Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

JUHO TUUNA

Keycloakin hyödyntäminen .NET Core -sovelluksessa

SÄHKÖ- JA AUTOMAATIOTEKNIIKAN
TUTKINTO-OHJELMA
2023

TIIVISTELMÄ

Tuuna, Juho: Keycloakin hyödyntäminen .NET Core -sovelluksessa.
Opinnäytetyö, AMK
Tutkinto-ohjelma: Sähkö- ja automaatiotekniikka
Joulukuu 2023
Sivumäärä: 26

Opinnäytetyössä tehtiin .NET Core -sovellukseen käyttäjänhallintatoiminto käyttämällä Keycloak -työkalua. Projektia varten esitellään pääpiirteissään toimintoja, jotka olivat toteutuksen kannalta oleellisia. Samalla tavalla esitellään Oauth- ja OpenIdConnect- tekniikoita. Työn tarkoituksena oli toimia selvänä ja ymmärrettävänä oppaana Keycloakin käyttöön ja asennukseen .NET Core sovelluksessa. Työn oppaassa ei käydä läpi .NET Core -sovelluksen luotia, sillä työ tehtiin osana jo olemassa olevan sovelluksen kehitystä ja pääkohteena tässä työssä on Keycloak eikä Keycloakia käyttävä sovellus.

Opinnäytetyön toteutus tehtiin osana suurempaa kirjanpitosovelluksen päivitystä. Työn julkaisun aikana työn toteutusta käyttävää versiota sovelluksesta ei ole vielä otettu käyttöön muiden sovellukseen tehtävien päivitysten takia.

Avainsanat: käyttäjähallinta, autentikointi, valtuutus, Keycloak, OpenIdConnect

Abstract

Tuuna Juho: Keycloak in a .NET Core -application.
Bachelor's Thesis
Electrical and Automation Engineering
November 2023
Number of pages: 26

In this thesis, I added a user management feature for a .NET Core application using a tool called Keycloak. During that I also explained and went through some functionalities of Keycloak, Oauth and OpenIdConnect protocols. The purpose of this thesis was to work as a clear and coherent guide for installing Keycloak and using it with a .NET Core application. In the guide I do not go through or explain the creation process for the application as the focus of this thesis is Keycloak and not the application using it.

During the publication of this thesis the version of the application using Keycloak has not yet been taken into use because of other changes needed to be implemented into the application.

Keywords: user management, authentication, authorization, Keycloak, OpenIdConnect

SISÄLLYS

1 JOHDANTO

Monissa yrityksissä käytetään erilaisia ohjelmia kirjanpidossa, joissa käsitellään tietosuojan alaisia asioita (Tietosuojalaki 1050/2018, 2 luku 4 § ja 6 §). Tässä opinnäytetyössä ratkaisen kirjanpidon autentikointiin ja valtuutukseen liittyviä haasteita ja tuon siitä esille ratkaisuja. Käyn läpi Keycloakin käyttöönottoa käyttäjähallintaan .NET 6 -pohjaisessa kirjanpitosovelluksessa. Työ on osa suurempaa sovelluksen ohjelmistokehyksen päivitystä yrityksessä, jonne opinnäytetyön tuote valmistuu. Opinnäytetyössä rajaan aiheeni sovellukseen toteutettavaksi käyttäjänhallinnaksi, käyttäjän autentikoinniksi ja valtuutukseksi.

Selkeyden vuoksi tulen tässä opinnäytetyössä pitäytymään yhdenmukaisesti, mahdollisuuksien mukaan, suomenkielisessä termistössä. Ict-alalla ja koodikielessä on kuitenkin paljon termejä, joita käyttäessä on miellekkäämpää käyttää alkuperäistä termistöä.

Tämä opinnäytetyö on tehty paikallisessa testiympäristössä ja käytän testiympäristön tekoon Docker-palvelua. Käytän koodin kirjoittamiseen ja muokkaamiseen Visual Studio 2022 Community edition -ohjelmointiympäristöä. Opinnäytetyön tuote otetaan päivittäiseen kirjanpidolliseen käyttöön, kun muut päivitettävän sovelluksen muutokset ovat valmiita ja kaikki päivitykset on todettu toimiviksi.

2 KÄYTTÄJÄNHALLINTA OPINNÄYTETYÖN PROJEKTISSA

Käyttäjänhallinta on monien erilaisten sovellusten ja sivustojen käyttämisen selkäranka, jonka ympärille kaikki muu liitetään. Tarkoituksenani on seuraavaksi avata, että mitä on käyttäjänhallinta ja mihin sitä todella tarvitaan. Käyttäjänhallintaan liittyen käyn läpi myös autentikoinnin ja valtuuttamisen perusteita ja konsepteja. Näiden perusperiaatteiden käyttö on tärkeää sovelluksissa, jotka tarvitsevat pääsyä käyttäjän tietoihin.

2.1 Käyttäjänhallinta kattoterminä

Käyttäjänhallinnalla kattoterminä viitataan käyttäjätilien ylläpitoon eli tilien luontiin, muokkaamiseen ja poistoon. Käyttäjänhallintaoikeudet annetaan ylläpitäjälle ja käyttäjätilin omistajalle. Ylläpitäjällä tarkoitan käyttäjää, jolla on yleensä valtuutukset muokata sovelluksen mahdollisia asetuksia, ja havainnoida ja jopa muokata muiden käyttäjien tietoja tarvittaessa. Omistajalla tarkoitan opinnäytetyössä henkilöä, jota varten käyttäjätili on tehty. Ylläpitäjä tarvitsee käyttäjänhallintaoikeuksia käyttäjien hallinnointiin. Omistaja taas tarvitsee oikeudet omien tietojensa hallitsemiseen.

Jotta käyttäjänhallintatoiminnot saavutetaan tehokkaasti, kannattaa kyseessä olevan sovelluksen vaatia käyttäjältä rekisteröinnin ohessa toimivaa sähköpostia tilin vahvistukseen. Yleensä vaaditaan myös vahva salasana, joka sovelluksen kannattaa hajauttaa ja salata tietoturvallisuuden lisäämiseksi.

2.2 Valtuutus käyttäjänhallinnassa

Valtuutus on sovelluksen tai sovelluksen osien käytön luvan myöntämistä tai kieltämistä. Jokaisessa sovelluksessa, jossa on minkäänlaista käyttäjänhallintaa, niin on jonkinlaista valtuutushallintaa. Mikäli sovelluksessa on käyttäjiä, niin on käyttäjällä yleensä valtuutukset muokata omia tietojaan ja mahdollisesti jopa poistaa oma käyttäjätilinsä sovelluksesta.

Opinnäytetyön projektiin liittyvässä kirjanpitosovelluksessa käyttäjänhallintaan liittyy valtuutus, kun käyttäjä pääsee esimerkiksi luomaan ja muokkaamaan itselleen kirjanpitomerkintöjä. Roolipohjainen valtuutus on sovelluksen käytön myöntämistä käyttäjälle asetetun roolin kautta. Sovelluksessa voi olla monia toimintoja, joihin vain tiettyihin rooleihin asetetut käyttäjät pääsevät käsiksi, joka lisää tietoturvaa (Metatavu 2023).

