Kauno technologijos universitetas

Informatikos fakultetas

Eligijus Kiudys

Rolėmis grįstas žiniatinklio programų prieigos valdymo metodas

Analizė

Vertino:

prof. Algimantas Venčkauskas

Kaunas 2022

Turinys

Paveikslėlių sąrašas

Įžanga

Rolėmis grįstas prieigos valdymo metodas yra vienas iš populiariausių metodų esančių, kurie skirti valdyti internetinių puslapių arba internetinių aplikacijų prieigą. Žinoma yra ne vienas metodas skirtas prieigos valdymui, bet dažniausiai tai būna minėto metodo variacijos arba visiškai skirtingas metodas. Prieigos valdymo problema išlieka ir šiai dienai. Vis daugiau ir daugiau plečiantis internetui ir didėjant tiekiamų paslaugų kiekiui internete reikia ir saugesnių ir patogesnių metodų valdyti internetines aplikacijas, puslapius. Rolėmis grįstas žiniatinklio programų prieigos valdymo metodas leis panaudoti rolėmis grįstą prieigos metodą žiniatinklio programoms. Kadangi žiniatinklio programų kiekis didėja reikia ir metodo, kuris leistų daugiai valdyti naudotojo prieigą prie programos funkcijų ir domenų. Metodas kurį analizuoju yra pagrindinis metodas prieigos valdymui. Analizės metu yra analizuojamos prieigos valdymo problemos, išsiaiškinama apie kitus esamus metodus.

Tikslas ir uždaviniai

Darbo tikslas – sukurti rolėmis grįstą prieigos valdymo metodą skirtą internetinėms programoms. Metodas turėtu leisti pritaikyti rolėmis grįstą prieigos valdymo metodą internetinių aplikacijų prieigos valdymui. Panaudojus naują metodą administratorius galės keisti roles, licencijas, naujoms internetinėms aplikacijoms.

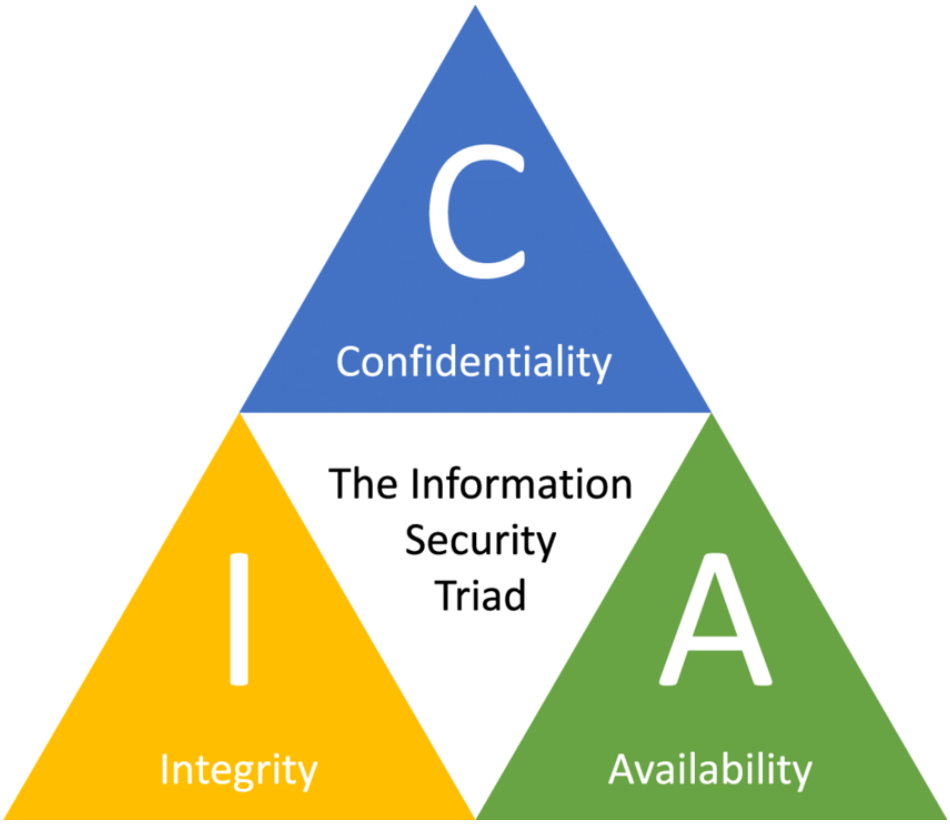
- Išanalizuoti žiniatinklio programų prieigos valdymo problemos.
- Išanalizuoti žiniatinklio programų prieigos valdymo metodus.
- Išanalizuoti rolėmis grįstas prieigos valdymo metodas.
- Išanalizuoti rolėmis grįsto prieigos valdymo metodo pritaikymą žiniatinklyje.
- Pasiūlyti rolėmis grįstą prieigos valdymo metodą skirtą žiniatinklių programoms.
- Įgyvendinti sukurtą metodą ir palyginti rezultatus.

Dokumento struktūra

Darbą sudaro keturi pagrindiniai skyriai - žiniatinklio programų prieigos valdymo problemos, žiniatinklio programų prieigos valdymo metodus, rolėmis grįstas prieigos valdymo metodas, rolėmis grįsto prieigos valdymo metodo žiniatinklyje modelį analizės skyriai. Pirmame skyriuje yra analizuojamos žiniatinklio programų prieigos valdymo esamos problemos. Kitame skyriuje yra analizuojama skirtingi metodai skirti valdyti žiniatinklio programų prieigą. Trečiame skyriuje yra analizuojamas rolėmis grįsto prieigos valdymo metodą, kaip veikia minimas metodas ir šiek tiek istorijos apie metodą. Ketvirtame skyriuje yra analizuojami rolėmis grįsto prieigos valdymo metodo žiniatinklyje.

Žiniatinklio programų prieigos valdymo problemos

Prieigos valdymo problema egzistuoja ne vienerius metus. Ar tai būtu internete ar bibliotekoje, ar universitete. Visą laiką reikia verifikuoti, kad esi tikrai tu, vienokiu ar kitokių būdu. Realybėje yra naudojamos ID kortelės arba pasas. Realybėje galima pamatyti paso nuotrauką ir žmogaus veidą, taip verifikuojant, kad jis tikrai tas žmogus. Internetinėse aplikacijose būtu lengva verifikuoti tapatybę su ID kortele arba pasu, bet juos perkelti į elektroninę erdvę nėra labai lengva. Internetinėse aplikacijos neužtenka patvirtini naudotojo tapatybę, bet reikia valdyti naudotojo prieigą vienu ar kitu būdu. Problemai spręsti visos internetinės aplikacijos ir net tik naudoja pasirinktą modelį naudotojų prieigos valdymui. Kiekvienas modelis turi savus pliusus ir minusus. Sukurti ar panaudoti esamą prieigos valdymo modelį yra sunki užduotis, kuri kankina interneto aplikacijų kūrėjus ne vienerius metus. Naudojamo prieigos modelio užduotis yra paprasta, prileisti naudotojus prie jiems galimos informacijos, nei daugiau, nei mažiau.



pav. 1 prieigos valdymo modelio problemos

- Konfidencialumas
- Vientisumas
- Prieinamumas

Prieigos valdymas (Separation of Duty)

Viena iš pirmujų problem yra naudotojų prieigos atsikyrimas. Šia problem bando išspręsti kiekvienas prieigos saugos modelis. Problema atsirado, kai keletas naudotojų pradėjo naudotis tokiu pačiu kompiuteriu, tada iškilo klausimas. Kaip galima atskirti naudotojo prieinamus resursus kiekvienam naudotojui, pavyzdžiui, kokias aplikacijas gali naudoti naudotojas, o kokių negali. arba kokius failus gali redaguoti, o kokiu negali. Minėta problema išlieka ir dabar. Naudojant internetines sistemas reikia atskirti teikiamus resusrsus pagal naudotojo prieigą.

Dinamiškumas

Pradėkime nuo pirmos problemos: modelio dinamiškumo. Žinoma pasirinkus prieigos valdymo modelį pradžioje, atrodo, kad gali tikti, bet kuris pasirinktas modelis. Padarius projektą ir administratoriui sukonfigūravus teises naudotojams atrodo viskas veikia kaip tūrėtu veikti. Pradedant plėsti esamą sistemą kūrėjai gali susidurti prieigos valdymo problemomis, kadangi ne visi modeliai leidžia dinamiškai plėsti projektą. Plečiant dinamišką sistemą su blogu prieigos valdymo metodu, dažniausiai yra pradedami naudoti įvairūs apėjimai, kad sistema veiktų. Tokios problemos sistemos administratoriams apsunkina darbą, kadangi realizacija yra neintuityvi, administratoriai valdant tokį projektą gali pridaryti klaidų, kurios veda prie kitų prieigos valdymo problemų.

