Lauri Pihlajamäki

Active Directoryn käyttäjien hallinta ASP.NET-sovelluksella

Opinnäytetyö
Syksy 2016
SeAMK Tekniikka
Tietotekniikan tutkinto-ohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Lauri Pihlajamäki

Työn nimi: Active Directoryn käyttäjien hallinta ASP.NET-sovelluksella

Ohjaaja: Markku Lahti

Vuosi: 2016 Sivumäärä: 49 Liitteiden lukumäärä: 3

Opinnäytetyön aiheena oli suunnitella ja luoda ASP.NET-sovellus, jolla pystyttäisiin uusimaan SeAMKin opiskelijoiden salasanoja. Päätavoitteena oli luoda ASP.NET-sovellus joka ottaa yhteyden Active Directory– ja ADAM-hakemistopalveluihin. Sovellus hakee listan opiskelijoista ja antaa mahdollisuuden vaihtaa tietyn opiskelijan salasana. Työ tehtiin SeAMKin Helpdesk-palvelu Jelpparille. Työn varmuuskopioinnissa käytettiin versionhallintaa.

Työn alussa käydään läpi hakemistopalveluiden historiaa ja keskitytään kertomaan Microsoftin Active Directory- ja ADAM-palveluista. Lisäksi kerrotaan myös ASP.NET-sovelluskehyksen historiasta ja sen toimintaperiaatteista. Opinnäytetyössä kerrotaan myös versionhallinnasta ja sen vaihtoehdoista, kuten Git ja TortoiseSVN. Samoin kerrotaan myös web-sovellusten turvaamisesta autentikointi- ja autorisointi-palveluiden avulla. Lopuksi käydään läpi sovelluksen toteutus.

ASP.NET on kehittyvä ja lisättävien kirjastojen avulla muokattavissa oleva alusta verkkosivujen tekoa varten. Sovelluskehyksen avulla voidaan luoda monimutkaisia web-sovelluksia yksinkertaisilla käyttöliittymillä.

Avainsanat: Active Directory, ADAM, versionhallinta, autentikointi, autorisointi, ASP.NET.

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Author: Lauri Pihlajamäki

Title of thesis: Management of Active Directory users using an ASP.NET web site

Supervisor: Markku Lahti

Year: 2016 Number of pages: 49 Number of appendices: 3

The aim of this thesis was to design and implement an ASP.NET web site that could retrieve a list of users from Active Directory and ADAM. The web site would also enable the resetting of a selected user's password. Version control was used to store the process of development.

The first half of the thesis introduces the history of directory services and focuses on Microsoft's Active Directory and ADAM. The thesis also studies the history of ASP.NET-development. Next the term and some examples of version control systems are explained. The thesis also describes the authentication and authorization within web sites. The final part of the thesis focuses on the development of the ASP.NET web application.

The ASP.NET web framework is still in development and continues to evolve with community made libraries. The ASP.NET framework makes it possible to produce complex programs with a simplistic user interface.

Keywords: Active Directory, ADAM, version control, authentication, authorization, ASP.NET.

SISÄLTÖ

Kuvioluettelo

Käytetyt termit ja lyhenteet

AD	Active Directory. Hakemistopalvelu, joka varastoi tietoa verkon tietokoneista ja käyttäjistä.
ADAM	Active Directory Application Mode. Kevyt versio AD-palvelusta, joka voi suorittaa samoja toimintoja kuin AD, ilman AD:n asennuksen vaatimia ominaisuuksia.
ADsPath	AD-polut. Tapa viitata AD-palvelussa oleviin objekteihin LDAP-standardin mukaisesti.
AJAX	Asynchronous JavaScript And XML. Joukko web–sovelluskehityksen tekniikoita, joiden avulla sovelluksesta saadaan vuorovaikutteisempi.
ASP.NET	Active Server Pages.NET. Microsoftin Web–sovellusten luomiseen tarkoitettu ohjelmointimalli.
DN	Distinguished Name. AD:n nimeämiskäytäntö, jolla viitataan tiettyyn objektiin AD-palvelussa.
HTML	Hyper Text Markup Language. Internetsivujen luomisessa käytettävä koodikieli.
LDAP	Lightweight Directory Access Protocol. Hakemistopalveluille tarkoitettu verkkoprotokolla.
Schema	Objektia kuvaava kaava. Listaa objektin attribuutit ja tiedot.
Versionhallinta	Järjestelmä, jolla tallennetaan alkuperäinen ohjelma ja siihen tehdyt muutokset.

