регистрацију корисника, пријаву и одјаву корисника са система, те су информације о корисницима и њиховим ролама неопходни. Самим тим, уколико постоји контролер за аутентификацију, неопходно је да постоје инстанце концепата *User* и *Role* (листинг 4.23). Могуће је постојање највише једног контролера за аутентификацију (листинг 4.24). Уколико контролер за аутентификацију постоји, дозвољено је постојање највише једног *endpoint-*a за регистрацију, пријаву и одјаву са система што спречава вишеструко дефинисање истих функционалности (листинг 4.25).

Листинг 4.23 – За дефинисан контролер за аутентификацију, неопходно је постојање инстанци концепата User и Role

**invariant** hasUserAndRoleForController ('Authentication controller requires at least one User entity and one Role entity!'):

*Entity*.*allInstances*() -> *exists*(e | *e*.*oclIsTypeOf*(*User*))

**and**

*Entity*.*allInstances*() -> *exists*(e | *e*.*oclIsTypeOf*(*Role*));

Листинг 4.24 – Могуће је постојање највише једног контролера за аутентификацију

**invariant** uniqueAuthenticationController('There can be at most one controller of type "Authentication" in the model!'):

*Controller*.*allInstances*() ->

*select*(c |*c*.*oclIsTypeOf*(*Authentication*)) -> *size*() <= 1;

Листинг 4.25 – Могуће је постојање највише једног *endpoint-*а за регистрацију, пријаву и одјаву са система

**invariant** authenticationLimits('Authentication can have at most one registration, login, and logout endpoint!'):

**self**.*controller\_endpoints* ->

*select*(e | *e*.*type* = *EEndpointType*::*REGISTRATION*) -> *size*() <= 1

**and**

**self**.*controller\_endpoints* ->

*select*(e | *e*.*type* = *EEndpointType*::*LOGIN*) -> *size*() <= 1

**and**

**self**.*controller\_endpoints* ->

*select*(e | *e*.*type* = *EEndpointType*::*LOGOUT*) -> *size*() <= 1;

### Концепт *Security*

Концепт *Security* представља важну апстракцију у моделу која је кључна за обезбеђивање сигурне апликације. Овај концепт омогућава имплементацију жељеног нивоа аутентификације и ауторизације, што представља основу ефикасне заштите података од потенцијалних напада и злоупотребе.

### Концепт *BasicAuthentication*

Концепт *BasicAuthentication* наслеђује *Security* и моделује основну безбедносну конфигурацију. У случају одабира основне аутентификације роле ће у апликацији бити представљене као енумерација. Самим тим нису дозвољена обележја, већ само инстанце ролa (листинг 4.26)*.*

**invariant** basicAuthNoRoleAttributes('Basic authentication cannot have role attributes!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*Role*))

> *forAll*(role | *role*.*entity\_attributes* -> *size*() = 0)

Листинг 4.26 – У случају основне аутентификације нису дозвољена придружена обележја конепту *Role*

### Концепт *JWT*

Аутентификацију помоћу *JSON Web* токена моделована је концептом *JWT* који наслеђује концепт *Security.*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *signatureAlgorithm* | *EString* | да | Алгоритам који се користи за потписивање *JWT* токена |
| *secret* | *EString* | не | Тајни кључ који се користи за потписивање *JWT* токена |

Табела 4.13 – Обележја концепта *JWT*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *claims* | *Claim* | 0..\* | Тврдње у оквиру *JWT* токена |

Табела 4.14 – Асоцијације концепта *JWT*

За аутентификацију помоћу токена *JWT* роле представљају класу са обележјима. Како би инстанце ролa биле омогућене, неопходно је да у случају одабира аутентификације на основу *JWT* токена за класу *Role* постоји тачно једно обележје које је типа *String.* Поред тога, дозвољено је само обележје које представља идентификатор. У случају да је идентификатор типа *String,* нису дозвољена додатна обележја. Ово је гарантовано обележјима *roleHasMaxTwoAttributes* и *roleHasStringAttribute* (листинг 4.27).