Opinnäytetyön kirjanpitosovelluksessa on käyttäjärooleina työntekijä, asiakas, projektipäällikkö ja hallinnoija. Työntekijä pystyy merkitsemään kirjanpitoja projekteihin, johon hänet on liitetty. Hänellä ei kuitenkaan ole valtuutuksia lisätä itseään mihinkään projektiin. Työnjohtajan valtuutuksiin kuuluu kyky kirjata omia kirjanpitoja projekteihin, johon hänet on liitetty. Työnjohtajalla on myös valtuutukset lisätä itsensä ja muita työntekijä-roolissa olevia käyttäjiä projekteihin.

Attribuuttipohjainen kulunvalvonta (ABAC) on valtuutusmenetelmä, joka myöntää tai kieltää lupia sovelluksen osien käyttöön riippuen käyttäjälle asetetusta attribuutista. Tämä attribuutti voi olla käyttäjälle asetettu maantieteellinen alue, sovellusta käyttävä laite tai sovelluksen käytön ajankohta. Yleisin esimerkki ABAC-valtuutusmenetelmästä on ryhmä tai organisaatio, johon käyttäjä on liitetty. Esimerkiksi opinnäytetyön kirjanpitosovellusta käyttävä rajatuilla valtuutuksilla toimiva työntekijä pystyy vain luomaan kirjanpitomerkintöjä niihin projekteihin, joihin hänet on liitetty ja joihin hänelle on myönnetty valtuus kirjata uusia merkintöjä kirjanpitoon.

Sovelluksen käyttäjän valtuutukset kannattaa tehdä vähimpien oikeuksien periaatteella. Tämä tarkoittaa, että käyttäjälle annettaisiin valtuutukset vain pakollisiin resursseihin toimintojen suoritukseen. Sillä, että annetaan käyttäjälle minimaaliset tarvittavat valtuutukset, mahdollisesti vähennetään tietomurtojen vaikutusta ja vähennetään sovelluksen rajoitettujen alueiden luvatonta käyttöä (Buck, Chiedo, Lamos, Macy, Murray & Omondi, 2023). Vaikka normaalin käyttäjän tili olisi kaapattu, ei kaappaaja välttämättä pääse tuottamaan paljoa vahinkoa sovellukseen normaalin käyttäjän rajallisilla oikeuksilla.

2.3 Käyttäjänhallinta ja autentikointi

Käyttäjätilin turvallisuuden varmistamiseksi on hyvä suorittaa autentikointi eli todennus käyttäjälle ennen sovelluksen toimintojen käyttöä. Autentikointi on prosessi, jossa varmistetaan sovelluksen käyttöä yrittävän käyttäjän identiteetti. Autentikointi toiminnon kannattaa olla turvallinen, luotettava ja helppokäyttöinen. On monia tapoja suorittaa autentikointi sovelluksessa. Suurin osa sovelluksista kuitenkin käyttää yhdistelmää käyttäjänimi- ja salasana, monivaiheinen tunnistautuminen ja istunnon hallinta. (Metatavu 2023.)

Käyttäjänimen ja salasanan todennukset ovat yksiä yleisimpiä tapoja vahvistaa käyttäjän identiteetti. Sovellukseen kirjautuessa käyttäjiltä vaaditaan käyttäjänimi ja salasana, joita verrataan sovelluksen käyttäjien tietoja sisältävään tietokantaan ja tarkistetaan, täsmäävätkö käyttäjänimi ja salasana yhdessä jonkin käyttäjän tietoihin. Salasanat yleensä hajautetaan ja salataan valtuuttamattoman tiedon käsittelyn estämiseksi. Yksin käyttäjänimen ja salasanan todennuksen käyttö ei välttämättä ole vielä turvallista, sillä salasanat ja käyttäjänimet voivat usein vaarantua esimerkiksi, jos samaa salasanaa käytetään monessa eri sovelluksessa tai palvelussa.

Autentikoinnin varmuuden lisäämiseksi voidaan käyttää myös monivaiheista varmennusta. Monivaiheisessa varmennuksessa vaaditaan käyttäjänimen ja salasanan lisäksi toinen autentikointitapa. Yleisinä esimerkkeinä ovat sovelluksen luoma kertakäyttöinen salasana, joka yleensä toimitetaan käyttäjälle sähköpostina tai puhelimitse sekä erillinen autentikointisovellus. Hyvänä esimerkkinä erillisestä autentikointisovelluksesta on Microsoftin Authenticator -sovellus. Monivaiheisen varmennuksen lisääminen vähentää tietoturvariskiä huomattavasti, vaikka käyttäjän salasana olisi vaarantunut.

Istunnonhallinta voi myös olla tärkeä osa sovelluksen todennusta. Alustavan todennuksen jälkeen luodaan istunto, joka ylläpitää valtuutusoikeuksia ilman tarvetta manuaalisesti autentikoitua ennen jokaista toimintaa. Tietoturvallisuuden lisäämiseksi istunnon olisi hyvä tulla mitätöidyksi, jos käyttäjä on toimettomana tietyn ajan tai jos käyttäjä kirjautuu ulos.

3 KEYCLOAK

Keycloak on erillinen lisätyökalu. Se käyttää OpenIDConnect ja Oauth 2.0-protokollia käyttäjien hallintaan. Keycloak toimii yhdessä tai useammassa sovelluksessa Java-pohjaisesti.

3.1 Keycloak yleisesti

Keycloak on Red Hat -organisaation luoma Java-pohjainen avoimen lähdekoodin ohjelma, joka toimii valtuutus- ja todennustyökaluna moderneihin websovelluksiin. Se perustuu identiteetin- ja pääsynhallintaan (IAM) ja helpottaa keskittämällä valtuutuksen ja todennuksen yhteen sovelluksen osaan. Keycloak on suunniteltu toimivan monilla ohjelmistokehyksillä ja ohjelmointikielillä, kuten Java, Node.js ja Python. Sitä ei alun perin kuitenkaan ole suunniteltu käytettäväksi C#-kielellä tai .NET-ohjelmistokehyksissä. (Keycloak 2023.)

Yksi Keycloakin toiminnan hyödyllisiä osia on tuki yleisille valtuutusprotokollille kuten OpenID Connect (OIDC), OAuth 2.0 ja SAML 2.0. Keycloak pystyy näiden protokollien avulla integroimaan monia yleisiä tunnistetoimittajia kuten Google tai Facebook.

Arkkitehtuuriltaan Keycloak on rakennettu modulaarisesti ja laajennettavaksi. Se koostuu useista osista, kuten Keycloak-palvelimesta ja Keycloak-sovittimesta, joita voidaan mukauttaa ja laajentaa vastaamaan tiettyjä vaatimuksia. (Keycloak 2023.)

Keycloak-palvelin on järjestelmän ydinkomponentti ja tarjoaa todennus- ja valtuutuspalvelut. Se on toteutettu Javalla ja perustuu Wildfly-sovelluspalvelimeen. Keycloak-sovitin on kirjasto, jonka sovellukset voivat integroida Keycloak-palvelimeen.

3.2 Keycloakin rakenne

Keycloakissa on neljä keskeistä entiteettiä. Ne toimivat yhdessä käyttäjien henkilöllisyyksien ja käyttöoikeuksien hallintaan. Nämä neljä ovat alueet, asiakkaat, ryhmät ja käyttäjät.

Alue (Realm) on virtuaalinen säilö, joka eristää ja hallitsee käyttäjien identiteettejä ja resursseja. Keycloak-asennuksessa voi olla useita ulottuvuuksia, joista jokainen sisältää oman joukon käyttäjiä, rooleja ja ryhmiä. Jokaisella alueella on omat todennusasetukset, käyttäjät ja ryhmät. Alueet voidaan ajatella turvatoimialueina, jotka erottavat ja hallitsevat käyttäjien todennusta, valtuutusta ja resurssien hallintaa.