1 JOHDANTO

1.1 Työn tausta

Seinäjoen Ammattikorkeakoulun opiskelijat ja opettajat kirjautuvat koulun tietokoneisiin omilla tunnuksillaan. Tunnuksille on asetettu omat salasanat, jotka vanhenevat tietyn ajan kuluessa. Jos opiskelija ei vaihda salasanaansa sitä ennen, tunnukset menevät lukkoon ja salasana on uusittava ATK-tuen kautta.

Helpdesk-palvelu Jelppari on osa SeAMKin tietohallintoa. Jelppari hoitaa tietoteknisen avun lisäksi opiskelijoiden ja opettajien salasanojen uusimisen. Tähän prosessiin Jelppari kuluttaa huomattavan määrän aikaa, sillä opiskelijan ja tunnusten tiedot on todennettava kolmen erillisen ohjelman kautta ennen salasanan uusimista. Voisiko tätä prosessia millään nopeuttaa?

Tietohallinto on kehittänyt web-sovelluksen, jolla voitiin vaihtaa opiskelijoiden salasanoja, mutta siitä puuttui ominaisuuksia, joten Jelppari ei voi käyttää sitä.

1.2 Työn tavoite

Työn tavoitteena on kehittää uusi web-sovellus, jolla hallittaisiin Seinäjoen koulutuskuntayhtymän ja ammattikorkeakoulun opiskelijoiden salasanoja. Sovelluksen tulisi hakea opiskelija hakusanojen avulla ja näyttää tarkat tiedot kyseisestä henkilöstä, kuten syntymäaika. Sovelluksen avulla on pystyttävä uusimaan tämän opiskelijan salasana. Sovelluksen varmuuskopioinnissa tullaan käyttämään Tietohallinnon tarjoamaa versionhallintapalvelinta.

1.3 Työn rakenne

Luvussa 2 kerrotaan nimipalveluista ja hakemistopalveluista, sekä Active Directorysta ja sen rakenteesta. Luvussa kerrotaan myös ADAM-palvelusta. Luvussa 3 kerrotaan ASP.NET-ohjelmistokehyksestä ja sen toimintamalleista. Neljännessa luvussa käydään läpi versionhallinnasta yleistä tietoa, kerrotaan sekä

Git-versionhallinnasta että tarkemmin työssä käytetystä TortoiseSVN-versionhallintaohjelmasta.

Viidennessä luvussa keskitytään web-sivujen tietoturvaan ja sovelluksen käyttäjän tunnistamiseen sekä oikeuksien tarkasteluun. Luvussa 6 esitellään sivuston työstämisen kulkua. Lopuksi viimeisessä luvussa kerrotaan työn tuloksesta, yhteenveto ja pohdintaa sovelluksen työstämisestä.

1.4 Tietohallinnon Helpdesk-palvelu Jelppari

Seinäjoen koulutuskuntayhtymän ja Seinäjoen Ammattikorkeakoulun oppilaitosten ATK-tukena toimii tietohallinnon Helpdesk-palvelu Jelppari. Jelppari vastaa työntekijöiden ja opiskelijoiden salasanojen hallinnoinnista, sekä SeAMKin sisällä tapahtuvien tietotekniikkaongelmien korjaamisesta. Jelpparin henkilöstöön kuuluu neljä henkilöä, ja palvelun toimipiste sijaitsee Seinäjoella. (Jelppari [Viitattu 18.5.2016].)