**invariant** roleHasMaxTwoAttributes ('Role entities can have at most two attributes!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*Role*)) ->

*forAll*(role | *role*.*entity\_attributes* -> *size*() <= 2);

**invariant** roleHasStringAttribute('Role entities must have either one identifier attribute of type String or both identifier and non-identifier attributes of type String!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*Role*)) ->

*forAll*(role | (*role*.*entity\_attributes* -> *select*(a | *a*.*identifier*

**and** *a*.*type* = *EType*::*\_'String'*) -> *size*() = 1

**and** *role*.*entity\_attributes* ->

*select*(a | *a*.*type* = *EType*::*\_'String'*) -> *size*() = 1)

**or** (*role*.*entity\_attributes* -> *select*(a | *a*.*identifier*

**and** *a*.*type* <> *EType*::*\_'String'*) -> *size*() = 1

**and** *role*.*entity\_attributes* ->

*select*(a | *a*.*type* = *EType*::*\_'String'*) -> *size*() = 1));

Листинг 4.27 – У случају аутентификације помоћу *JWT* токена*,* рола може да има тачно једно обележје типа *String*

### Концепт *Claim*

Концепт *Claim* представља трврдње које садржи *JWT* токен, односно податке о кориснику, стању апликацје или самом токену које се преносе путем токена. Постоје три типа тврдњи: регистроване (предефинисане), приватне и јавне. Поред типа, тврдња има назив који представља идентификаотр пода тка и вредност која се односи на дати назив. Уколико се тврдња односи на податке о кориснику, она је повезана са атрибутом који садржи њену вредност. Све тврдње морају имати јединствен назив (листинг 4.28) и за један атрибут може бити везана највише једна тврдња (листинг 4.29)

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *type* | *EClaimType* | да | Тип тврдње, при чему су могуће вредности *PRIVATE, PUBLIC* и *REGISTERED* |
| *name* | *EString* | да | Назив трвдње |
| *value* | *EString* | не | Вредност тврдње |

Табела 4.16 – Обележја концепта *Claim*

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *claim\_attributes* | *Attribute* | 0..1 | Атрибут на ког се тврдња односи и који садржи додатне информације о њој |

Табела 4.17 – Асоцијације концепта *Claim*

**invariant** uniqueClaimNames('Claims must have unique names'):

*Claim*.*allInstances*() -> *isUnique*(c | *c*.*name*);

Листинг 4.28 – Тврдње морају имати јединствен назив

**invariant** uniqueClaimAttribute('Claim attributes must be unique!'):

*Claim*.*allInstances*() -> *select*(c | *c*.*claim\_attribute* <> **null**) ->

*isUnique*(c | *c*.*claim\_attribute*);

Листинг 4.29 – Атрибут може бити повезан са највише једном тврдњом

Основне регистроване, односно предефинисане трврње су време важења токена након ког он више није валидан (*expirationTime),* корисници или субјекти којима је токен намењен (*audience*)*,* издавач токена (*issuer)* и идентификатор корисника (*subject*) те оне морају бити типа *REGISTERED.* Поред тога, тврдње *expirationTime* и *audience* су обавезне јер су неопходне за аутентификацију путем токена(листинг 4.30)*.* С обзиром да *еxpirationTime, audience* и *issuer* не садрже податке о кориснику, оне не могу бити повезане са атрибутом корисника, већ морају имати дефинисану вредност (листинг 4.31), док остале тврдње морају бити повезане са неким атрибутом (4.32). Тврдња везана за време важења токена мора да има позитивну вредност (листинг 4.33).

