Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/12BE05Stanarevic

ANGULAR КОМПОНЕНТА ЗА КОНТРОЛУ ПРИСТУПА БАЗИРАНУ НА КОРИСНИЧКИМ УЛОГАМА

ANGULAR COMPONENT FOR ROLE-BASED ACCESS CONTROL

Јелена Станаревић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У раду је описана Angular компонента која омогућује динамичку ауторизацију на клијентској страни апликације. Ауторизација, односно, контрола приступа базирана је на RBAC (Role-based Access Control) моделу. Angular компонента представља проширење постојећих могућности Angular програмског оквира и омогућава уклањање/приказивање компоненти и делова компоненти, као и дозволу/забрану измене или уноса одређених поља у оквиру компоненти. Верификација креиране Angular компоненте приказана је кроз информациони систем за физикалну терапију.*

Кључне речи: *Kontrola pristupa, RBAC, Angular framework, dinamička autorizacija*

Abstract – *This paper describes the Angular component that is responsible for dynamic authorization on the client side of the application. Authorization (access control) is based on the RBAC (Role-based Access Control) model. The Angular component is an extension of the existing capabilities of the Angular framework and allows the removal / display of components and parts of components, as well as the permission / prohibition of modification or entry of certain fields within the components. Verification of the created Angular component is presented through the information system for physical therapy.*

Keywords: *Access control, RBAC, Angular framework, dynamic authorization*

1. УВОД

Често се *Role-based Access Control* модел контроле приступа имплементира на серверској страни апликације. Међутим, овакав приступ има великуману, а то је брзина којом се проверава да ли корисник има или нема право приступа делу система. Приступ имплементиран у оквиру система за физикалну терапију, јесте динамичка ауторизација на клијентској страни апликације. Претходно споменута брзина провере права приступа корисника се знатно повећава са овим приступом.

У оквиру овог рада биће описана и презентована посебно креирана Angular компонента која је одговорна за примену контроле приступа базиране на корисничким улогама (*Role-based Access Control, RBAC*), односно, за динамичко генерисање различитих компоненти и делова компоненти у складу са правима приступа.

2. КОНТРОЛА ПРИСТУПА У ИНФОРМАЦИОНИМ СИСТЕМИМА

У области информационе безбедности (енгл. Information Security) контрола приступа подразумева праксу спречавања неовлашћеног приступа, коришћења, откривања, модификације, снимања или уништавања информација [1].

2.1. Role-based access control

Концепт контроле приступа заснован на улогама (енгл. Role-based access control, RBAC) јавио се са увођењем вишекорисничких *online* система и апликација око 1970. године. Централно место у оквиру RBAC модела заузимају појмови и односи улога, дозвола и корисника. Дозволе су везане за улоге, док се корисницима те улоге додељују.

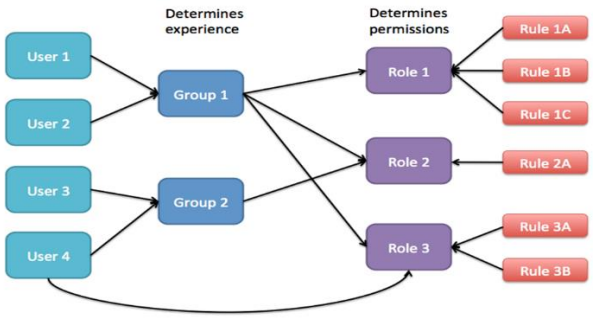
Улоге се креирају у складу са различитим корисничким функцијама у организацији или компанији. Дакле, корисницима се додељују улоге у складу са одговорностима и надлежностима које имају у оквиру компаније. С обзиром да су промене у оквиру било ког информационог система неизбежне, корисницима је могуће додељивати друге улоге у току времена.

Такође, улогама могу да се уклањају претходне и додају нове дозволе [1]. У оквиру организације улоге се ретко мењају, док се дозволе и корисници чешће могу мењати. Однос између улога, корисника и дозвола приказан је на слици 2.1.

Основна предност RBAC модела јесте управо централизована администрација улога, дозвола и корисника.

Централизована администрација обезбеђује једноставније управљање у смислу да се све своди на додељивање одговарајућих улога корисницима, за разлику од ACL листи (енгл. Access Control Lists). У оквиру ових листи неопходно је да се за сваког новог корисника прође кроз сваки ресурс система и његовој листи придружи корисник са одговарајућим правом које има над тим ресурсом [1].