Vientisumo problema tarp sistemu

Prisijungimas prie sistemos per kitus puslapius. Naudojantis internetinėmis aplikacijomis galima prisijungti prie sistemos naudojant kitą sistemą. Dažnai iškyla įvairios problemos prisijungiant su kita sistema, priklausant nuo pasirinkto prieigos valdymo modelio. Viena iš pagrindinių problemų yra naudotojo teisės. Tarp naudojamų metodų turi būti sudarytas susitarimas, kaip koks prieigos valdymo metodas su kitu metodu, jei norima prisijungti prie skirtingu internetiniu sistemų su viena paskyra.

Modelio valdymas

Modelio valdymas yra dar viena problema su kuria susiduriame pritaikant pasirinktą prieigos valdymo metodą. Pritaikius pasirinktą prieigos valdymo metodą, administratorius dažniausiai turi valdyti naudotojų dalinę arba pilną prieigą. Dažnai neintuityvus modelio valdymas gali privesti prie administratoriaus klaidų, kurios suteikia naudotojams prie mažai teisių arba per daug. Žiūrint iš rolėmis grysto ar kitokio saugos modelio perspektyvos, visada gali atsirasti valdymo prieigos spragų. Paemus pavizdy kaip atributais paremta prieigos valdyma, panadudojant blogą contekstą implementacijos metu, gali atsirasti prieigos valdymo spragos, naudotojas kuris neturi teises prieiti prie informacijos, ją gali pasiekti.

Modelio valdymo problema (Role explosion),

Kiekviena sistema turi saugumo spragų, ar tai būtu pritaikymo problemos, ar tai būtu administravimo spragos, ar kažkokios kitos klaidos kurios priveda prie nesaugios sistemos. Vienas iš pavyzdžių būtu rolėmis paremto prieigos metodo rolių sprogimo problema. Šitame pavyzdyje yra aprašoma kaip rolių kiekis didėja iki tokių skaičių kai jų nebegalima suvaldyti. Pavyzdžiui yra vienas naudotojas, dešimt sistemų ir dvi rolės per sistemą. Naudotojas reikalauja dviejų rolių per aplikaciją gaunasi taip, kad reikalauja dvidešimt rolių. Turint daugiau negu vieną naudotoją ir jiems visiems prašant dviejų rolių per aplikaciją, tikėtina kad tvarkant roles administratorius gali įvelti ne vieną klaidą.

User Authorization Query

Problema, kuri aprašo prieigos reikalavimo problemą, kaip galima efektyviai pareikalauti teisę į tam tikrą sistemą ar resursą. Ši problema ypač iškyla naudojant rolėmis grystą metodą. Norint gauti papildomą prieigą prie resursų reikia nuspręsti kokią rolę reikia suteikti, kad butu suteikti tam tikri resursai, ar tai butu galima pakeisti esamas roles i kažkokią vieną rolę.

Sudėtingumas

Sistemos sudėtingumas irgi yra problema. Norint integruoti pasirinktą prieigos valdymo metodą, sistemos sudėtingumas gali kišti koją. Programuotojai gali nesuprasti sudėtingos sistemos, ją pritaikant gali padaryti saugos klaidų. Žinoma administratoriui bus lengviau dirbti, bet kartai geriau yra pasirinkti sunkiau valdomą sistemą, bet lengviau suprantamą ir įdiegiamą.

Momentinė prieiga prie informacijos

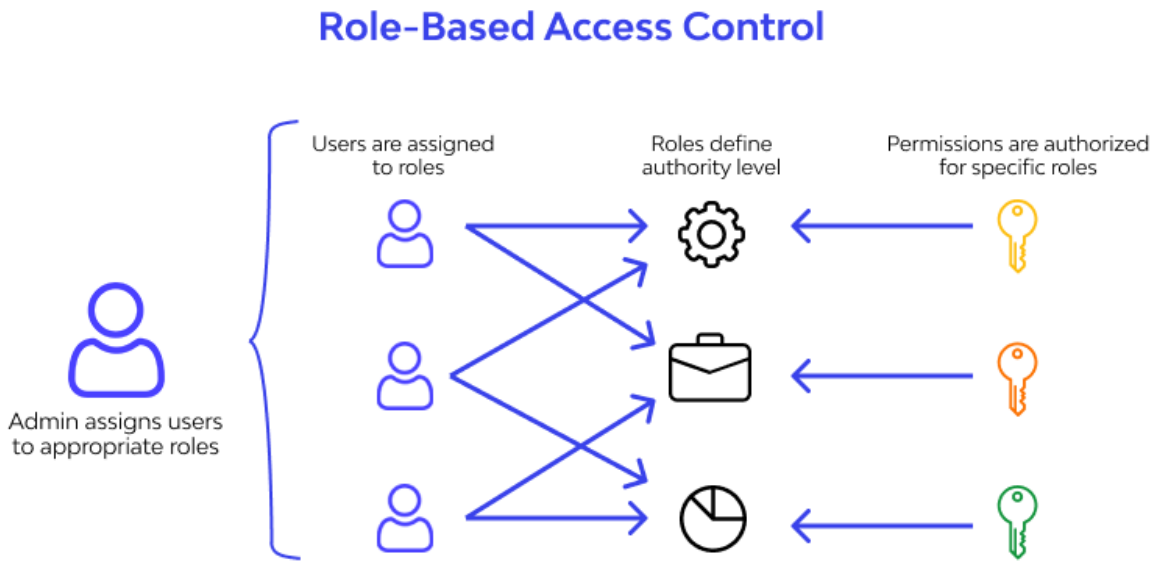
Paprastas prieigos valdymo saugos modelis, nesprendžia reikiamos momentinės prieigos problemos, kai naudotojui reikia prieigos prie sistemos vienam ar keliesm kartams ir poto jos nebereikia. Atsiranda problemos kaip reikėtu suteikti tokią prieigą, kad nebutu galima prieiti prie kitų resursų. Kaip galima efektyviai suteikti naudotojui tokią galimybę ir kaip užtikrinti, kad naudotojas tikrai gali prieiti prie prašomų resursų.

Žiniatinklio programų prieigos valdymo metodai

- Object-Specific Role-Based Access Control (ORAC)[2]
  Konkrečių objektų rolėmis pagrystąs prieigos valdymo metodas. Šitas metodas valdo naudotojų prieigą rolėmis ir specifiniais objektais. Administratorius gali nustatyti naudotojui rolę, arba specifinį objektą, prie kurio naudotojas gali prieiti.
- Attribute-based access control[3]
  Naudojant atributais pagrįsto prieigos valdymo metodo prieiga prie išteklių gali būti nustatyta pagal skirtingus požymius pvz. vardas, IP adresas, laikas ir tt. Pagrindinė idėja ABAC yra, kad nėra tiesiogiai priskirti leidimų vartotojui, dėl to objektų prieiga leidžia visų objektų prieigą remiantis atributais. Atributas vaidina svarbų vaidmenį sistemoje ABAC leidimų suteikimas įgaliotiems vartotojams, pvz., vardas, vieta, IP adresas, vieta ir tt. Atributo reikšmė nusprendžia, ar vartotojas yra įgaliotas naudoti tam tikrą ištekli, ar ne. Vartotojas taip pat gali būti nurodyta kaip subjektas.
- Role-based Access Control[4]
  Pagrindinė RBAC koncepcija yra rolės, kurios gali parodyti prieigos kontrolės politiką tam tikrai organizacijai, institucijai ar įmonei. Leidimai sukuriami, kai objektai atliekami veiksmai, o po to šioms rolės priskiriami leidimai. Vartotojai nėra tiesiogiai priskirtas leidimams. Rolė yra tiltas tarp leidimų ir vartotojai. Vartotojams priskiriami nurodytos rolės, kad jie galėtų naudotis skirtingais leidimais.
- Attributed Role Based Access Control Model[5]
  Modelis naudoja roles kaip tiltą tarp objekto leidimų ir naudotojo. Administratorius naudotojui nustato pasiriktą rolę. Šitas modelis skiriasi nuo paprasto rolėmis pagrįsto prieigos metodo tuo, kad objektų leidimai yra priskiriami automatiškai prie rolių, naudojant atributais paremto prieigos valdymo modelio. Rolės yra išskirstytos lygiais, pvz. naudotojas kuri priklauso pirmo lygio rolei gali atlikti skaitymo, rašymo, tvarkymo ir trynimo veiksmus.
- RBAC-SC: Role-based Access Control using Smart Contract[6]
  Rolėmis grįstas prieigos valdymo metodas naudojant išmanų kontraktą, veikia laba panašiai kaip Rolėmis grįstas prieigos valdymo metodas, tik yra vienas skirtumas, minėtas metodas naudoja blockchain technologiją saugiai gauti rolės prieigą ir atlikti tik naudotojui skirtas funkcijas.
- Intent-Based Access Control (IBAC)[7]
  Šiek tiek istorijos apie prieigos valdymo metodus. Pradedant naudotis kompiuteriais, žmonės suprato, kad reikėjo neleisti naudotojams neleisti vienas kitam kištis į darbus, kai naudotojai naudojasi vienu kompiuteriu. Problemai spręsti buvo sukurtas IBAC modelis, kuris priklauso nuo naudotojo tapatybės. Leidimas naudoti sistemos resursus pzv. Failus, buvo indeksuojami pagal vartotojo tapatybę, tai reiškia, kad failą gali redaguoti vienas arba keli žmonės priklausant nuo leidimo.