Asiakas on sovellus tai palvelu, jonka täytyy todentaa käyttäjät ja käyttää resursseja. Asiakkaat voivat olla verkkosovelluksia, mobiilisovelluksia tai muun tyyppisiä palveluita. Jokaisella asiakkaalla on omat käyttöoikeudet ja todennusasetukset, jotka määrittävät, kuinka käyttäjät voivat käyttää resursseja. Asiakkaat voidaan myös määrittää käyttämään erilaisia todennusprotokollia, kuten OpenID Connect tai Oauth 2.0.

Ryhmät ovat käyttäjien ryhmiä, joilla on yhteiset attribuutit tai roolit. Ryhmiä voidaan käyttää pääsynvalvontaan, käyttöoikeuksien delegointiin ja käyttäjien hallinnan yksinkertaistamiseen. Yritys voi esimerkiksi luoda ryhmän kaikille työntekijöille tai tietylle osastolle ja määrittää tälle ryhmälle erityisiä käyttöoikeuksia

Käyttäjät ovat henkilöitä, jotka todentavat ja käyttävät resursseja. Käyttäjiä voidaan luoda ja hallita alueen sisällä, ja ne voidaan määrittää eri ryhmiin ja rooleihin. Käyttäjiä voidaan myös todentaa useilla eri tavoilla, kuten käyttäjätunnuksella ja salasanalla, sosiaalisella kirjautumisella tai monitekijätodennusta käyttämällä.

4 OAUTH JA OIDC

OAuth 2.0 ja OpenID Connect (OIDC) ovat kaksi suosittua protokollaa, joita käytetään todentamiseen ja valtuutukseen verkossa. OAuth 2.0 on suunniteltu antamaan suojattu pääsy resursseihin paljastamatta käyttäjän tunnistetietoja. Sitä käytetään laajalti suojatun kolmannen osapuolen pääsyn mahdollistamiseen, ja sitä käytetään usein esimerkiksi sosiaalisen median kirjautumisissa tai API-valtuutuksessa.

OpenID Connect on OAuth 2.0:n laajennus, joka lisää protokollaan identiteettikerroksen, joka tarjoaa asiakkaille standardoidun tavan pyytää ja vastaanottaa tietoja loppukäyttäjästä. Yhdessä OAuth 2.0 ja OpenID Connect muodostavat vankan perustan turvalliselle ja yhteen toimivalle käyttäjänhallinnalle verkkosovelluksissa.

4.1 OAuth 2.0

OAuth 2.0 on valtuutuskehys, jonka avulla kolmannen osapuolen sovellukset voivat käyttää suojattuja käyttäjän resursseja ilman, että heidän tarvitsee tietää käyttäjän kirjautumistietoja. Tämä voidaan saavuttaa delegoimalla todennus- ja valtuutusvastuut valtuutuspalvelimelle. OAuth 2.0 sisältää useita osia, mukaan lukien resurssin omistaja, asiakas, valtuutuspalvelin ja resurssipalvelin. Omistajalla tarkoitetaan käyttäjää, joka omistaa suojatut resurssit. Asiakas on sovellus, joka haluaa käyttää suojattuja resursseja. Valtuutuspalvelin todentaa käyttäjän ja antaa käyttöoikeuksia, kun taas resurssipalvelin isännöi suojattuja resursseja. (Auth0 2022a.)

OAuth 2.0:n kulku sisältää useita vaiheita. Ensin asiakas lähettää valtuutuspalvelimelle valtuutuspyynnön, joka sisältää asiakkaan tunnistetiedot, pyydetyn käyttöoikeuden ja uudelleenohjaus-URI:n. Valtuutuspalvelin todentaa sitten käyttäjän ja pyytää hänen suostumustaan pääsyn myöntämiseen asiakkaalle. Käyttäjä joko myöntää tai kieltää pääsyn asiakkaalle. Mikäli käyttäjä myöntää käyttöoikeuden, valtuutuspalvelin antaa asiakkaalle

käyttöoikeustunnuksen. Asiakas sisällyttää käyttöoikeustunnuksen resurssipalvelimelle lähetettyihin pyyntöihin päästäkseen suojattuihin resursseihin. Resurssipalvelin vahvistaa käyttöoikeustunnuksen ja joko myöntää tai estää pääsyn suojattuihin resursseihin.

OAuth 2.0 tukee useita lupatyyppejä, jotka on suunniteltu tiettyihin käyttötapauksiin ja joilla on erilaisia turvallisuusvaikutuksia. Näitä aputyyppejä ovat valtuutuskoodilupa, implisiittinen lupa, resurssin omistajan salasanan käyttöoikeuslupa ja asiakkaan tunnistetietojen lupa. (Auth0 2022a.)

Kaiken kaikkiaan OAuth 2.0 on laajalti käytetty ja vankka valtuutuskehys. Tämän kehyksen avulla kolmannen osapuolen sovellukset voivat käyttää käyttäjän suojattuja resursseja käyttäjän suostumuksella. Samalla pystytään olemaan vaarantamatta hänen kirjautumistietojaan.

4.2 OpenID Connect

OpenID Connect (OIDC) on todennusprotokolla. Sen avulla sovellukset voivat tarkistaa käyttäjän henkilöllisyyden hänen kirjautumistietojensa perusteella. Todennusprotokolla rakentuu valtuutukseen tarkoitetun OAuth 2.0 -protokollan päälle.

OIDC määrittää ytimessä joukon päätepisteitä, joita kutsutaan OIDC-päätepisteiksi ja joita käytetään käyttäjän todentamiseen. Protokolla tarjoaa JSON-pohjaisen identiteettitunnisteen, joka sisältää käyttäjätietoja. OIDC:ssä on neljä pääkomponenttia: valtuutuspalvelin, asiakassovellus, käyttäjä ja resurssipalvelin. Valtuutuspalvelin on vastuussa käyttäjän todentamisesta sekä käyttöoikeus- ja tunnusmerkkien myöntämisestä. Asiakassovellus on sovellus, jota käyttäjä yrittää käyttää. Käyttäjä on henkilö, joka yrittää käyttää asiakassovellusta. Resurssipalvelin isännöi käyttäjän resursseja, joita asiakassovellus yrittää käyttää. (Auth0 2022b.)

OIDC:n kulku alkaa, kun käyttäjä yrittää käyttää asiakassovellusta. Asiakassovellus ohjaa käyttäjän valtuutuspalvelimen valtuutuspäätepisteeseen, jossa käyttäjän tulee todentaa oikeutensa käyttää sovellusta. Valtuutuspalvelin antaa sitten käyttöoikeus- ja tunnustunnisteet asiakassovellukselle. Asiakassovellus lähettää käyttöoikeustunnuksen resurssipalvelimelle pyytääkseen pääsyä käyttäjän resursseihin, ja resurssipalvelin varmistaa tunnuksen aitouden varmistaakseen, että asiakassovelluksella on oikeus käyttää käyttäjän resursseja. Lopuksi resurssipalvelin palauttaa pyydetyt resurssit asiakassovellukselle. (Auth0 2022b.)

Valtuutuspalvelimen myöntämä ID-tunnus sisältää tietoja käyttäjästä, kuten hänen nimensä ja sähköpostiosoitteensa. Asiakassovellus voi hyödyntää näitä tietoja tarjotakseen käyttäjälle henkilökohtaisen käyttökokemuksen.