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био проф. др Горан Сладић.



Слика 2.1. *Однос корисника, група, улога и дозвола* [1]

3. СПЕЦИФИКАЦИЈА КОМПОНЕНТЕ ЗА КОНТРОЛУ ПРИСТУПА

У оквиру овог поглавља биће презентована улога компоненте за контролу приступа, али без детаља око саме имплементације.

3.1. Постојеће могућности Angular framework-а

Angular представља платформу и програмски оквир (енгл. framework) за израду клијентске стране интернет апликација. Програмски оквир је направљен у JavaScript-у и омогућава израду тзв. *single-page* апликација користећи TypeScript и HTML. Angular нуди још једну могућност динамичке измене структуре HTML фајла одређене компоненте уз помоћ тзв. структуралних директива. Структуралне директиве, попут **ngIf* или **ngFor* омогућавају да се мења изглед *Document Object Model* стабла дефинисаног у оквиру HTML странице у зависности од вредности појединих података доступних конкретној компоненти [2]. **ngIf* директива представља уграђену директиву програмског оквира, која уклања или креира део *Document Object Model* стабла. Претходно наведено зависности од резултата валидације израза на који се односи (слика 3.1) [2].

```
<div *ngIf="condition">Content to render when condition is true.
</div>
```

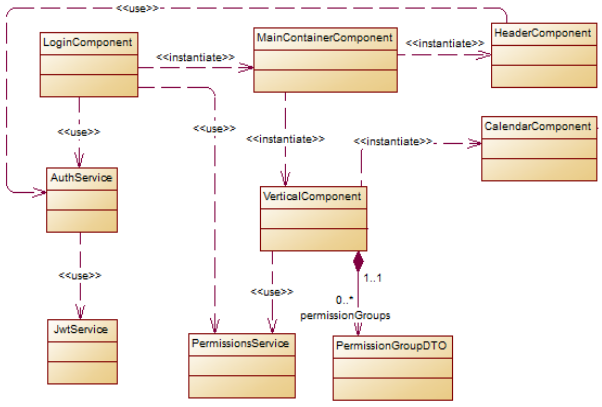
Слика 3.1 – *Пример примене *ngIf директиве*

3.2. Модел компоненте за контролу приступа

На сликама у наставку биће приказан део UML дијаграма класа за клијентску страну апликације. Конкретно, приказане су постојеће компоненте у оквиру Angular програмског овкира, сервисне и модел класе, као и везе између њих. Комуникација ових компоненти је кључна у реализацији динамичке ауторизације.

На слици 3.2. приказан је део UML дијаграма класа који се односи на компоненте клијентске стране које се инстанцирају приликом пријаве на систем, везе између њих, као и везе компоненти са сервисним класама. *LoginComponent* компонента, путем *dependency injection* концепта, садржи везе ка уграђеним компонентама Angular програмског оквира [2].

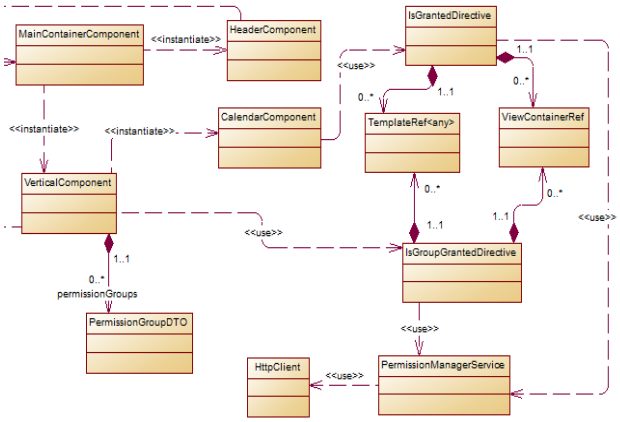
Такође, садржи везе према сервисима задуженим првенствено за управљање процесом пријаве корисника на систем (*AuthService* и *PermissionsService*). *AuthService* у себи садржи везу ка *JwtService* сервис класи. Везе које садржи *LoginComponent* компонента омогућавају јој да користи све методе претходно набројаних сервиса. *JwtService* сервис класа омогућава управљање JWT токенима. У оквиру система за физикалну терапију имплементирана је аутентификација учесника система уз помоћ JSON Web Токена или, скраћено, JWT [3].