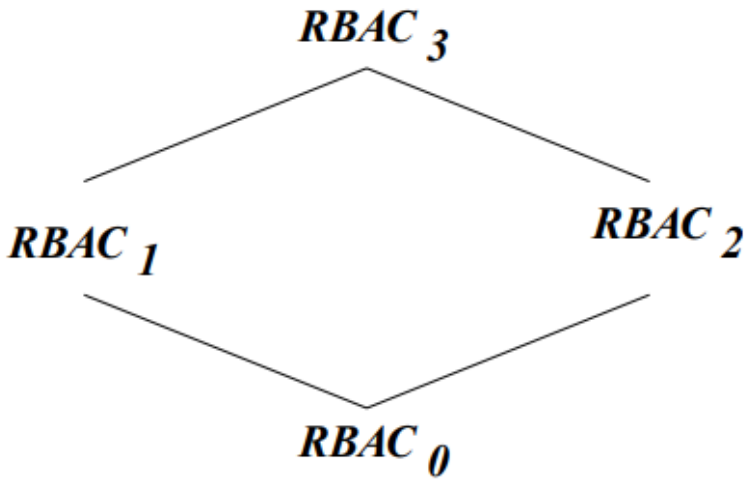
Rolėmis grįstas prieigos valdymo metodas

Taigi kas yra ta rolėmis grįstas prieigos valdymo metodas. Pagrindinė RBAC koncepcija yra rolės, kurios valdo naudotojo prieigą prie sistemos. Rolės yra tiltas tarp naudotojo ir sistemos licencijų. Naudotojas gavęs rolę gali pasiekti administratoriaus skirtą funkcionalumą. Administratorius valdo naudotojus, roles, licencijas. Dažniausiai administratoriui yra skirta valdymo rolė, su kuria gali valdyti naudotojus, roles ryšiu ir t.t. Niekas kitas be administratoriaus negali prieiti prie rolių valdymo.

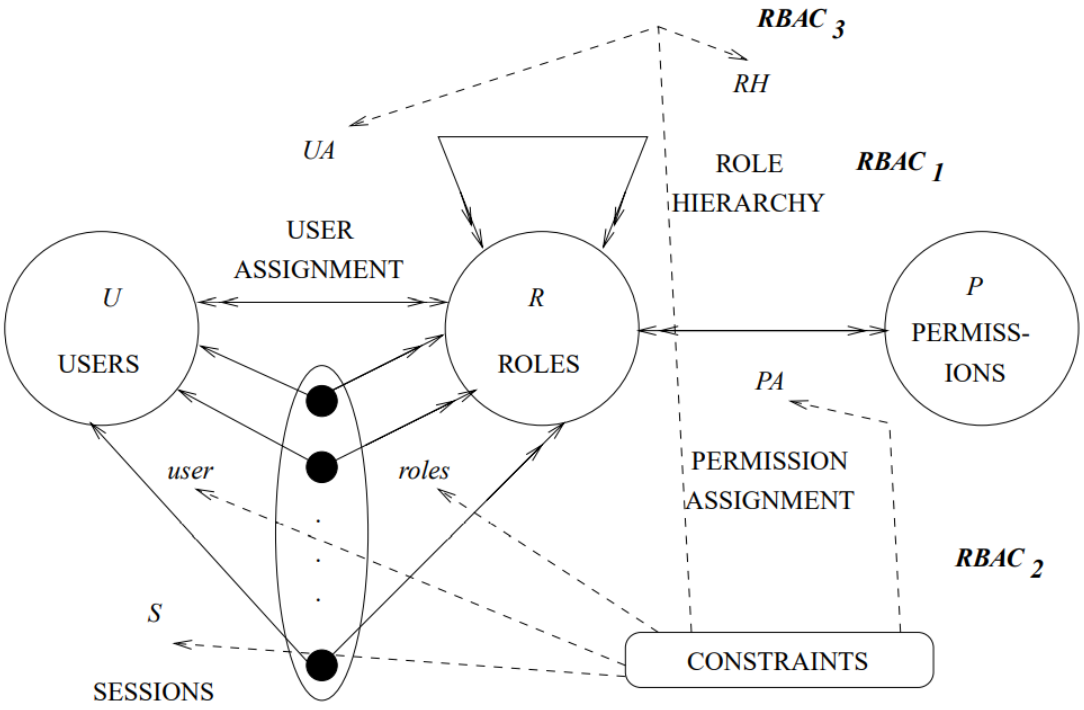


pav. 2 Rolių valdymo modelis

Rolėmis grįstas prieigos metodas atsirado 1990-ais, kaip patikima technologija valdant dideles sistemas. Rolėmis grįsto prieigos metodo paprastas paaiškinimas yra naudotojams yra priskirtos rolės kurios yra susietos su leidimais. Toks metodas palengvina sistemos valdymą. Galime pagalvoti, kas būtu jei nebūtu rolėmis pagrįsto prieigos valdymo metodo. Greičiausia arba naudotume senesnį modelį arba dar nematytą modelį.[7]



(a) Relationship among RBAC models



(b) RBAC models

Fig. 1. A family of RBAC models.

Be abejonių, ši technologija yra dabar viena iš populiariausių technologijų, valdant prieigą internete ir įvairiuose įrenginiuose.

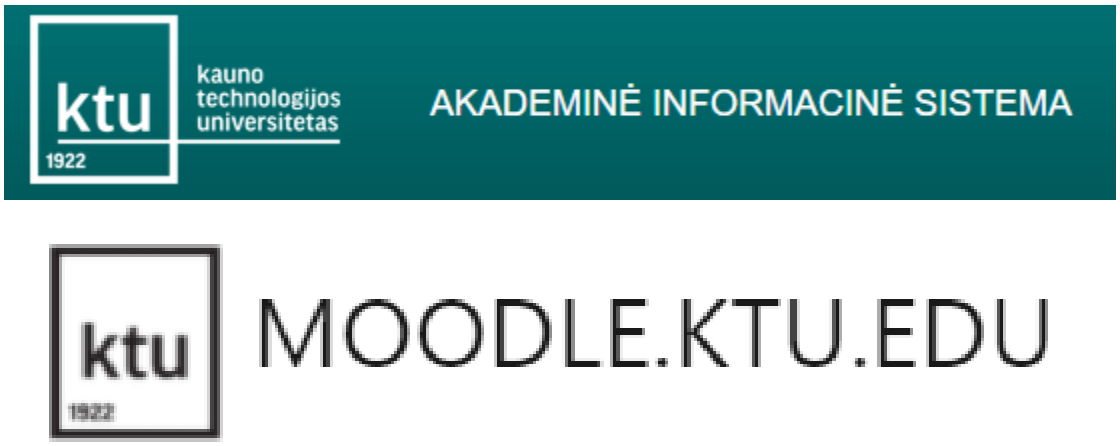
Naudodami RBAC, sistemos administratoriai gali kurti roles, suteikti toms rolėms leidimus ir priskirti vartotojų roles pagal jų konkrečias darbo pareigas ir politiką. Visų pirma, vaidmenų ir teisių ryšiai gali būti nustatyti iš anksto, todėl vartotojus lengva priskirti iš anksto nustatytoms rolėms. Be RBAC būtu sunku nustatyti, kokie leidimai turi būti suteikti naudotojams.

RBAC Administravimas

Pradėkime nuo to kas valdo roles. Tai roles valdo administratorius. Administratorius yra asmuo kuris sutvarko sistemos saugos politiką. Administratorius atlieka sistemos auditą, tvarko sistemos roles, tvarko sistemos leidimus kurie yra susieti su rolėmis, kitaip sakant priskiria leidimus prie rolių. Administratorius yra pagrindinis asmuo kuris prižiūri sistemos saugumą. [8]

Į administravimo funkcijas įeina leidimų kūrimas ir palaikymas elementams. Su administravimo funkcijomis galima sukurti naudotojus, roles, operacijas ir objektus. Turint prieigą prie tokių funkcijų administratorius gali valdyti visos sistemos prieiga naudotojams i naudotojų kiekį. Naudojant administravimo funkcijas galime nustatyti ryšius tarp naudotojų, rolių ir objektų. Pasinaudojus funkcijomis taip pat galima ir panaikinti sukurtus ryšius. Tokios funkcijos suteikia administratoriui valdyti visą sistema efektyviai ir dinamiškai. Žinoma augant sistemai auga ir darbo kiekis, kadangi reikia valdyti didesnį kiekį rolių ir jų ryšių.