Yhteenvetona tästä on mahdollista havaita, että OIDC tarjoaa suojatun ja standardoidun lähestymistavan sovelluksille käyttäjien todentamiseksi ja heidän resurssiensa käyttöön. Voidaan myös todeta, että OAuth 2.0 on protokolla, jolla myönnetään kolmansille osapuolille pääsy resursseihin. OpenID Connect on vastaavasti kätevä protokolla käyttäjien todentamiseen ja identiteettitietojen jakamiseen eri verkkosivustojen ja sovellusten välillä. (Auth0 2022a, Auth0 2022b.)

4.3 JSON Web Token

JSON Web Token (JWT) on suosittu menettelytapa tiedon siirtämiseen turvallisesti osapuolten välillä, kompaktin, URL-suojatun merkkijonon muodossa. JWT:itä voidaan käyttää todentamiseen, valtuutukseen ja tiedonvaihtoon ja niitä käytetään yleisesti nykyaikaisissa verkkosovelluksissa. Keycloak avoimen lähdekoodin identiteetin ja pääsynhallintaratkaisuna, tukee JWT:tä yhtenä tapana suojata verkkosovelluksia.

On tärkeää ymmärtää JWT:n perusrakenne, jos haluaa tietää, kuinka JWT:t toimivat Keycloakissa. JWT koostuu kolmesta osasta: otsikosta,

hyötykuormasta ja allekirjoituksesta. Otsikko sisältää JWT:tä koskevia metatietoja, kuten tunnuksen allekirjoittamiseen käytetyn algoritmin. Hyötykuorma sisältää todelliset siirrettävät tiedot, kuten käyttäjätiedot tai valtuutusvaatimukset. Allekirjoitusta käytetään varmistamaan, että JWT:tä ei ole peukaloitu lähetyksen aikana.

Keycloakissa JWT:eitä käytetään osana todennuskulkua. Kun käyttäjä kirjautuu sisään Keycloak-suojattuun sovellukseen, Keycloak luo JWT:n, joka sisältää tietoja käyttäjästä, kuten hänen käyttäjänimensä ja kaikki siihen liittyvät roolit tai käyttöoikeudet. Tämä JWT välitetään sitten sovellukselle, joka voi käyttää sitä käyttäjän todentamiseen myöhemmissä pyynnöissä.

Mikäli on tarve käyttää JWT:eitä Keycloakin kanssa, sinun on ensin määritettävä Keycloak antamaan JWT:t. Tämä edellyttää asiakkaan luomista Keycloakissa ja sen asetusten määrittämistä JWT-todennuksen mahdollistamiseksi. On myös mahdollista määrittää tietyt vaatimukset, jotka tulee sisällyttää JWT-hyötykuormaan, kuten käyttäjäroolit tai mukautetut attribuutit.

JWT-todennuksen käyttöönoton jälkeen on mahdollista käyttää Keycloakin todennussovellusliittymää JWT:n hankkimiseen käyttäjälle. Tämä edellyttää pyynnön tekemistä Keycloakin tunnuksen päätepisteelle käyttäjän tunnistetiedoilla, jotka palauttavat JWT:n, jos todennus onnistuu. JWT voidaan sitten välittää sovellukselle myöhemmissä pyynnöissä, joissa se voidaan tarkistaa allekirjoituksella varmistamaan, ettei sitä ole peukaloitu.

Todennuksen lisäksi Keycloak tukee myös JWT:iden käyttöä valtuutukseen. Tämä edellyttää valtuutusvaatimusten sisällyttämistä JWT-hyötykuormaan, jonka avulla sovellus voi määrittää, onko käyttäjällä oikeus käyttää tiettyjä resursseja tai suorittaa tiettyjä toimintoja.

Yhteenvetona voidaan todeta, että JSON-verkkotunnukset ovat tehokas ja joustava tapa suojata verkkosovelluksia, ja Keycloak tukee niitä osana todennus- ja valtuutuskulkuaan. Määrittämällä Keycloakin myöntämään JWT:itä ja käyttämällä sen todennussovellusliittymää niiden hankkimiseen, niin on

mahdollista helposti integroida JWT:t Keycloak-suojattuun sovellukseen ja tarjota turvallinen ja luotettava todennusmekanismi.

5 TOTEUTUS

5.1 Alustava työ

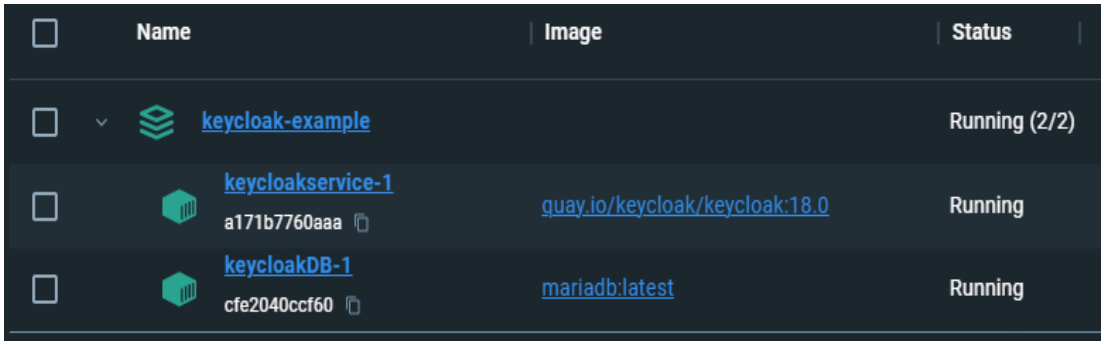
Työn aloituksessa pitää ensin asentaa Docker Desktop -ohjelma, jolla saadaan Keycloak -paketti toimimaan. Tätä varten kannattaa käyttää Docker-compose.yml tiedostoa (ohjelma 1).

```
version: '3.2'
services:
  keycloakDB:
    image: mariadb:latest
    ports:
      - "3406:3306"
    environment:
      MYSQL_ROOT_PASSWORD: keycloak
      MYSQL_DATABASE: keycloak
      MYSQL_USER: keycloak
      MYSQL_PASSWORD: keycloak
    volumes:
      - demo_keycloakdata:/var/lib/mysql
  keycloakservice:
    image: quay.io/keycloak/keycloak:18.0
    ports:
      - 8080:8080
    command:
      - start-dev
    environment:
      KEYCLOAK_ADMIN: keycloak
      KEYCLOAK_ADMIN_PASSWORD: keycloak
      KEYCLOAK_USER: user
      KEYCLOAK_PASSWORD: password
volumes:
  demo_keycloakdata:
```

Ohjelma 1. yml-tiedosto Keycloak paketille.

Tällöin on mahdollista saada Keycloak-sovellus toimimaan testiympäristössä avaamalla terminaalin tai päätteen Docker-compose.yml -tiedoston

ympäristössä. Tämän jälkeen suoritetaan komento ”docker compose up” tai ”docker compose -f Docker-compose.yml up”. Seuraavaksi on mahdollista havaita Docker Desktop- sovelluksesta, että ovatko Keycloak -kontit käynnistyneet (kuva 1).