2 HAKEMISTOPALVELUT

Suurten tietomäärien hallinnoiminen manuaalisesti on erittäin työlästä. Pienissä lähiverkoissa tämä on vielä mahdollista, mutta suurempien verkkojen kanssa työskentely on jo mahdotonta. Tietotekniikan kehittyessä esimerkiksi yhtiöt käyttävät erilaisia **nimipalveluita** verkon resurssien nimeämiseen ja tunnistamiseen. Esimerkkinä voidaan käyttää sähköpostijärjestelmää. Jotta viesti voidaan lähettää toiselle henkilölle, on kerrottava sähköpostijärjestelmälle tarkat tiedot vastaanottajasta eli vastaanottajan sähköpostiosoite. (Talvivaara [Viitattu 13.5.2016].)

Toisena esimerkkinä voidaan käyttää tiedostojen muokkausta. Jos halutaan esim. avata, tallentaa tai poistaa tiedostoja, on kerrottava kyseisen tiedoston yksiselitteinen nimi. Samat käytännöt kuuluvat myös nimipalveluihin. Käsiteltäessä tietoverkon kautta resursseja, on niihin viitattava nimen perusteella. Nimipalveluiden avulla voidaan paikantaa nämä resurssit niiden nimien avulla. (Talvivaara [Viitattu 13.5.2016].)

Nimipalveluista puhuttaessa vastaan tulee käsite ”**hakemistopalvelut**”. Useissa nimipalveluissa on ominaisuuksia, jotka mahdollistavat hakemistopalveluiden (directory services) käytön. Hakemistopalvelu käsittelee objekteja (object) niiden nimien perusteella. Hakemistopalvelu mahdollistaa myös tavan muokata objekteja niiden ominaisuuksien (attributes) avulla. Käytettäessä hakemistopalveluita verkon resursseista puhutaan objekteina, ja palvelun avulla voidaan hakemistojen objekteja muokata. (Talvivaara [Viitattu 14.5.2016].)

Objekti tarkoittaa yleistä käsitettä niistä asioista, mistä hakemistopalveluun talletetaan tietoa. Objektilla voidaan tarkoittaa esimerkiksi tietokonetta, tulostinta, käyttäjää, jne. Objektin tiedot tallennetaan objektin ominaisuuksiin eli attribuutteihin. **Attribuutit** voivat tarkoittaa esim. käyttäjätunnusta, nimeä, puhelinnumeroa, käyttöoikeuksia tai tulostimesta puhuttaessa sen valmistajan tunnusta, tulostusnopeutta, käytettävän paperin kokoa jne. Attribuutit riippuvat käytössä olevasta hakemistopalvelusta, ja mitä tietoja objekteihin voidaan tallentaa. (Talvivaara [Viitattu 14.5.2016].)

Esimerkkejä hakemistopalveluista ovat esimerkiksi Unix-ympäristöissä käytetty **Network Information System** (NIS) ja Novell-ympäristöjen **Novell Directory Service** (NDS). **NIS** on tietokoneverkkoja varten suunniteltu hakemistopalvelu, joka antaa verkkoon kytketyille laitteille tarvittavat tiedot kyseisestä verkosta ja sen palveluista, kuten käyttäjistä, tulostimista ja muista verkoista. **NDS** on samankaltainen hakemistopalvelu, joka tarjoaa tietoa verkon tiedosto- ja tulostuspalveluista. (Talvivaara [Viitattu 14.5.2016].)