Rolėmis grįsto prieigos valdymo metodo pritaikymas žiniatinklyje



pav. 3 KTU Sistema.

Galime pradėti nuo dažniausiai studentų naudoajamų sistemų, kaip moodle ir KTU akademinės sistemų. Minėtos sistemos turi vieną naudojamą prisijungimą. Kiekviena sistema valdo prieigą skitą naudotojui, pagal reikiamą funkcionalumą. Galime palyginti studento ir destytojo prieigą. Studentas gali peržiūrėti, ką destytojas yra ikėlęs ir naudotis pateikta medžiaga. Destytojas gali tvarkyti ikeltą medžiagą, trinti, keisti ir pridėti, destytojas turi turėti skirtingus leidimus nei studentas, kad galėtų tvarkyti reikiamus duomenis. Naudonat KTU sistemos prisijungimus galima prisijungti ir prie kitų sistemų, kaip Microsoft sistemų, moodle sistemos ir kitų.

pav. 4 Microsoft

Microsoft kompanija naudoja vieną iš prieigos valdymo metodų. Mircrosoft teikia ne vieną paslaugą kuri reikalauja prisijungti prie sistemos, kaip Outlook, Azure, Windows, One Drive, Office. Su kompanijos paskyra galima prisijungti prie visų minėtų sistemų. Kiekviena sistema suteikia skirtingą prieigą prisijungus prie sistemos. Vienas prisijungimas leidžia lengvai pasiekti visas teikiamas paslaugas. Visoms šioms paslaugoms reikia valdyti suteikiamus leidimus. Kiekviena sistema suteikia skirtingus leidimus. Su Microsoft paskyra taip pat galima prisijungti ir prie kitų sistemų, kurios palaiko paskyrą. Prisijungimas suteikia paskyrios duomenis kurie yra naudojami kitose sistemose.

pav. 5 Google

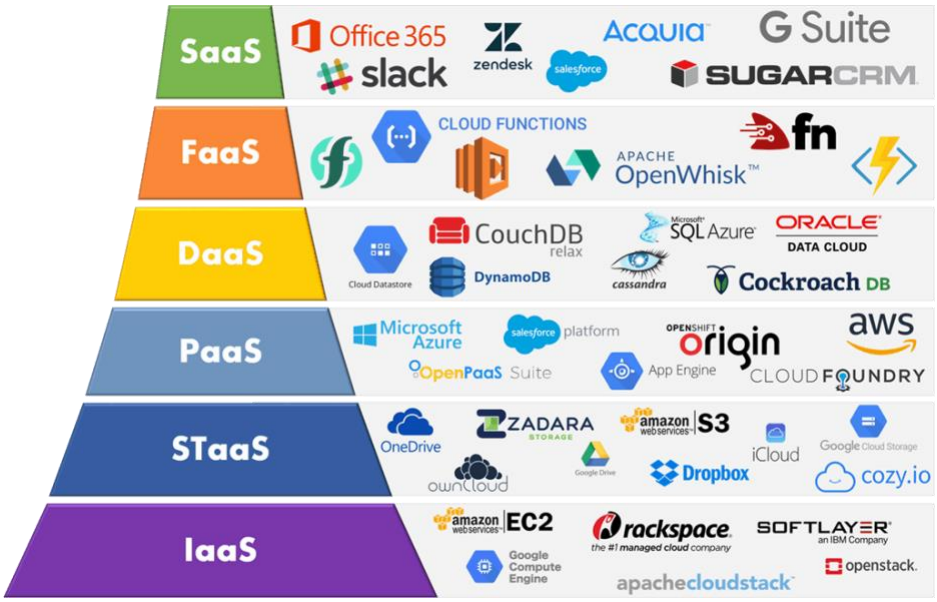
Didžioji dalis žmonių naudojasi Google paslaugomis. Kompanija turi ne vieną paslaugą kaip, gmail, drive, docs ir daug kitų paslaugų. Gauti prieigą prie minėtų paslaugų yra ganėtinai lengva. Užtenka turėti Google paskyrą ir yra gaunama prieiga prie įvairių paslaugų. Žinoma paslaugos turi skirtingus leidimus. Galime išanalizuoti Google Drive paslaugą. Naudotojai kurie naudojasi minėta paslauga nemokamai turi mažiau vietos, negu naudotojai, kurie moka mėnesinį mokestį. Čia yra tik vienas iš pavyzdžių kur yra naudojamas prieigos valdymas. Žinoma yra ir ne Google sistemų kurios naudotojo prisijungimui naudoja Google paskyras, tokių sistemų prieigos valdymas yra valdomas pačiose sistemose.

pav. 6 Facebook

Klausimas kyla kodėl butu galima panaudoti socialiniuose tinkluose. Užtenka truputį pagalvoti ir viskas tampa aišku. Socialinėse medijos paaiškintas prieigos valdymo metodas yra naudojamas naudotojų ir prieigos valdymui. Socialinės medijos turi didelį funkcionalumo kiekį, didelį vidinių grupių bei vidinių puslapių kiekį. Kiekvieną puslapį ar grupę kažkas valdo. Būtent tam valdymui yra naudojamas rolėmis pagrystą prieigos valdymo metodas, arba nors panašus metodas kuris remiasi šiuo metodu. Paaiškintas naudojimas yra tik vienas iš keleto. Kaip ir Microsoft ir Google, kompanija suteikia galimybę prisijungti prie kitų sistemų su viena paskyra. Jei kitos sistemos integruoja prisijungimą su Facebook paskyra, jos valdymą sukurtoje sistemoje vistiek reikia sukurti, kadangi tokios kompanijos suteikia integraciją paskyros prisijungimui.

Kitos debesų paslaugos

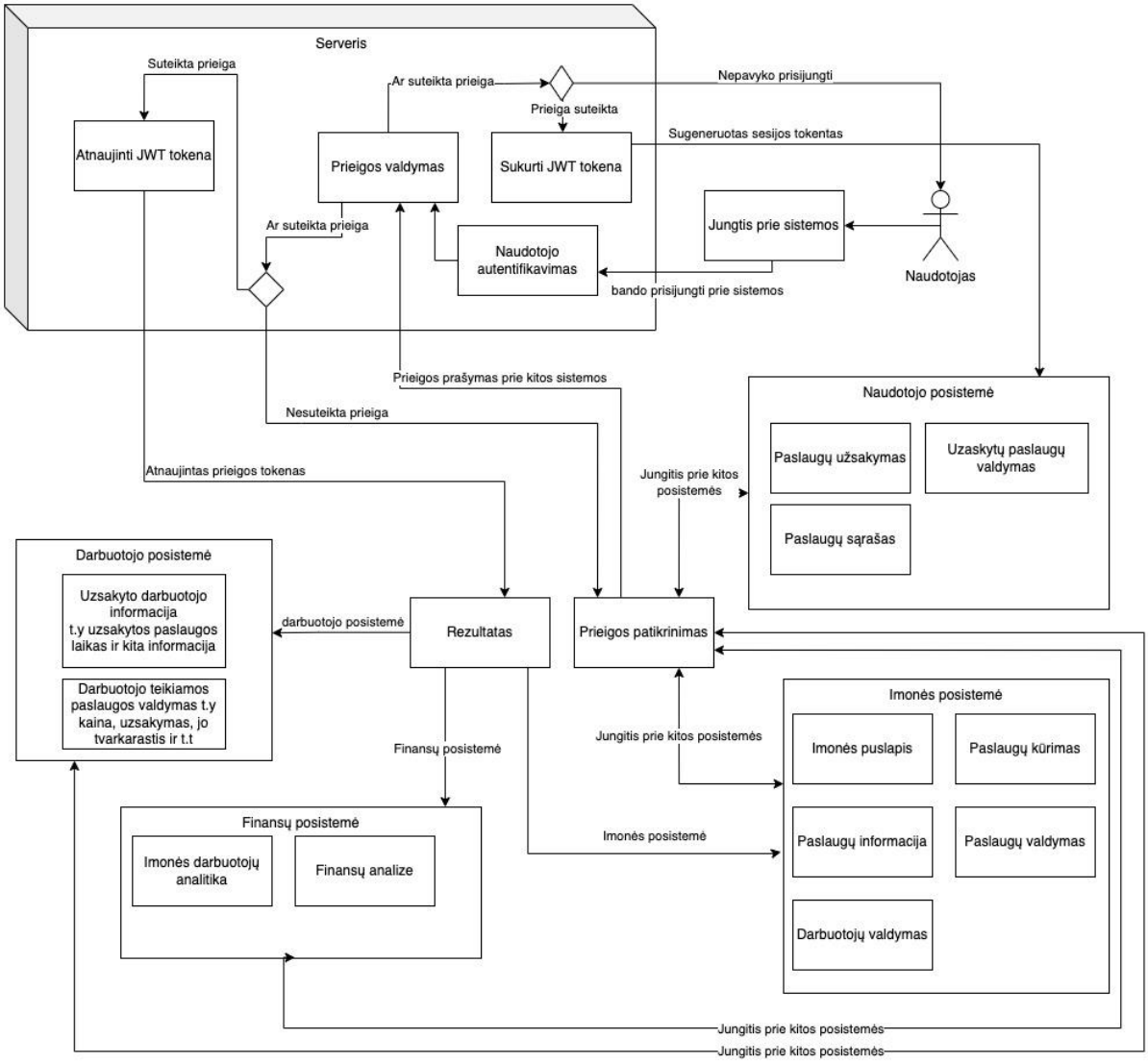
Vis dažniau ir dažniau išgirstame terminą debesų paslaugos. Taigi pirma išsiaiškinkime kas yra debesų paslaugos ir tada galėsime pardėti kalbėti apie rylelėmis grįsto prieigos metodo taikymą šituose paslaugose. Debesų paslaugos yra teikiamos įvairios paslaugos per internetą. [10] Tokioms paslaugoms reikia valdyti naudotojų prieigą prie aplikacijos ar paslaugos. Žinoma galima įvairiai valdyti naudotojų prieigą, bet šiuo momentu kalbame apie rolėmis grįsto prieigos metodą. Taigi kaip galima pritaikyti rolėmis grįstą prieigos metodą debesio paslaugomas. Na pirmiausia kaip ir prie visų paslaugų naudotojas turi prisijungti prie sistemos. Paslaugų teikimas gali būti nevienodas visiems, tam galima pritaikyti rolėmis grįstą prieigos metodą. Kiekviena rolė turi prieiga prie tam tikrų paslaugų. Žinoma administratoriui atitenka daug darbo reguliuoti roles, todėl yra naudojamas kitas metodas arba automatinis rolių valdymo metodas.