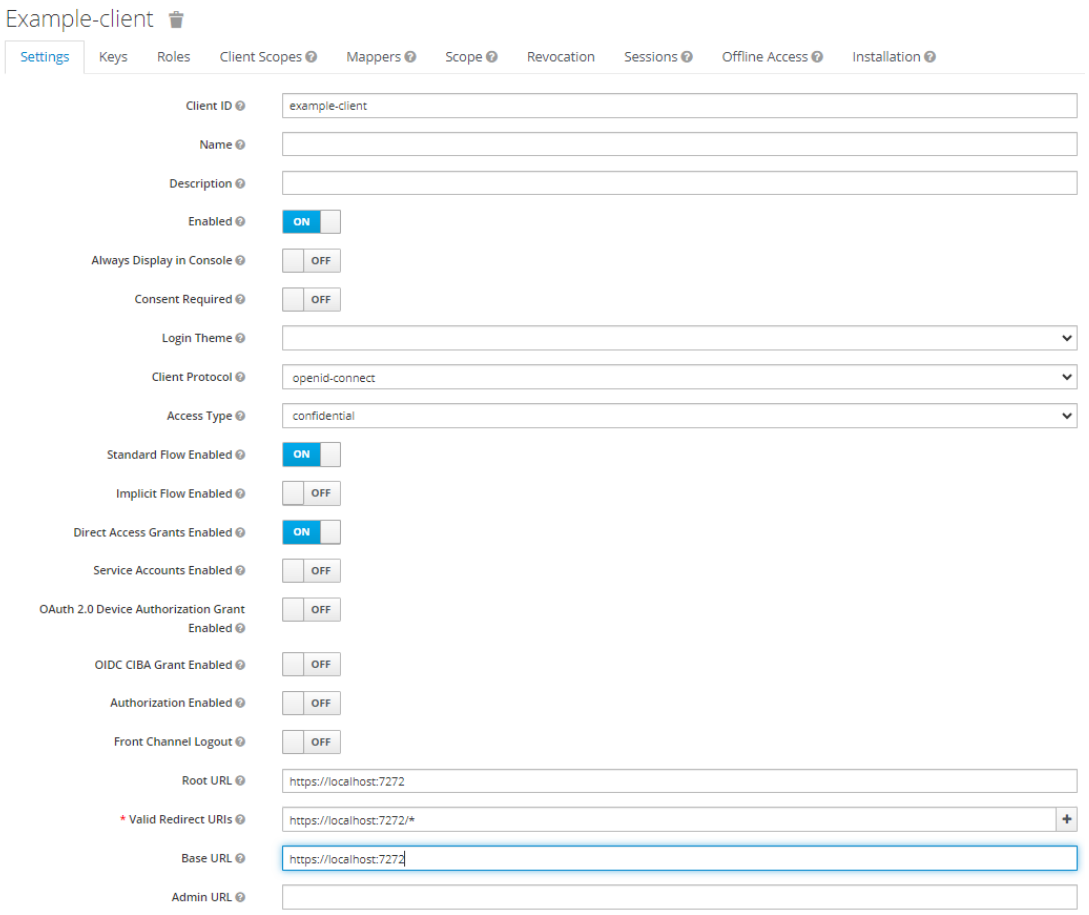
Kuva 1. kuvakaappaus Docker Desktop -sovelluksesta, jossa nähdään keycloak paketin olevan päällä

5.2 Keycloak-asetukset

Nyt päästään sisälle Keycloakin ylläpitäjäpaneeliin menemällä paikalliseen osoitteeseen ”localhost” tai 127.0.0.1 ”docker-compose” tiedostossa asetetun portin kautta. Tässä tilanteessa se on 8080. Keycloakin hallintapaneelin osoite on siis http://localhost:8080 tai http://127.0.0.1:8080. Tässä osoitteessa kirjaudutaan Keycloakin hallintapaneeliin Docker-compose.yml -tiedostossa ilmaistulla ylläpitäjän käyttäjätunnuksella ja salasanalla, jotka ovat esimerkissä keycloak ja keycloak.

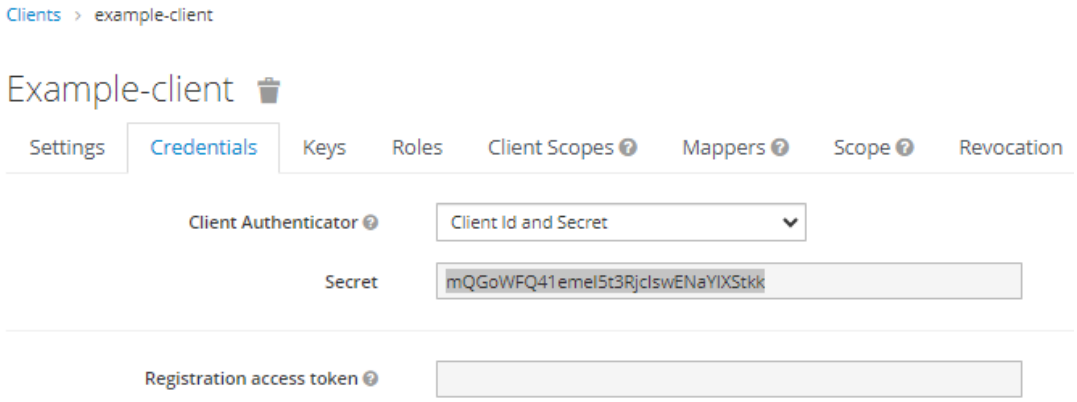
Sisäänkirjautumisen jälkeen on mahdollista aloittaa luomalla sovellukselle oma alue menemällä hiirellä ”Master” -alueen päälle ja painamalla ”Add realm” - nappia. Aluetta luodessa täytyy asettaa sille vain nimi, joka tässä esimerkissä tulee olemaan ”example-realm”, ja varmistaa, että alue on käytössä. Tämän jälkeen luodaan uuteen alueeseen asiakas menemällä ”Clients” välilehdelle ja painetaan ”Create” -nappia. Client ID:ksi asetetaan esimerkissä ”example-client”.

Tämän jälkeen muokataan asiakkaan asetuksia (kuva 2). Aloitetaan asettamalla ”Access Type”:n arvoksi ”confidential” ja varmistetaan ”Client Protocol” olevan OpenIdConnect. ”Root URL” asetetaan sovelluksen osoitteeksi ja ”Valid Redirect URIs” kohtaan kirjoitetaan kaikki sovelluksen alla olevat osoitteet. Tässä esimerkissä nämä arvot olisivat ”https://7272” ja ”https://7272/*”. (Keycloak 2022.)