**invariant** subjectRegisteredClaim('If claim name is "subject", it must be of type REGISTERED'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* = 'subject') ->

*forAll*(sc | *sc*.*type* = *EClaimType*::*REGISTERED*);

**invariant** issuerRegisteredClaim('If claim name is "issuer", it must be of type REGISTERED'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* = 'issuer') ->

*forAll*(sc | *sc*.*type* = *EClaimType*::*REGISTERED*);

**invariant** hasExpirationTimeClaim('The claim "expirationTime" must exist and be of type REGISTERED'):

*Claim*.*allInstances*() ->

*exists*(c | *c*.*name* = 'expirationTime' **and** *c*.*type* = *EClaimType*::*REGISTERED*);

**invariant** hasAudienceClaim('The claim "audience" must exist and be of type REGISTERED'):

*Claim*.*allInstances*() ->

*exists*(c | *c*.*name* = 'audience' **and** *c*.*type* = *EClaimType*::*REGISTERED*);

Листинг 4.30 – Основне регистроване тврдње морају бити типа *REGISTERED* и тврдње *expirationTime* и *audience* су обавезне

**invariant** issuerValueNotNull('If the claim name is "issuer", it must have value and must not be linked to an attribute'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* = 'issuer') ->

*forAll*(sc | *sc*.*value* <> **null** **and** *sc*.*claim\_attribute* = **null**);

**invariant** audienceValueNotNull('If the claim name is "audience", it must have value and must not be linked to an attribute'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* = 'audience') ->

*forAll*(sc | *sc*.*value* <> **null** **and** *sc*.*claim\_attribute* = **null**);

**invariant** expirationTimeValueNotNull('If the claim name is "expirationTime", it must have value and must not be linked to an attribute'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* = 'expirationTime') ->

*forAll*(sc | *sc*.*value* <> **null** **and** *sc*.*claim\_attribute* = **null**);

Листинг 4.31 – Тврдње *issuer, audience* и *expirationTime* морају имати дефинисану вредност и не могу бити везане за атрибут ентитета *User*

**invariant** otherClaimsNoValue('Claim must be linked to an attribute'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* <> 'issuer' **and** *c*.*name* <>

'audience' **and** *c*.*name* <> 'expirationTime') ->

*forAll*(sc | *sc*.*value* = **null** **and** *sc*.*claim\_attribute* <> **null**);

Листинг 4.32 – Тврдње које нису *issuer, audience* и *expirationTime* не могу имати дефинисану вредност и морају бити везане за атрибут ентитета *User*

**invariant** expirationTimeValueIsPositiveNumeric('If the claim name is "expirationTime", the value must be a positive number'):

*Claim*.*allInstances*() -> *select*(c | *c*.*name* = 'expirationTime') ->

*forAll*(sc | *sc*.*value* <> **null** **and** *sc*.*value*.*toInteger*() > 0);

Листинг 4.33 – Тврдња везана за време важења токена мора да има позитивну вредност

### Концепт *OAuth2*

*OAuth2* концепт моделује *Open Authorization* конфигурацију и омогућава корисницима пријаву на систем посредством провајдера. У раду је подржана само пријава на систем за овај безбедоносни механизам, те се подаци о корисницима не складиште у бази података апликације, већ у бази одабраног провајдера. Због тога су креирана ограничења која гарантују да неће бити дефинисани ентитети за кориснике и роле, као ни контролер за аутентификацију (листинг 4.34).Неопходно је да сваки тип провајдера буде конфигурисан највише једном, односно да назив провајдера буде јединствен (листинг 4.35).

|  |  |  |  |
| --- | --- | --- | --- |
| Назив асоцијације | Референцирани концепт | Кардиналитет | Опис |
| *providers* | *Provider* | 1..\* | Конфигурисани провајдери у апликацији |

Табела 4.18 – Асоцијације концепта *OAuth2*

**invariant** doesntHaveRoleForOauth('OAuth2 authentication requires no Role entities!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*Role*)) -> *size*() = 0;

**invariant** doesntHaveUserForOauth('OAuth2 authentication requires no User entities!'):

*Entity*.*allInstances*() -> *select*(e | *e*.*oclIsTypeOf*(*User*)) -> *size*() = 0;

**invariant** doesntHaveAuthControllerForOauth('OAuth2 authentication requires no Authentication controller!'):

*Controller*.*allInstances*() ->

*select*(e | *e*.*oclIsTypeOf*(*Authentication*)) -> *size*() = 0;