pav. 7 Debesų paslaugos

Projekto konceptas ir metodo konceptas

Rolėmis grįstas prieigos valdymo metodas yra naudojamas didelėse sistemose, prieigos valdymui. Prieigos valdymas padeda valdyti naudotojus ir jų prieigą prie sistemų. Mano sistema nėra labai didelė, bet koncepto įrodymui užtenka ir mažos sistemos. Mano sistema turės penkias posistemes.



Pav. 8 Prototipas.

Trumpas paaiškinimas apie sistemą. Sistema yra skirta parduoti ir pirkti paslaugas. Sistemoje yra įmonės, darbuotojai, naudotojai ir finansininkai, kurie turi savo mažas posistemes sistemai valdyti. Sistema nėra didelė, bet rolėmis grįsto prieigos metodo panaudojimui užteks, kadangi reikia tikrinti skirtingų posistemių prieigą naudotojui. Sistemoje gali būti sukurtas įmonės puslapis, kuris gali turėti daug specialistų, kaip santechnikų, statybininkų ir kitų darbuotojų. Kiekvienas darbuotojas yra valdomas įmonės savininko. Darbuotojams gali būti nustatytas

valandinis darbo mokestis, už kurį jie turės dirbti. Finansininkai valdys atskiras įmonės finansus, finansininkai gali valdyti daugiau negu vieną įmonę. Kaip ir įmonės savininkas gali turėti daugiau negu vieną įmonę. Taip pat yra paprasti sistemos naudotojai. Paprasti naudotojai gali užsisakyti paslaugas iš įmonės arba pasirinkti specifinį darbuotoją, kuris gali atlikti darbą. Įmonės darbuotojai gali valdyti savo laiką ir informaciją, į informaciją gali įeiti atlikti darbai, sertifikatai ir kita informacija.

Sistemos dalys ir jų aprašas:

- Darbuotojo Posistemė – sistema skirta valdyti naudotojų duomenis, algą ir kitą informaciją
- Finansų posistemė – buhalterijos dokumentai
- Naudotojo posistemė – produkto bei darbuotojų atliktų darbų analizė
- Įmonės posistemė– produkto valdymo sisteminė dalis
- Autentifikacijos posistemė

Duomenys, kurie galimai bus naudojami sistemoje

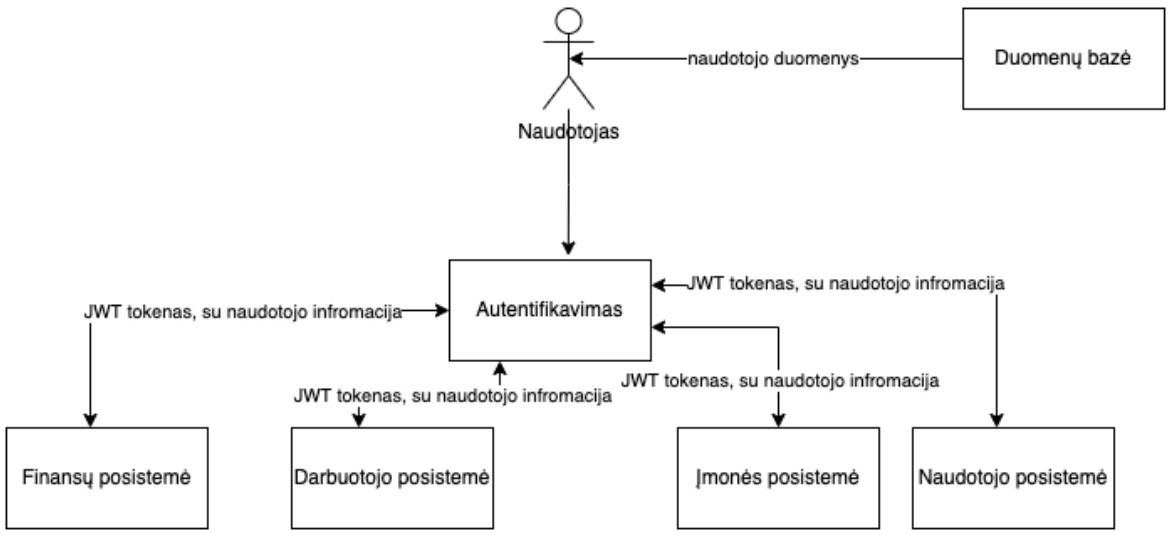
- Darbuotojų duomenys
    - Vardas
    - Pavardė
    - Adresas
    - Pozicija
    - El. paštas
    - Slaptažodis
- Atributų duomenys
    - Analizės prieigai reikalingas atributas
    - Darbų valdymas reikalingas atributas
    - Prieiga prie duomenų valdymo reikalingas atributas
- Darbų duomenys
    - Darbo aprašas
    - Darbo pavadinimas
    - Priskirtas naudotojui darbui
- Rolių duomenys
    - Sudaryta rolių matrica, prie kurių sistemų naudotojas turi prieigą
    - Naudotojas
    - Posistemės pavadinimas
    - Rolė
- Dokumentų informacija
    - Dokumentai

JSON puslapio žetonai (JWT)

JWT tai yra atviras standartas, skitas puslapių autentifikacijai, bei duomenų pernešimui iš vieno puslapio į kitą. Duomenys kurie turi būti perduoti yra saugojami JSON failo tipu. Toks failo tipas yra labai plačiai naudojamas. Dažniausiai JSON duomenys yra persiunčiami tarp puslapių, arba jie yra siunčiami į serverį. Kadangi šito failo tipas yra labai lengvai koreguojamas, tiek iš kodo pusės tiek tvarkant kodą ranka. Į JWT įeina daugiau nei JSON, kadangi tai yra saugus standartas, saugiems duomenims persiųsti. Norint perduoti duomenis saugiai jie turi būti kažkokiu būdu užslaptinti, kad eitu užslaptinti ir atslaptinti naudojant tą patį saugų metodą. JWT naudoja vieną iš pasirašymo algoritmų ar tai būtu rsa, hs256 ar bet koks kitas algoritmas. Dažniausiai JWT susideda iš trijų elementų „Header“, „Payload“ ir „Signature“. Į pirmąjį elementą įsideda algoritmas ir žetono tipas, šiuo atveju „JWT“. Į antrąjį elementą yra įdedami duomenis kuriuos norime perduoti ir į trečiąjį elementą įsideda algoritmo parašas. Kiekvienas elementas yra atskirai šifruojamas. Trečias elementas yra pasirašomas su paslapties žodžiu ar raidžių kratiniu. Dažniausiai į JWT yra saugojama tik reikalinga informacija naudotojo verifikacijai.