Слика 3.2 – *Део UML дијаграма класа који се односи на компоненте одговорне за пријаву корисника на систем*

На слици 3.3 приказан је део UML дијаграма класа који се односи на везе између компоненти приказа и директива које омогућавају приказ/уклањање компоненти и делова компоненти. Директиве задужене за динамичку ауторизацију у систему су *IsGrantedDirective* и *IsGroupGrantedDirective*.

Обе су путем *dependency injection* концепта повезане са *PermissionManagerService* сервис класом, која, преко комуникације са *PermissionsService* сервис класом, омогућава преузимање и проверавање дозвола које улоговани корисник поседује. *CalendarComponent* компонента такође користи *IsGrantedDirective* директиву и има везу ка њој.



Слика 3.3 – *Део UML дијаграма класа који се односи на везу компоненти са директивом*

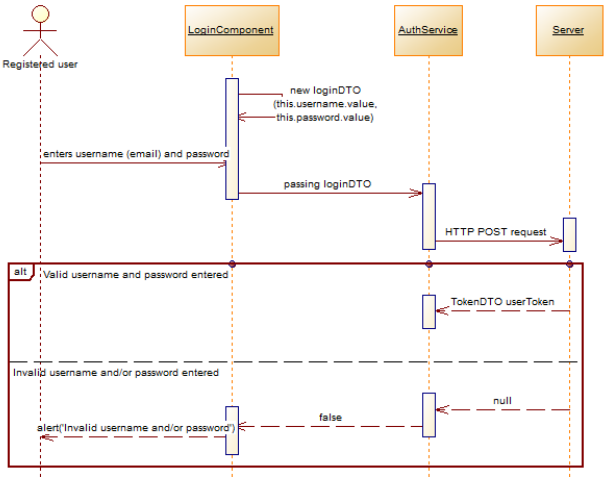
IsGrantedDirective и *IsGroupGrantedDirective*, преко инстанци *TemplateRef<any>* и *ViewContainerRef*, омогућавају да се део HTML садржаја креира, обрише, или да се забрани, односно, омогући измена тог садржаја.

3.3. Интеракција компоненте за контролу приступа са компонентама система

На сликама у наставку биће презентован целокупан процес и логика иза креирања/уклањања компоненти и делова компоненти, као и омогућавања/забране уноса и измене појединих поља у складу са скупом дозвола пријављеног корисника. На слици 3.4 приказана је комуникација компоненти и сервиса приликом пријаве корисника на систем. Регистровани корисник уноси емаил и лозинку у одговарајућа поља *LoginComponent* компоненте, која од датих информација формира објекат типа *LoginDTO* и прослеђује формирани објекат *AuthService* сервис класи [4].

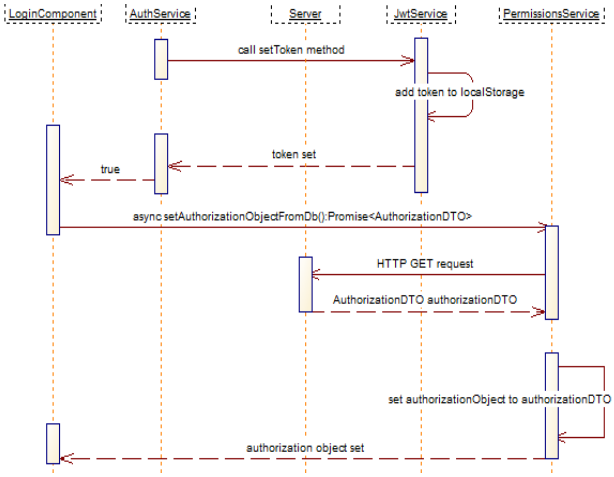
Ова класа комуницира са серверском страном апликације, ради верификовања исправности унетих креденцијала корисника. Серверска страна је у дијаграмима секвенци приказана по тзв. *Black-Box* принципу. Овај принцип подразумева да се одређени систем или део система представи само преко улазних и излазних параметара, без самих детаља имплементације и начина функционисања истог [5]. Комуникација са серверској страном обавља се путем HTTP POST захтева.