Листинг 4.34 – Није могуће дефинисани ентитете за кориснике и роле, као ни контролер за аутентификацију

**invariant** uniqueProviders('Providers must have unique names!'):

**self**.*providers* -> *isUnique*(p | *p*.*name*);

Листинг 4.35 – Назив провајдера је јединствен

### Концепт *Provider*

Концепт *Provider* описује компотенту *OAuth2.0* протокола који представља ентитет који пружа услуге аутентификације, односно проверава креденцијале које корисник уноси и утврђује њихову исправност. Провајдери могу бити различити, најчешће су то друштвене мреже, пружаоци услуга електронскае поште итд.

|  |  |  |  |
| --- | --- | --- | --- |
| Назив обележја | Тип податка | Обавезно | Опис |
| *name* | *EString* | да | Назив провајдера |
| *clientId* | *EString* | да | Идентификациони број клијента додељен од стране провајдера |
| *clientSecret* | *EString* | да | Тајни кључ клијента додељен од стране провајдера за сигурносну контролу приликом комуникације са провајдером |
| *redirectUri* | *EString* | не | Адреса за преусмерење након успешне аутентификације |

Табела 4.19 – Обележја концепта *Provider*

## Конкретна синтакса

Коришћењем радног оквира *Xtext*, на основу мета-модела генерисана је почетна верзија конкретне синтаксе, односно граматике која описује текстуалну репрезентацију наменског језика. Ова граматика прилагођена је домену, како би била лако читљива и интуитивна за развојне тимове којима је *securityDSL* намењен.

Ово поглавље обухватиће увид у структуру конкретне синтаксе и опис како су концепти мета-модела преведени у текстуалне елементе језика.

### Граматика наменског језика *securityDSL*

Иницијална граматика наменског језика *securityDSL* креирана на основу мета-модела помоћу радног оквира *Xtext* креће од коренског концепта *Application*. Листинг 4.36 приказује почетно правило за опис коренског концепта. Примећује се да почетна верзија граматике подсећа на уобичањене синтаксе за дефинисање разних конфигурационих фајлова, те таква граматика не захтева велике измене узимајући у обзир да циљну групу чине експерти у пољу безбедносне конфигурације.

Ради прегледности, уведене су мале измене: уклањање витичастих заграда са одређених места, додавање двотачке након назива обележја а пре дефинисања вредности обележја, коришћења угластих заграда за обележавање листи итд. Ново правило за опис коренског концепта *Application*  (листинг 4.37) прилагођено је тако да буде уредније и прегледније, иако није дошло до значајних измена. Оваква ажурирана граматика омогућава једноставно дефинисње различитих аспеката безбедносне конфигурације, као и свих осталих концепата наменског језика. У остатку поглавља приказана су правила за опис свих концепата наменског језика *securityDSL,* укључујући и енумерације.

|  |
| --- |
| Application **returns** *Application*:  {*Application*}  'Application'  name=EString  '{'  'artifact' artifact=EString  ('name' = name=EString)?  'group' group=EString  ('packageName' packageName=EString)?  ('description' description=EString)?  ('port' port=ELongObject)?  ('hostname' hostname=EString)?  ('app\_database' app\_database=Database)?  ('app\_entities' '{' app\_entities+=Entity  ( "," app\_entities+=Entity)\* '}' )?  ('app\_controllers' '{' app\_controllers+=Controller  ( "," app\_controllers+=Controller)\* '}' )?  ('app\_security' app\_security=Security)?  '}'; |

Листинг 4.36 – Иницијално правило за опис концепта *Application*

|  |
| --- |
| Application **returns** *Application*:  {*Application*}  'application:'  'artifact:' artifact=EString  ('name:' name=EString)?  'group:' group=EString  ('packageName:' packageName=EString)?  ('description:' description=EString)?  ('port:' port=ELongObject)?  ('hostname:' hostname=EString)?    ('database:' app\_database=Database)?    ('model:'  ('user:' app\_entities+=User)?  ('role:' app\_entities+=Role)?  )?    ('security:' app\_security=Security )?    ('controller:'  ('auth:' app\_controllers+=Authentication)?  )?; |