Duomenys kurie bus saugojami JWT:

- Naudotojo duomenys
    - Vardas
    - Pavarde
- Rolė kurią gauna prisijungus
- Įmonės identifikacijos kodas, jei naudotojas dirba įmonėje
- Darbuotojo identifikacijos kodas, jei naudotojas dirba.



Pav. 9 Duomenų srauto diagrama

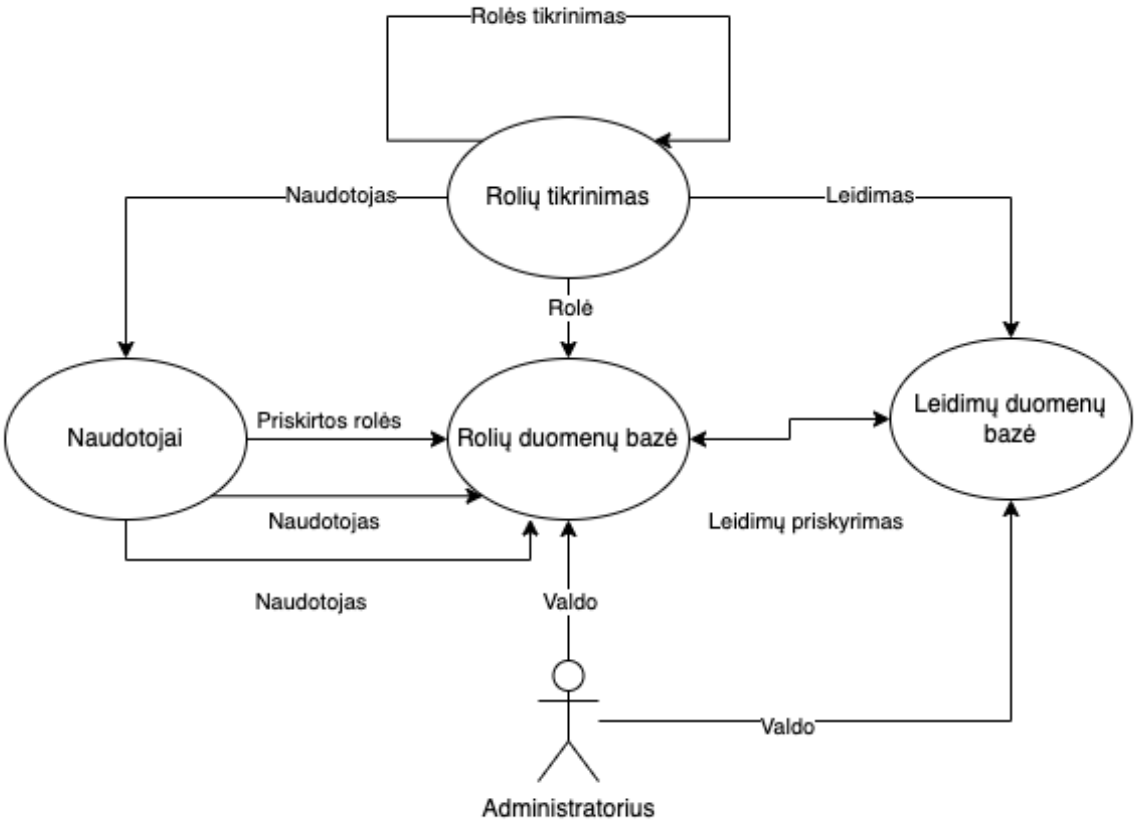
Sistemoje bus duomenys, kurie pastoviai bus persiunčiami, kadangi to reikalauja rolėmis grįstas prieigos metodas. Naudotojui norint pasiekti kitą posistemę reikia turėti roles, pagal kurias sistema gali patikrinti, ar naudotojas gali gauti prieigą prie posistemės. Duomenys turi būti persiunčiami, kadangi gauti visus duomenis iš duomenų bazės ar failo nėra efektyvu. Dažniausiai būna persiunčiami tokie duomenys, kaip darbuotojo duomenys, rolių ir naudotojo duomenys.

Prieigos valdymo metodai

Problemos prieigos valdymo metoduose yra ne viena problema, bet pati pagrindinė yra saugumas ir jo valdymas. Saugumas yra viena iš didžiausių problemų apskritai. Nesaugios sistemos yra greitai nulaužiamos. Dabar pradėkime apie Rolėmis grįsto prieigos valdymo metodo ir atributais grįsto prieigos valdymo metodo problemas, kurios iškyla naudojantis šiais metodais. Naudojantis rolėmis grįstu prieigos valdymo metodu yra sunku, kadangi rolių kiekis yra didelis, kaip ir posistemių kiekis dažniausiai būna didelis. Dažniausiai roles valdo administratorius, kuris valdant tokias sistemas gali įvelti daug klaidų, kurios gali likti nepastebėtos. Rolėmis grįsto prieigos valdymo metodo pagrindinė problema yra dinamiškumas. Kadangi rolės yra kuriamos ir priskiriamos reikia bent vieno darbuotojo, kuris jas prižiūrėtu. Yra sukurtos sistemos, kurios tai padeda išspręsti, bet jos būna labai didelės. Paėmus kitą metodą, kuris yra atributais grįstas prieigos valdymo metodas, jis taip pat nėra tobulas. Šitas metodas yra dinamiškas, bet tai reiškia, kad yra sunkus valdymas, kiekvienas posistemė turi turėti savo taisykles, kurių pagalba bus nustatyta ar naudotojo esami atributai atitinka taisykles. Naudojant paremtą prieigos valdymo metodą iškyla problemų bandant prieiti prie posistemės prieigos per visiškai kitą sistemą, kadangi šitas metodas yra statinis, todėl negalima nuspręsti, kokias roles galima skirti naudotojui iš kitos sistemos.

Rolėmis grįstas prieigos valdymo metodas

Rolėmis grįstas metodas suteikia prieigą tik skirtiems asmenims. Rolės turi asociacijas su funkcijomis ar mini sistemomis. Kiekviena rolė suteikia skirtingą prieigą prie skirtingos sistemos tai leidžia atskirti naudotojų prieigą ir juos pačius, kai naudotojas naudojasi sistema. Šitas standartas yra naudojamas jau seniai. Pavyzdžiui Microsoft naudoją šitą valdymo sistemą Azure paslaugoms. Kadangi šitas metodas vis dar naudojamas yra įvairiose kompanijose jis yra patikimas, bet su savo spragomis.