Након провере послатих креденцијала, серверска страна има два алтернативе. Једна подразумева да унети емаил и/или лозинка нису исправни и у том случају сервер враћа null вредност и клијентска страна обавештава корисника о неисправности креденцијала. Друга алтернатива подразумева да сервер одговори са формираним JWT токеном, као потврдом успешне пријаве корисника на систем [3].



Слика 3.4 – *пријава корисника на систем*

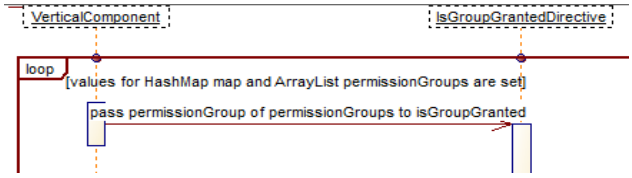
Преузети JWT токен се шаље од *AuthService* сервис класе до *JwtService* сервис класе која смешта тај токен у *locale storage* интернет претраживача. Информација о успешно ускладиштеном JWT токену враћа се до *LoginComponent* компоненте. Даље следи иницијализација сервисних класа и компоненти почетне странице апликације [3]. *PermissionsService* сервис класа добавља објекат ауторизације.
Наведени објекат садржи информације о свим улогама, дозволама и групама дозвола корисника путем комуникације са серверском страном (слика 3.5).



Слика 3.5 *Сценарио успешне пријаве корисника на систем*

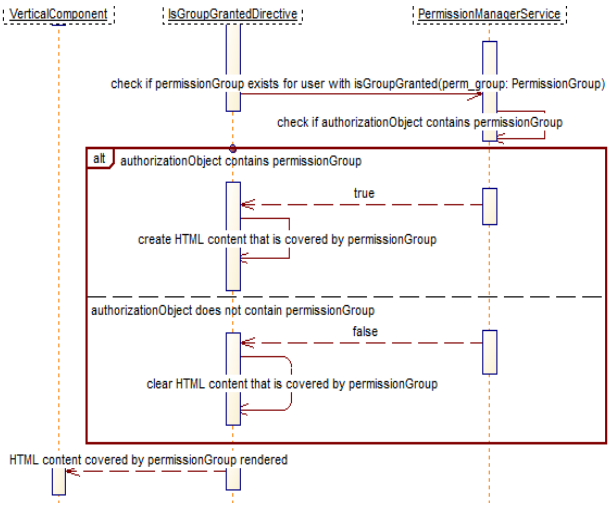
Иницијализација *VerticalComponent* компоненте, која репрезентује вертикални мени почетне странице и попуњава се линковима за извршавање појединих операција (додавање новог пацијента, измена одређеног термина, итд.) врши се путем комуникације са *IsGratedDirective* и *IsGroupGrantedDirective* директивама. Попуњавање компоненте линковима зависи од скупа дозвола пријављеног корисника. Најпре се кроз комуникацију са *PermissionsService* сервис класом (последично и са серверском страном апликације), добављају одвојено листа свих група дозвола и мапа груписаних свих дозвола у систему.

С обзиром да су дозволе распоређене по групама у облику мапе у компоненти *VerticalComponent*, приликом њене иницијализације најпре се проверава да ли корисник има право да приступа одређеној групи дозвола. Та провера врши се преузимањем по једне из листе свих група и комуникацијом са *IsGroupGrantedDirective* директивом. Споменута директива се обраћа *PermissionManagerService* сервис класи која ће, на основу претходно преузетог *authorizationDTO* објекта од *PermissionsService* класе (види слику 3.5) вратити директиви информацију о томе да ли корисник поседује прослеђену групу дозвола [3]. У случају да корисник поседује прослеђену групу дозвола, HTML садржај се креира, док се у супротном брише у оквиру директиве (слике 3.6 и 3.7).



Слика 3.6 – *Слање по једне групе дозвола*

Пре него што се провери да ли корисник поседује наредну групу, постојећа група која је анализирана се користи као кључ у мапи у оквиру *VerticalComponent* компоненте. На основу кључа преузимају се и проверавају дозволе везане за исти.



Слика 3.7 – *Proveravanja prisustva prosleđene grupe dozvola*

У овој ситуацији *VerticalComponent* компонента комуницира са *IsGrantedDirective* директивом, прослеђујући јој дозволу по дозволу. Директива у комуникацији са *PermissionManagerService* сервис класом проверава присуство сваке дозволе у скупу дозвола пријављеног корисника. У складу с тим *PermissionManagerService* сервис класа враћа вредности *true* или *false* уколико корисник поседује, односно, не поседује дозволу у оквиру свог скупа дозвола. *IsGrantedDirective* директива ће у складу са тим креирати или уклонити одређени HTML садржај.