Листинг 4.37 – Правило за опис концепта *Application*

|  |
| --- |
| Database **returns** *Database*:  'vendor:' vendorName=EDatabaseType  'url:' url=EString  'username:' username=EString  'password:' password=EString;  **enum** EDatabaseType **returns** *EDatabaseType*:  PostgreSQL = 'PostgreSQL' | Oracle = 'Oracle' | MySQL = 'MySQL'; |

Листинг 4.38 – Правило за опис концепта *Database*

|  |
| --- |
| Attribute **returns** *Attribute*:  '{'  (identifier?='identifier')?  (credential?='credential')?  'name:' name=EString  'type:' type=EType  ('collumnName:' collumnName=EString)? '}';  **enum** EType **returns** *EType*:  String = 'String' |  Float = 'Float' |  Long = 'Long' |  Integer = 'Integer' |  Double = 'Double' |  DateTime = 'DateTime' |  Date = 'Date'; |

Листинг 4.39 – Правило за опис концепта *Attribute*

|  |
| --- |
| Role **returns** *Role*:  {*Role*}  ('tableName:' tableName=EString)?  ('attributes:'  '['entity\_attributes+=Attribute ("," entity\_attributes+=Attribute)\*']')?  ('roles:'  '['role\_instances+=RoleInstance ("," role\_instances+=RoleInstance)\*']')?; |

Листинг 4.40 – Правило за опис концепта *Role*

|  |
| --- |
| RoleInstance **returns** *RoleInstance*:  {*RoleInstance*}  (client?='client')? name=EString; |

Листинг 4.41 – Правило за опис концепта *RoleInstance*

|  |
| --- |
| User **returns** *User*:  {*User*}  ('tableName:' tableName=EString)?  ('attributes:'  '['entity\_attributes+=Attribute ("," entity\_attributes+=Attribute)\*']')?; |

Листинг 4.42 – Правило за опис концепта *User*

|  |
| --- |
| Endpoint **returns** *Endpoint*:  '{'  'type:' type=EEndpointType  'url:' url=EString  'methodName:' methodName=EString  ('method:' method=EEndpointMethod)?  ('roleAuthorities:'   '['role\_authorities+=[*RoleInstance*|EString]   ( "," role\_authorities+=[*RoleInstance*|EString])\* ']' )?  '}';  **enum** EEndpointType **returns** *EEndpointType*:  REGISTRATION = 'REGISTRATION' |  LOGIN = 'LOGIN' |  LOGOUT = 'LOGOUT' |  OTHER = 'OTHER';  **enum** EEndpointMethod **returns** *EEndpointMethod*:  GET = 'GET' | POST = 'POST' | PUT = 'PUT' | DELETE = 'DELETE'; |

Листинг 4.43 – Правило за опис концепта *Endpoint*

|  |
| --- |
| Authentication **returns** *Authentication*:  {*Authentication*}  'name:' name=EString  'path:' path=EString  ('endpoints:'  '['controller\_endpoints+=Endpoint  ( "," controller\_endpoints+=Endpoint)\* ']' )?; |

Листинг 4.44 – Правило за опис концепта *Authentication*

|  |
| --- |
| BasicAuthentication **returns** *BasicAuthentication*:  {*BasicAuthentication*} 'basicAuthentication'; |

Листинг 4.45 – Правило за опис концепта *BasicAuthentication*

|  |
| --- |
| JWT **returns** *JWT*:  'jwt:'  'signatureAlgorithm:' signatureAlgorithm=EString  'secret:' secret=EString  'claims:' '[' claims+=Claim ( "," claims+=Claim)\* ']'; |

Листинг 4.46 – Правило за опис концепта *JWT*

|  |
| --- |
| Claim **returns** *Claim*:  '{'  type=EClaimType','  name=EString ':'  (value=EString)?  ('attribute' claim\_attribute=[*Attribute*|EString])?  '}';  **enum** EClaimType **returns** *EClaimType*:  PRIVATE = 'PRIVATE' | PUBLIC = 'PUBLIC' | REGISTERED = 'REGISTERED'; |