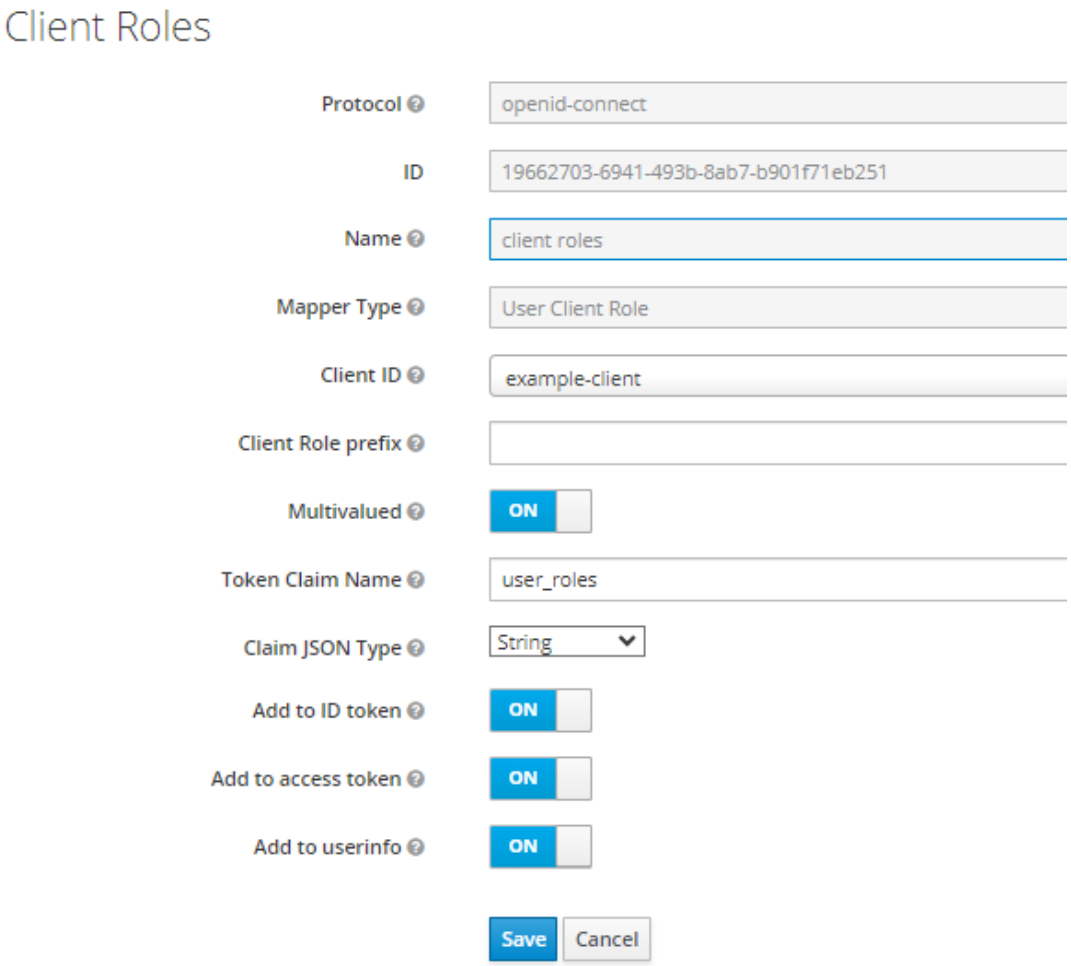
Kuva 2. kuvakaappaus asiakkaan asetus -sivusta Keycloak hallintapaneelissa.

Käydään lisäksi ”Credentials” -välilehdellä ja otetaan ”Secret” -arvo talteen sovellusta varten, joka kuvassa 3 on maalattuna. ”Secret” tai sala-avain muodostuu Keycloakin generoimana automaattisesti ja jota käytetään käyttäjän kutsun autentikoinnin varmistamiseen.



Kuva 3. Asiakkaan ”Secret” on maalattuna harmaalla.

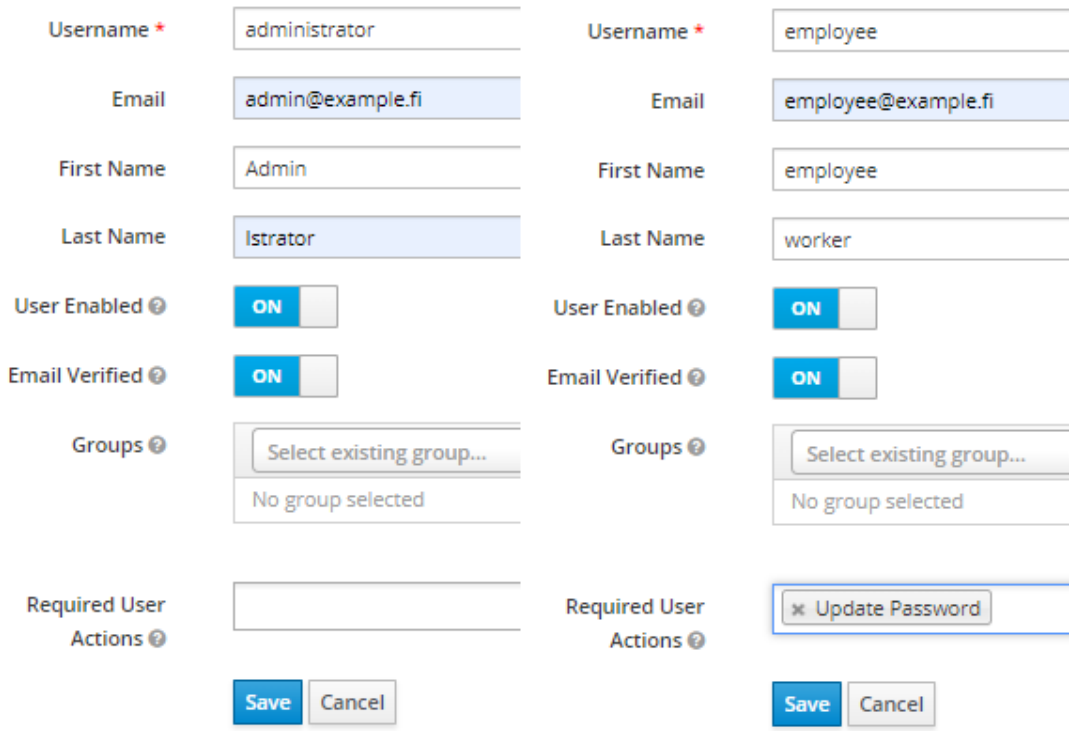
Sovelluksen käyttäjille tarvitaan roolit. Ne voidaan tehdä asiakkaan ”Roles” -välilehdellä. Painamalla ”Add Role” -nappia, voidaan luoda rooli, jolle tarvitaan nimi ja voidaan antaa kuvaus. Esimerkissä halutut roolit ovat ”Admin”, ”Manager”, ”Client” ja ”Employee”. Roolien luonnin jälkeen pitää rooleille luoda mapper, joka asettaa roolin mukaan Keycloakin väitteisiin. Asiakkaan ”Mappers” -välilehdellä voidaan luoda uusi kartoittaja painamalla ”Add Builtin” -nappia. Luodaan kartoittaja luoduille rooleille valitsemalla listasta ”client roles” -nimellä oleva rivi. Tätä joudutaan vielä muokkaamaan ”Edit” -nappia painamalla. Kuvan 4 tavoin, kartoittajan ”Client ID” asetetaan luodun asiakkaan ID:ksi, ”Add to ID token” asetetaan muotoon ”ON” kuten samoin myös ”Add to access token” ja ”Add to userinfo”. ”Token Claim Name” -arvoksi asetetaan tässä esimerkissä ”user roles”, jotta käyttäjän rooleihin päästään helpommin käsiksi sovelluksen puolella.



Kuva 4. roolien kartoittajan muokattavat asetukset.