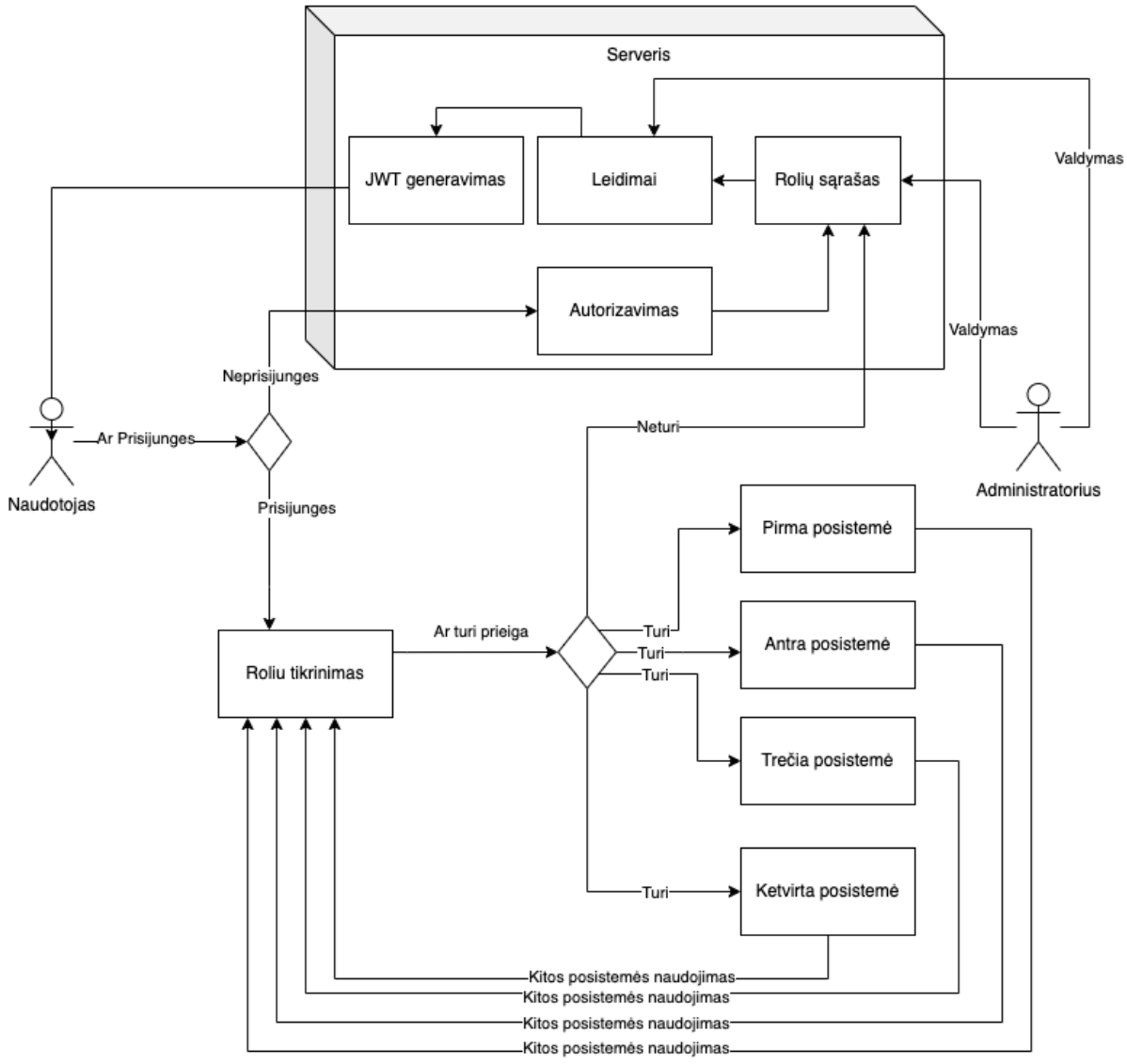


Pav. 10 Rolėmis paremto valdymo metodo modelis.

Pradėkime nuo to, kad reikia turėti registruotų naudotojų sąrašą, su naudotojų informaciją. Tada yra sukuriamas rolių asociacijų sąrašas su paslaugomis, turint tokį sąrašą galima pradėti valdyti prieigą prie paslaugų. Rolės turi būti susietos su teikiamomis paslaugomis, kitu atveju arba niekas neturės prieigos prie paslaugų, arba visi turės prieigą prie visų paslaugų. Turi būti ir paslaugų sąrašas, kitaip nebus galima priskirti rolių prie tam tikrų paslaugų. Turint tris sąrašus naudotojų, paslaugų ir rolių galima pagalvoti kur visi duomenys bus saugomi. Duomenų saugojimui pasirinkau MySQL duombazę. Tai reiškia, kad visi duomenys bus saugomi sistemos duombazėje. Toks pasirinkimas leis greitai tvarkyti reikalingus duomenis, bei leidžia greitą prieigą tikrinant roles ir sistemas. Sudarius sąrašus galima apjungti visą sistemą. Sudarius visus sąrašus viskas turi būti apjungta. Rolės su paslaugomis turi būti surištos arba atvirkščiai. Surišus roles galima jas priskirti prie naudotojų. Kiekvienas naudotojas turi turėti priskirtą rolę, kad gauti prieigą prie reikalingos darbo posistemės. Galima pradėti spręsti, kaip bus tikrinamos rolės ir kaip veiks prieiga prie posistemių. Pirmiausia bus patikrinama posistemė prie kurios bandoma prieiti. Gavus posistemės duomenis galima gauti rolių matricą, naudotojui bandant jungtis prie posistemės yra gaunama jo rolė. Gavus naudotojo rolę galima pradėti tikrinti rolę gautoje rolių matricoje. Patikrinus ar su naudotojo role galima prieiti prie posistemės naudotojas gauna arba negauna prieigą prie posistemės.

Žiniatinklių programoms skirtas rolėmis grįstas prieigos valdymo metodas

Šitas metodas yra patobulintas paprastas rolėmis grįstas prieigos metodas, paprastas rolėmis grįstas prieigos metodas yra skirtas paprastoms aplikacijoms ar sistemoms. Rolėmis grįstas prieigos metodas skirtas puslapiams skiriasi nuo paprasto rolėmis grįsto prieigos metodo, kadangi skiriasi prieiga prie puslapių. Rolių valdymas dažniausiai būna pusiau automatinis, kadangi registracijos metu naudotojai registruojasi pasirenkant specifinę posistemę. Naudotojas gali susikurti ir paprastą paskyrą, bet jis gali būti pridėtas prie posistemės, vadovo ar kito žmogaus kuris dalinai valdo posistemę. Sukurtas paprastas naudotojas iš karto būna priskiriama paprasta rolė.



Pav. 11 rolėmis paremto metodo puslapių aplikacijos modelis

Trumpam sistemos paaiškinimą matome nuotraukoje. Pirmiausia Naudotojas turi roles ir su kuriais yra susietas arba dinamiškai arba statiškai. Naudotojui norint prisijungti prie sistemos turi praeiti verifikaciją. Pirmiausia yra gaunama naudotojo rolė, kadangi kiekvienas naudotojas turi savo asmenine informaciją, kurią gali valdyti, tai yra vadinamoji naudotojo posistemė. Prisijungus prie sistemos ir gavus naudotojo rolę yra generuojamas „JSON Web Token“ (JWT). Šiame žetone yra saugoma informacija, kaip naudotojo rolė, naudotojo vardas pavardė ir kompanijos duomenys, jei naudotojas yra susietas su kompanija nepriklausant nuo kompanijos rolės. Žetonai dažniausiai saugo tik reikiamą mažą duomenų kiekį, kad nesulėtinti puslapio, krovimo metu. Norint naudotojui gauti prieigą prie kitos posistemės, naudotojas bando pasiekti posistemės puslapį prie kurio turi turėti prieigą. Naudotojui bandant pasiekti kitą posistemę, yra siunčiama užklausą į API, kuris valdo prieigą, gavus atsakymą iš API naudotojui yra arba pakeičiamos rolės ir atnaujinamas „JWT“ , kuris skirtas tikrinti prieigą tarp posistemės puslapių. Naudotojui negavus prieigos prie kitos posistemės naudotojas yra nukreipiamas į buvusį puslapį iš kurio bandė prieiti prie kitos posistemės. Prieigos tikrinimo metu „JWT“ yra generuojamas iš naujo visas arba tik dalis, kuri saugo duomenis apie naudotoją. „JWT“ yra užšifruotas pagal pasirinktą algoritmą ir yra slaptažodis užšifruojamas (parašas), kuris yra žinomas tik esamam puslapiui. Pagal užšifruotą parašą, sistemos patikrina ar „JWT“ yra galiojantis. Patikrinus ar žetonas yra galiojantis ir ar naudotojas turi prieigą prie posistemės, naudotojas yra įleidžiamas į sistemą. Naudotojas gali turėti iškarto priskirtas teises, kurios skirtos tik vienai posistemei. Tokiu atveju naudotojas gauna prieigą tik prie vienos posistemės, bet bandant prieiti prie kitų posistemių, sistema tikrina ar jis turi reikiamą prieigą prie jos. Neturint reikiamos prieigos naudotojas negali prieiti prie kitos posistemės. Sistemos administratorius yra skirtas prižiūrėti, kad nekiltų nesklandumų. Šiuo atveju, jis tikriausiai bus vadovas, arba vadovo priskirtas asmuo. Rolių administratorius, administruos finansų posistemę, darbuotojų posistemę ir vadovų posistemę. Jam naudotojų posistemės nereikia valdyti, kadangi kiekvienas prisiregistravęs naudotojas yra paprastas sistemos naudotojas.

Hierarchinis rolių valdymas

Buvo pasirinkta naudoti hierarchinį rolių valdymą, kadangi šioje sistemoje jis atitinka geriau, negu rolių matricą. Hierarchinis rolių valdymas yra skirtas tada, kada yra naudotojų hierarchija. Šio metu sistemos vizijoje yra numatoma, kad bus keletas naudotojų tipų. Aukščiausias naudotojo tipas yra Vadovas, kuris valdo įmonę, kiti toliau naudotojai yra finansininkai ir paprasti darbuotojai ir tada eina rolių administratoriai, kurie yra priskirti vadovo valdyti įmonės darbuotojų teises ir paprasti sistemos naudotojai.