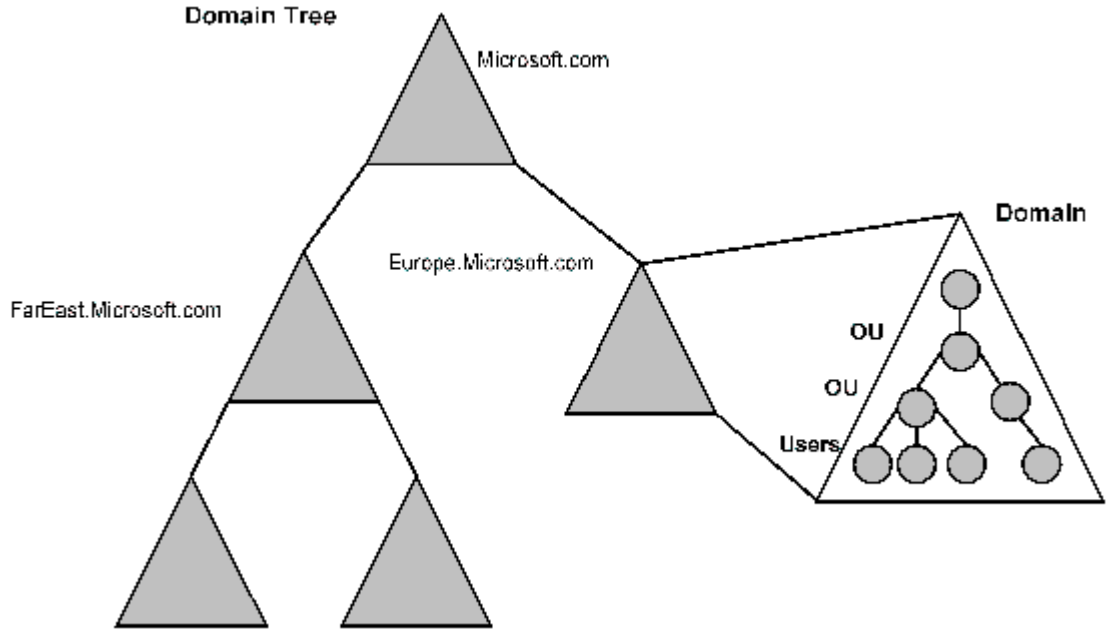
Muita hakemistopalveluita ovat Microsoftin **Active Directory** ja SUN Microsystemsin **SunONE Directory Server**. Kehittyvien hakemistopalveluiden avulla isonkin organisaation tiedot voidaan hallita sujuvasti ja tehokkaasti. (Talvivaara [Viitattu 14.5.2016].)

2.1 Active Directory

Microsoftin kehittämä Active Directory on hakemistopalvelu, joka on osa Windows 2000 -käyttöjärjestelmäarkkitehtuuria. Active Directorystä puhuttaessa käytetään yleensä lyhennettä AD. AD:n ensimmäiset testiversiot esiteltiin vuonna 1996, mutta virallinen julkaisu tapahtui vuonna 1999–2000 osana Windows 2000 -käyttöjärjestelmää. (Talvivaara [Viitattu 15.5.2016].)

2.2 AD-palvelun rakenne

Active Directoryn sisältämä data esitetään käyttäjälle hierarkkisena järjestelmänä. Jokaiseen järjestelmän merkintään viitataan objektina. Näitä objekteja on kahta eri tyyppiä: säiliö (container) ja säiliötön (non-container), jota kutsutaan myös lehtisolmuksi (leaf node). Esimerkkinä hierarkisesta järjestelmästä on puu-rakenne. Puun juuri ja sen rungosta lähtevät oksat ovat säiliöitä. Jokainen säiliö voi sisältää muita säiliöitä tai lehtisolmuja. Huomattavana erona on, että lehtisolmu ei voi sisältää muita säiliöitä. (Richards, Allen & Lowe-Norris 2006, 15.) Kuviossa 1 havainnollistetaan Active Directoryn hierarkinen järjestelmä.



Kuvio 1. Active Directoryn hierarkinen järjestelmä
(Microsoft 2000.)

Juurena toimii Microsoft.com, josta lähtee haarautumaan kaksi säiliötä. Näistä FarEast-säiliö sisältää kaksi muuta säiliötä. Europe-säiliön sisällöstä on esitetty ”räjähdyskuva.” Säiliöiden väliset yhteydet kuvaavat ns. isä-lapsi-suhdetta. Microsoft on FarEast- ja Europe-säiliöiden isä. Vastaavasti Europe- ja FarEast-säiliöt ovat Microsoft-säiliön lapsia. Tällainen puujärjestelmä tunnetaan Active Directoryssä verkkoalueena (domain). Yleisin säiliötyyppi Active Directoryssa on nimeltään Organizational Unit (OU). (Richards, Allen & Lowe-Norris 2006, 15-16.)