Seuraavaksi käydään läpi Keycloak -käyttäjän luonti. Admin-paneelin ”Users” -välilehdeltä päästään luomaan uusia käyttäjiä ”Add user” -napin kautta. Esimerkkiä varten tehdään käyttäjä, jonka arvot nähdään kuvassa 5. Käyttäjänimi on ”administrator”, sähköpostina ”admin@example.fi”, etunimenä ”Admin”, sukunimenä ”Istrator” ja ”Email Verified” -vaihtoehto asetetaan päälle. Kuvan 5 tavoin luodaan esimerkkiä varten myös toinen käyttäjä ”employee”, jolle annetaan tiedot: käyttäjänimi ”Employee”, sähköposti ”employee@example.fi”, etunimenä ”employee”, sukunimenä ”worker” ja ”Email Verified” -vaihtoehto päälle. Käyttäjältä voidaan myös vaatia varmistamaan sählöpostinsa tai vaihtamaan salasanansa, kun käyttäjä ensi kertaa koittaa kirjautua tunnuksillaan. halutut vaaditut toiminnot voidaan lisätä kenttään ”Required User Actions”. Jos halutaan lisätä käyttäjälle uusi kertakäyttöinen tai pysyvä salasana, voidaan mennä ”Credentials välilehdelle, lisätä uusi salasana ”Password” kenttään ja

asettaa se joko väliaikaiseksi tai vakituiseksi ”Temporary” -kentän kanssa. (Keycloak 2022.)



Kuva 5. Vasemmalla tulevan ”administrator” -käyttäjän tiedot luodessa ja oikealla ”employee” -käyttäjän tiedot. ”employee”-käyttäjän pitää päivittää salasana ensi kirjautumiskerralla.

Käyttäjien luonnin jälkeen asetetaan käyttäjät rooleihin menemällä ”Role Mappings” -välilehdelle, valitsemalla ”Client Roles”:iin luotu client, valitaan halutut roolit ja lisätään ne käyttäjille painamalla ”Add selected” -nappia. Esimerkissä lisätään ”administrator” -käyttäjälle ”Admin” -rooli ja ”employee” -käyttäjälle ”Employee” -rooli example-client” asiakkaasta. Tämän jälkeen tarvittavat muutokset ja lisäykset Keycloakin hallintapaneelissa on tehty.

5.3 .NET Core

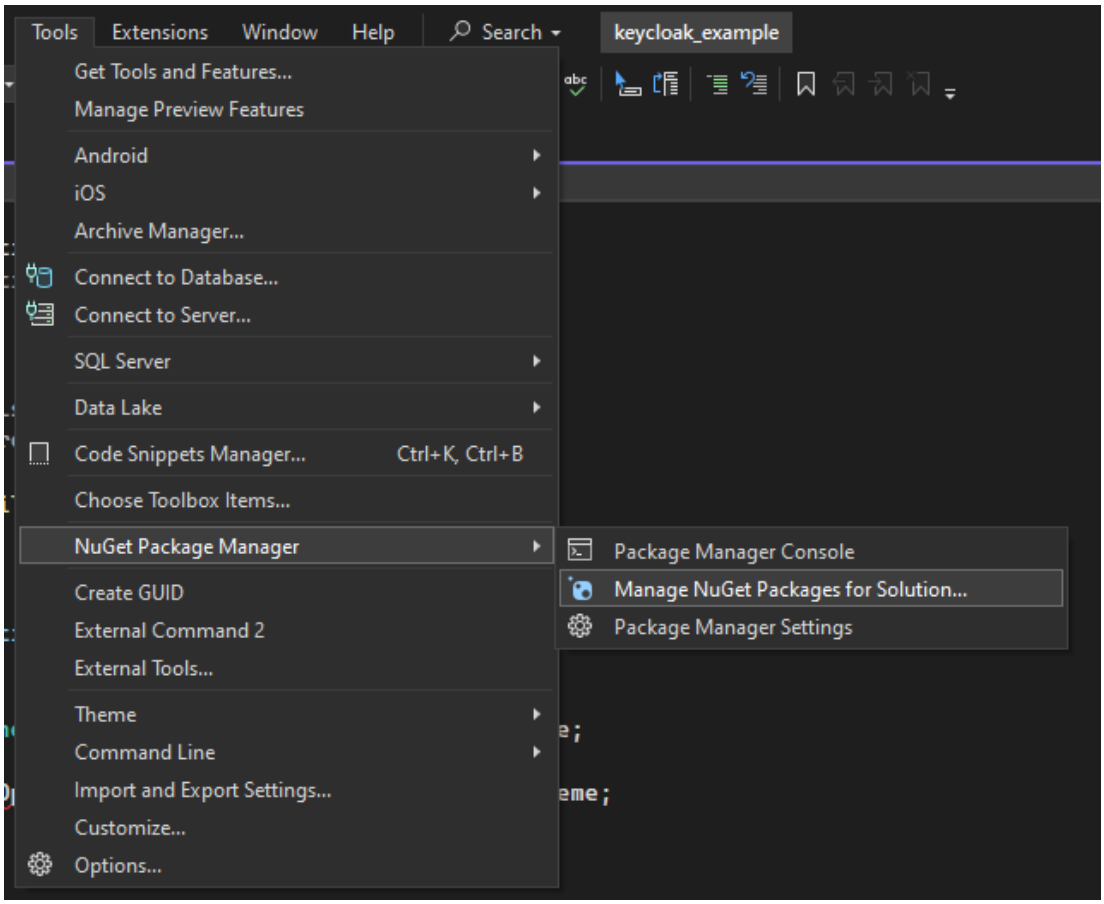
.NET Core -sovelluksen puolella luodaan yhteys Keycloakiin käyttämällä ympäristömuuttujia, jotka tullaan asettamaan LaunchSetting.json -tiedostossa ohjelma 2:n tapaan. Kyseessä olevat ympäristömuuttujat ovat

KEYCLOAK_ENDPOINT, KEYCLOAK_CLIENT_ID ja KEYCLOAK_CLIENT_SECRET. Ensimmäiseen muuttujaan asetetaan Keycloak -alueen rajapinta. Toinen muuttuja on alueessa olevan asiakkaan ID. Kolmas on asiakkaan sala-avain.

```
"profiles": {
  "Example_Project": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "applicationUrl": https://localhost:7272,
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development",
      "KEYCLOAK_ENDPOINT": "http://lo-
calhost:8080/realms/example-realm",
      "KEYCLOAK_CLIENT_ID": "example-client",
      "KEYCLOAK_CLIENT_SECRET": "mQGoWF41emel5t3Rjcls-
wEaYIXStkk"
    }
  }
}
```

Ohjelma 2. asetetaan ympäristömuuttujat sovelluksen käynnistysasetuksiin.

Jotta voidaan saada yhteys Keycloakiin, niin tulee ladata myös Nuget -paketti OICD-protokollan käyttöön. Haetaan Visual Studion työkaluista ”Nuget Package Manager”:in vaihtoehto ”Manage NuGet Packages for Solution” -työkalu (kuva 6). Tämän jälkeen haetaan ja ladataan haluttu paketti, joka on ”Microsoft.AspNetCore.Authentication.OpenIdConnect”.



Kuva 6. kuvakaappaus, josta nähdään, mistä Nuget -pakettien hallinointityökalu löytyy.

Seuraavaksi käytetään ympäristömuuttujia Program.cs -tiedostossa, jotta on mahdollista saada sovelluksen käynnistyksessä yhteys Keycloak -sovellukseen. Sovelluksen luontiasetuksiin lisätään OpenId Connect -asetukset, jossa asetetaan LaunchSettings.json -tiedoston ympäristömuuttujat kuten kuvassa näkyy. Asetetaan myös valtuutusparametreihin Keycloakissa kartoitettu ”user_roles” -attribuutti roolivaatimukseksi (ohjelma 3).