Листинг 4.46 – Правило за опис концепта *Claim*

|  |
| --- |
| OAuth2 **returns** *OAuth2*:  'OAuth2.0:'  'providers:' '[' providers+=Provider ( "," providers+=Provider)\* ']'; |

Листинг 4.47 – Правило за опис концепта *OAuth2*

|  |
| --- |
| Provider **returns** *Provider*:  '{'  'name:' name=EString ','  'clientId:' clientId=EString ','  'clientSecret:' clientSecret=EString  (',' 'redirectUri:' redirectUri=EString)?  '}'  ; |

Листинг 4.48 – Правило за опис концепта *Provider*

## Примери модела описаних наменским језиком *securityDSL*

Ово поглавље пружа преглед претходно описане конкретне синтаксе кроз примере за сваки од подржаних безбедносних механизама. Кључне речи, називи обележја и вредности енумерација приказане су бордо бојом, вредности текстуалних обележја представљене су плавом, док су нумеричке вредности приказане сивом бојом. За сваки од примера, коренски концепт Application дефинисан је након кључне речи application навођењем вредности обележја овог концепта. Параметри за конфигурацију базе податакадефинишу се након кључне речи *database,* ентитети након кључне речи *entity*, контролери након кључне речи *controller*, а безбедносни аспекти након кључне речи *security.*

Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *Basic Authentication*

Концепт *Basic Authentication* не захтева додатан опис конфигурације. Битно је дефинисати једно обележје које је креденцијал навођењем кључне речи *credential.* Инстанце рола наводе се након кључне речи *roles* у оквиру концепта *role.* Уколико је инстанца клијент, наводи се кључна реч *client.*

|  |
| --- |
| **application:**  **artifact:** "securityDsl"  **group:** "uns.ftn"  **description:** "This is an app that is generated with security DSL"  **port:** 8080  **hostname:** "localhost"    **database:**  **vendor:** **PostgreSQL**  **url:** "localhost:5432/securityDsl"  **username:** "securityDsl"  **password:** "securityDsl"    **entity:**  **user:**  **tableName:** "users"  **attributes:** [  {  **identifier**  **name:** "id"  **type:** **Long**  },  {  **credential**  **name:** "username"  **type:** **String**  **collumnName:** "username"  },  {  **name:** "firstName"  **type:** **String**  **collumnName:** "first\_name"  },  {  **name:** "lastName"  **type:** **String**  **collumnName:** "last\_name"  }  ]  **role:**  **roles:** ["admin", **client** "user"]  **security:**  **basicAuthentication**    **controller:**  **auth:**  **name:** "AuthController"  **path:** "/auth"  **endpoints:** [  {  **type:** **REGISTRATION**  **url:** "/registration"  **methodName:** "registration"  },  {  **type:** **LOGIN**  **url:** "/login"  **methodName:** "login"  },  {  **type:** **LOGOUT**  **url:** "/logout"  **methodName:** "logout"  }  ] |

Листинг 4.49 – Пример модела веб апликације са конфигурисаним безбедносним механизмом *Basic Authentication* у радном оквиру *Spring*

Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *JWT*

Након кључне речи *jwt* дефинишу се неопходни параметри за аутентификацију помоћу JWT токена. Пре навођења назива и вредности тврдње, потребно је нагласити ког је она типа, а уколико је тврдња везана за атрибут, неопходно је навести кључну реч *attribute.*