Naudotojai	Rolių valdymo posistemė	Naudotojo posistemė	Finansų posistemė	Darbuotojų posistemė	Įmonės posistemė
Jonas	V	V			
Kęstutis		V			
Petras		V	V		
Mantas	V	V	V	V	V

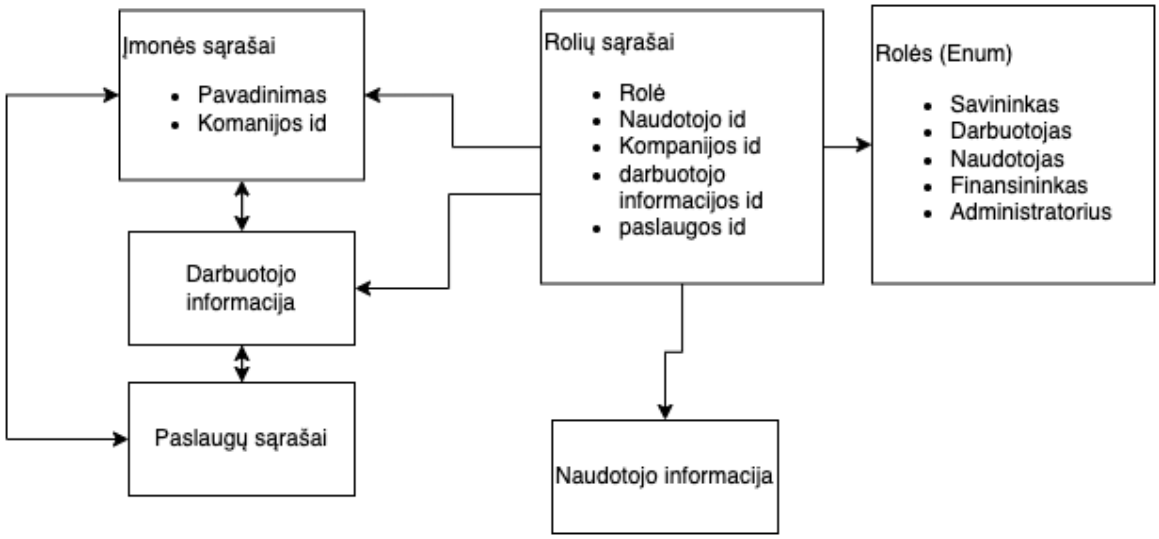
Benas		V		V	

Lentelė 1 Rolių pavyzdys

Pavyzdinėje naudotojų prieigos valdymo matricoje, matome, kad tinka naudoti hierarchinę rolių valdymą. Šitas metodas tinka, nes yra naudotojo posistemė, kuri siejasi su visomis kitomis rolėmis. Prie finansų posistemė yra susieta irgi su paprasto naudotojo posisteme. Finansininkas nebūtinai turi prižiūrėti finansus, bet gali užsisakyti ir asmenines paslaugas su vienu prisijungimu. Įmonės posistemė yra surišta su visomis kitomis posistemės. Įmonės vadovui, gali reikėti prieiti prie finansų posistemės, darbuotojų posistemės ir kitų posistemių, todėl ja reikia suteikti nemaža prieigą prie posistemių. Kad veiktų hierarchinė sistema, turi būti nustatytos taisyklės, kaip turi veikti hierarchija.

Duomenų struktūra

Naudojant rolėmis pagrįstą prieigos metodą, kuris yra skirtas puslapių aplikacijoms reikia sudaryti, bent paprastą duomenų architektūrą. Kadangi bus naudojama hierarchinės rolės, duomenų bazę bus šiek tiek lengviau valdyti, kadangi nelieka naudotojų dublikatu.



Pav. 12 Duomenų bazės pavyzdinė struktūra

Buvo nubraižyta paprasta schema, kuri padeda orientuotis, kaip turėtų atrodyti duomenų bazė. Taigi duomenų bazėje yra Įmonės sąrašai, rolių sąrašai, rolės ir kitos lentelės. Pagrindinės lentelės yra rolių sąrašai ir pačios rolės. Rolių sąrašai yra susieti su kitomis lentelėmis. Rolių lentelėje yra pagrindinės rolės, kurias naudos būsima sistema. Rolių sąrašuose yra naudotojai kurie yra susieti su kitomis lentelėmis. Rolių tikrinimo metus, naudotojas yra surandamas rolių sąraše ir patikrinamas ar turi tam skitą prieigą prie posistemės.

Išvados

- Išanalizavus žiniatinklio programų prieigos valdymo problemas pastebėta, kad kuriant prieigos valdymo modelį yra ne viena problema su kuria susiduriama. Visos problemos gali būti suskirstytos į tris pagrindines problemas. Pirmiausia tai yra žinomiausia problema duomenų apsaugos problema. Kita viena iš pagrindinių problemų yra naudotojo konfidencialumas, apie kurį reikia pagalvoti. Paskutinė didelė problema yra sistemos prieiga, reikia pagalvoti apie sistemos apkrovą naudojant prieigos valdymo metodą. Žinoma yra ir kitų problemų su kuriomis galima susidurti.
- Išanalizavus žiniatinklio programų prieigos valdymo metodus buvo pastebėta, kad jų yra daug ir skiriasi drastiškai nuo analizuojamo metodo, arba yra metodai yra paremti rolėmis grįsto valdymo metodu.
- Atlikus rolėmis grįsto prieigos valdymo metodo analizę buvo suprasta, kad šitas metodas nėra toks lengvas kaip iš pradžių buvo galvota. Buvo pamatytą, kad jį panaudoti reikia daug žingsnių kuriuos reikia atlikti. Pastebėta kaip veikia naudotojų valdymas, rolių valdymas ir licencijų tvarkymas. Išsiaiškinta, kaip turėtų veikti prieigos valdymas iš administratoriaus pusės.
- Buvo išanalizuota kur žiniatinklyje yra pritaikytas rolėmis grįstas metodas. Buvo pastebėta, kad yra metodų kurie remiasi rolių prieigos valdymo metodus svetainėse, kurios naudoja programinę įrangą susijusią tarp skirtingų puslapių. Taip pat buvo išsiaiškinta kad rolėmis grįstas prieigos valdymo metodas yra naudojamas debesų paslaugoms valdyti.
- Buvo sukurta projekto vizija. Buvo sugalvota pavyzdinio projekto idėja. Sugalvota idėja buvo išvystyta iki projekto aprašymo ir schemų. Išsiaiškinus, kaip daugiau mažiau turi veikti projektas, buvo išsiaiškinta, kokie duomenys turi būti naudojami autentifikavimui ir prieigos suteikimui prie posistemių. Buvo paanalizuotas teorinis rolėmis grįsto prieigos metodas. Paanalizavus teorinį modelį buvo pateiktas Rolėmis grįstas žiniatinklio programų prieigos valdymo metodas.

Literatūros sąrašas

[1] D. R. K. R. C. David F.Frraiolo, *Role-Based Access Controll*. 2003.

[2] N. Mundbrod and M. Reichert, “Object-Specific Role-Based Access Control,” *Int. J. Coop. Inf. Syst.*, vol. 28, no. 1, pp. 1–30, 2019, doi: 10.1142/S0218843019500035.

[3] S. Bhatt, T. K. Pham, M. Gupta, J. Benson, J. Park, and R. Sandhu, “Attribute-Based Access Control for AWS Internet of Things and Secure Industries of the Future,” *IEEE Access*, vol. 9, pp. 107200–107223, 2021, doi: 10.1109/ACCESS.2021.3101218.

[4] M. Belchior, D. Schwabe, and F. Silva Parreiras, “Role-based access control for model-driven web applications,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7387 LNCS, pp. 106–120, 2012, doi: 10.1007/978-3-642-31753-8_8.

[5] M. U. Aftab *et al.*, “Permission-based separation of duty in dynamic role-based access control model,” *Symmetry (Basel).*, vol. 11, no. 5, 2019, doi: 10.3390/sym11050669.

[6] J. P. Cruz, Y. Kaji, and N. Yanai, “RBAC-SC: Role-based access control using smart contract,” *IEEE Access*, vol. 6, no. c, pp. 12240–12251, 2018, doi: 10.1109/ACCESS.2018.2812844.

[7] A. Karp, H. Haury, and M. Davis, “From ABAC to ZBAC: The evolution of access control models,” *5th Eur. Conf. Inf. Manag. Eval. ECIME 2011*, vol. 9, no. 2, pp. 202–211, 2011.

[8] L. Dongdong, X. Shiliang, Z. Yan, T. Fuxiao, N. Lei, and Z. Jia, “Role-based access control in educational administration system,” *MATEC Web Conf.*, vol. 139, pp. 1–8, 2017, doi: 10.1051/matecconf/201713900120.

[9] J. Li, Y. Tang, C. Mao, H. Lai, and J. Zhu, “Role based access control for social network sites,” *2009 Jt. Conf. Pervasive Comput. JCPC 2009*, pp. 389–393, 2009, doi: 10.1109/JCPC.2009.5420153.

[10] W. T. Tsai and Q. Shao, “Role-based access-control using reference ontology in clouds,” *Proc. - 2011 10th Int. Symp. Auton. Decentralized Syst. ISADS 2011*, vol. 2, pp. 121–128, 2011, doi: 10.1109/ISADS.2011.21.