Tämän OpenIdConnect -asetuksen lisäämisen jälkeen lisätään sovelluksen rakennuksen jälkeen vielä asetus ” UseAuthorization”. Nyt kontrollerissa voidaan lisätä ”Authorize” -attribuutti kontrollerin metodeihin tai jopa kontrolleriin itseensä. Ohjelma 4 tavoin, näihin attribuutteihin voidaan myös tarkentaa haluttu rooli. (Hahn 2017)

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddOpenIdConnect(options =>
{
    options.Authority = Environment
         .GetEnvironmentVariable("KEYCLOAK_ENDPOINT");
    options.ClientId = Environment
         .GetEnvironmentVariable("KEYCLOAK_CLIENT_ID");
    options.RequireHttpsMetadata = false;
    options.SaveTokens = true;
    options.ClientSecret = Environment
         .GetEnvironmentVariable(
                  "KEYCLOAK_CLIENT_SECRET");
    options.GetClaimsFromUserInfoEndpoint = true;
    options.ResponseType = OpenIdConnectResponseType
         .CodeIdToken;
    options.TokenValidationParameters = new Microsoft
         .IdentityModel.Tokens.TokenValidationParameters
    {
        NameClaimType = "email",
        RoleClaimType = "user_roles",
    };
    options.Events
         .OnSignedOutCallbackRedirect += context =>
    {
        context.Response.Redirect(
         context.Options.SignedOutRedirectUri
         );
        context.HandleResponse();
        return Task.CompletedTask;
    };
});
```

Ohjelma 4. Sovellukseen lisätään OpenIdConnect asetukset, joissa käytetään luotuja ympäristömuuttujia

Nyt joka kerta, kun siirrytään kontrolleriin tai metodiin, jossa on ”Authorize” -attribuutti kuten ohjelma 5:ssä, niin joudutaan Keycloakin sisäänkirjautumissivulle. Kirjautumisen jälkeen pääsee takaisin osoitteeseen, josta sisäänkirjautumissivulle tultiin.

```
[ApiController]
[Route(“[controller]”)]
[Authorize(Roles = “Admin”)]
public class UserController : ControllerBase
{
        public ActionResult Index()
        {
                Return View();
        }
}
```

Ohjelma 5. kontrolleri ”userController”, jonka metodeihin päästään vain ”Admin” -roolilla.

Sovellukseen on mahdollista liittää personoituja asioita kuten tuntimerkintöjä. Tällöin voi liittää Keycloak-käyttäjän mahdolliseen omaan tietokantaan lisäämällä tietokantakäyttäjään kentän, jossa Keycloakin ID:tä varastoidaan. Tämän jälkeen on mahdollista hakea nykyinen tietokanta -käyttäjä kontrollerista nykyisen käyttäjän väitteistä. Väitteen avain on tyyppiä ClaimTypes.NameIndentifier ja väitteen arvoa verrataan tietokannan käyttäjän Keycloak ID:n kenttää vasten.

```
protected DBUser  GetCurrentUser()
{
        string? keycloakID = User.Claims
                .FirstOrDefault(c =>
                c.Type == ClaimTypes
                        .NameIdentifier)?.Value;
        var dbUser = new Repository<DBUser>(_context)
                .Select.FirstOrDefault(x =>
                x.KeycloakId == keycloakID);
        return dbUser;
}
```

Ohjelma 6. metodi, jolla haetaan käyttäjän Keycloak ID ja sen avulla haetaan sille sopiva tietokantakäyttäjä.

Näiden askelien jälkeen on Keycloak onnistuneesti liitetty .NET Core -sovellukseen. Nyt sovelluksen osoitteeseen mentäessä tarvitsee käyttää ja kirjautua sisään Keycloak-tunnuksilla.

6 LOPPUPÄÄTELMÄ

Pääsin opettelemaan opinnäytetyössä laajasti Keycloakin käyttöä sekä kirjoittamaan ammatillisesti. Tämän projektin pohjalta minulle muodostui opinnäytetyön tekijänä näkemystä Keycloakista työkaluna. Mielestäni Keycloak on vahva ja hyödyllinen työkalu käyttäjänhallintaan, jota voi käyttää myös .Net Core -pohjaisissa sovelluksissa yllättävän pienellä vaivalla. Keycloakin hallintapaneeli on suurimmaksi osaksi intuitiivinen ja helppokäyttöinen ja käyttäjänhallinta onnistuu myös monen eri sovelluksen välillä.

.NET Core -sovelluksessa OpenID Connect -protokollan implementointi voi vaikuttaa hankalalta ja epäselvältä, mutta se käy helposti. Autentikointi saatiin toimimaan erittäin helposti. Myös Keycloak -käyttäjien liittäminen mahdolliseen sovelluksen tietokantaan ei vaadi paljoa enempää kuin yhden kentän lisäämisen tietokantakäyttäjään, missä säilytetään Keycloak -käyttäjän ID, mikä yllätti positiivisesti.

Opinnäytetyön kirjoittaminen sekä teoriatiedon tuottaminen projektista osoittautui haastavimmaksi osaksi opinnäytettä. Kehityin opinnäytetyöprojektin aikana ammatillisessa kirjoittamisessa. Koen, että opinnäytetyötä tehdessäni sain tukea lähipiiriltäni, mutta myös runsaasti taholta, jolle opinnäytetyöprojektini tein.

Jatkokehitystä varten voisi Keycloakin asetukset asettaa erilliseen JSON-tiedostoon. Tällöin olisi mahdollista muokata myös Keycloakin hallintapaneelin ja sisäänkirjautumissivun ulkonäköä, vaikkei tässä opinnäytteessä sitä

tehtykään. Toimimalla näin Keycloak sovelluksen ulkonäkö pysyisi yhdenmukaisena myös sisäänkirjautuessa.

LÄHTEET

Auth0. (2022a). OAuth 2.0 Authorization Framework. Haettu 20.10.2022 osoitteesta https://auth0.com/docs/authenticate/protocols/oauth

Auth0. (2022b). OpenID Connect Protocol. Haettu 16.10.2022 osoitteesta https://auth0.com/docs/authenticate/protocols/openid-connect-protocol

Buck A., Chiedo C., Lamos B., Macy M., Murray D. & Omondi J. (23.10.2023). Enhance security with the principle of least privilege. Microsoft dokumentaatio. Haettu 10.11.2023 osoitteesta https://learn.microsoft.com/en-us/entra/identity-platform/secure-least-privileged-access

Hahn X. (13.11.2017). Adding authorization to Asp.Net Core app using Keycloak. Haettu 16.10.2022 osoitteesta https://medium.com/@xavier.hahn/adding-authorization-to-asp-net-core-app-using-keycloak-c6c96ee0e655

Keycloak. (2022). Docker Get started with Keycloak on Docker. Haettu 16.10.2022 osoitteesta https://www.keycloak.org/getting-started/getting-started-docker

Keycloak. (2023). Etusivu. Haettu 28.2.2023 ositteesta https://www.keycloak.org/

Metatavu (2023). Keskitetty identiteetin- ja pääsynhallinta vähentää vaivaa ja parantaa tietoturvaa. Haettu 7.8.2023 osoitteesta https://metatavu.fi/keskitetty-identiteetin-ja-paasynhallinta-vahentaa-vaivaa-ja-parantaa-tietoturvaa/

Tietosuojalaki 1050/2018. Haettu 4.10.2023 osoitteesta https://www.finlex.fi/fi/laki/ajantasa/2018/20181050