4. ИМПЛЕМЕНТАЦИЈА КОМПОНЕНТЕ ЗА КОНТРОЛУ ПРИСТУПА

4.1. Провера права приступа ресурсу система на клијентској страни

Константно пресретање захтева и провера скупа дозвола које корисник са одређеном улогом има над сваким ресурсом представља безбедан и поуздан начин имплементације ауторизације. Међутим, овај начин имплементације је знатно спорији од алтернативе која ће бити презентована у овом одељку и која постоји у систему. Алтернатива подразумева динамичку ауторизацију на клијентској страни апликације, односно, проверу скупа дозвола над ресурсима система без учешћа пресретача (енгл. Interceptor) на серверској страни.

4.1.1 Динамичка ауторизација у систему за физикалну терапију

У систему креирана је нова директива као проширење постојећих, која омогућава динамичко проверавање дозвола пријављеног корисника и приказ или уклањање компоненти или делова компоненти, као и забрану/дозволу измене појединих поља. Директива је означена са *@Decorater({selector: '[appIsGranted]'})*. Као и код структуралних директива, примена кода нове директиве подразумева да се HTML садржај обухвати са селектором директиве, односно, у овом случају – са **appIsGranted*. Кључне елементе у оквиру директиве представљају *ViewContainterRef* и *TemplateRef*, који су уједно и поља у оквиру исте. *ViewContainterRef* представља контејнер у који се може додати прикази дефинисани *TemplateRef* шаблоном.

Тих приказа може бити један или више. Овај шаблон садржи део *Document Object Model* стабла у оквиру HTML фајла конкретне компоненте. Тај део ће бити генерисан, елиминисан или ће се омогућити/ онемогућити унос и измена истог у зависности од скупа дозвола пријављеног корисника.

6. ЗАКЉУЧАК

У раду су описани основни појмови везани за контролу приступа, са посебним нагласком на *Role-based Access Control* или *RBAC* модел контроле приступа. Овај модел описан је у посебном одељку. Спецификација Angular компоненте, односно, директиве, одговорне за динамичку ауторизацију, налази се у оквиру одвојеног поглавља. У оквиру истог поглавља презентоване су постојеће могућности Angular програмског оквира у погледу динамичког генерисања компоненти и делова компоненти, затим презентовање компоненте путем дела UML дијаграма класа и одређених UML дијаграма секвенци.

Кроз пример генерисања вертикалног менија објашњена је примена креиране директиве и начин на који компоненте, сервиси и директиве комуницирају у циљу реализације динамичке провере дозвола корисника у систему. Потенцијална проширења представљеног решења би могла бити могућност креирања нових дозвола у систему и њихово додељивање постојећим или потпуно новим улогама. Затим могућност да се привремено модификује постојећи скуп дозвола (нпр. да се поједине дозволе ипак онемогуће уколико се у оквиру система дода потпуно нови корисник, без искуства у раду са системом, са улогом администратора).

7. ЛИТЕРАТУРА

[1] Role-based Access Control, Ravi S. Sandhu, http://www.profsandhu.com/articles/advcom/adv_comp_rbac.pdf
[2] Izrada web aplikacije putem Angulara 6 razvojnog okvira, F Tuđan, https://repozitorij.unin.hr/islandora/object/unin:2023/datastream/PDF/download
[3] Security of JSON Web Tokens (JWT), https://cyberpolygon.com/materials/security-of-json-web-tokens-jwt/
[4] Create Data Transfer Objects (DTOs), https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5
[5] Black-Box Model, https://www.sciencedirect.com/topics/engineering/black-box-model

Кратка биографија:

Јелена Станаревић рођена је 28.07.1995. године у Новом Саду. Завршила је Основну школу ”Васа Стајић”, а потом и Гимназију ”Јован Јовановић Змај” (природно-математички смер), у Новом Саду. Гимназију је завршила 2014., и исте године је уписала Факултет техничких наука у Новом Саду, одсек Рачунарство и аутоматика. Завршила је основне академске студије 27.09.2018.