|  |
| --- |
| **application:**  **artifact:** "securityDsl"  **group:** "uns.ftn"  **description:** "This is an app that is generated with security DSL"  **port:** 8080  **hostname:** "localhost"    **database:**  **vendor:** **PostgreSQL**  **url:** "localhost:5432/securityDsl"  **username:** "securityDsl"  **password:** "securityDsl"    **entity:**  **user:**  **tableName:** "users"  **attributes:** [  {  **identifier**  **name:** "id"  **type:** **Long**  },  {  **credential**  **name:** "username"  **type:** **String**  **collumnName:** "username"  },  {  **name:** "firstName"  **type:** **String**  **collumnName:** "first\_name"  },  {  **name:** "lastName"  **type:** **String**  **collumnName:** "last\_name"  }  ]  **role:**  **attributes:**  [  {  **identifier**  **name:** "id"  **type:** **Long**  },  {  **name:** "name"  **type:** **String**  }  ]  **roles:** ["admin", **client** "user"]  **security:**  **jwt:**  **signatureAlgorithm:** "HS512"  **secret:** "somesecret"  **claims:**  [  {**REGISTERED** , subject : **attribute** username},  {**REGISTERED** , audience : "AUDIENCE\_WEB"},  {**REGISTERED** , expirationTime : "333"},  {**REGISTERED** , issuer : "securityDSL"},  {**PUBLIC** , firstName : **attribute** firstName}  ]    **controller:**  **auth:**  **name:** "AuthController"  **path:** "/auth"  **endpoints:** [  {  **type:** **REGISTRATION**  **url:** "/registration"  **methodName:** "registration"  },  {  **type:** **LOGIN**  **url:** "/login"  **methodName:** "login"  },  {  **type:** **LOGOUT**  **url:** "/logout"  **methodName:** "logout"  }  ] |

Листинг 4.50 – Пример модела веб апликације са конфигурисаним безбедносним механизмом *JWT* у радном оквиру *Spring*

Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *OAuth2*

Придржавањем *OCL* ограничења, за безбедносну конфигурацију *OAuth2.0* мехеанизма нису дефинисани концепти *User, Role* и *Authentication.* При дефинисању концепта *OAuth2.0,* након кључне речи *providers* наведени су провајдери са свим неопходним обележјима.

|  |
| --- |
| **application:**  **artifact:** "securityDsl"  **group:** "uns.ftn"  **description:** "This is an app that is generated with security DSL"  **port:** 8080  **hostname:** "localhost"    **database:**  **vendor:** **PostgreSQL**  **url:** "localhost:5432/securityDsl"  **username:** "securityDsl"  **password:** "securityDsl"    **security:**  **OAuth2.0:**  **providers:**[  {  **name:** "google",  **clientId:** "x",  **clientSecret:** "x"  },  {  **name:** "github",  **clientId:** "x",  **clientSecret:** "x"  }  ] |

Листинг 4.51 – Пример модела веб апликације са конфигурисаним безбедносним механизмом *OAuth2.0* у радном оквиру *Spring*

# Генерисање веб апликација са безбедносном конфигурацијом

# Закључак

# Скраћенице

* *ecurityDSL – Security Domain-Specific Language*
* *JWT - JSON Web Token*
* *OAuth2.0 – Open Authorization 2.0*
* *OCL* – *Object Constraint Language*
* URL – *Uniform Resource Locator*
* HTTP – *Hypertext Transfer Protocol*

# Литература

*[1] Java Documentation,* [*https://docs.oracle.com/en/java/*](https://docs.oracle.com/en/java/)

*[2] Spring Boot Documentation,* [*https://spring.io/projects/spring-boot/*](https://spring.io/projects/spring-boot/)

*[3] PostgreSQL,* [*https://www.postgresql.org/about/*](https://www.postgresql.org/about/)

*[x] Basic Authentication,* [*https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/basic.html#page-title*](https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/basic.html%23page-title)

# Биографија

Јелена Хрњак рођена је 21. августа 1999. године у Бачкој Тополи где је стекла основно образовање у основној школи ,,Никола Тесла”. Даље школовање је наставила у Суботици где је завршила Гимназију ,,Светозар Марковић”, природно-математички смер. Школске 2018/2019 године уписује се на Факултет техничких наука Универзитета у Новом Саду, смер Рачунарство и аутоматика. Основне академске студије завршила је 2022. године и исте године уписује се на мастер академске студије на студијском програму Рачунарство и аутиматика Факултета теничких наука. Положила је све испите предвиђене планом и програмом мастер академских студија.