2.3 Nimeämiskäytännöt

Hakemistopalvelut voivat sisältää miljoonia objekteja. Jokaisen objektin on oltava löydettävissä ja tunnistettavissa. Active Directoryssä jokaisella objektilla on oma globaalisti uniikki tunnisteensa (GUID). GUID on 128-bittinen numero, jonka takia sitä on vaikea muistaa eikä siitä voi päätellä objektin hierarkiaa. Tämän takia yleisempi tapa viitata objekteihin on AD-polun (ADsPath) avulla. (Richards, Allen & Lowe-Norris 2006, 17.)

ADsPaths. AD-polut kuvataan LDAP-standardien mukaisen kieliopin ja sääntöjen avulla. Esimerkiksi kuvion 1 *microsoft.com*-juuren AD-polku kuvataan seuraavasti:

LDAP://dc=microsoft,dc=com

Esimerkissä DC tarkoittaa verkkoalueen nimen osaa (Domain Name Component). (Richards, Allen & Lowe-Norris 2006, 17.)

DN. DN-nimellä (Distinguished Name) viitataan tiettyyn objektiin hakemistopuussa (Directory Information Tree). Esimerkiksi *microsoft.com*-verkkoalueen Users-säiliössä sijaitsevan järjestelmävalvojan tilin AD-polku kuvataan seuraavasti:

LDAP://cn=Administrator,cn=Users,dc=microsoft,dc=com

DN-nimessä LDAP–tunniste on poistettu, joten se kuvataan näin:

cn=Administrator,cn=Users,dc=microsoft,dc=com

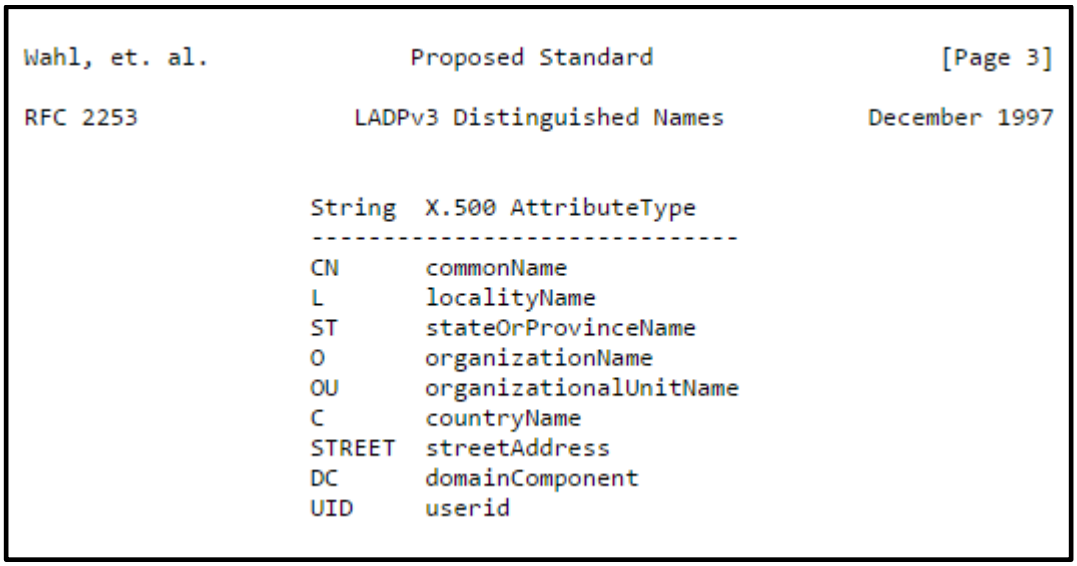
RDN. Suhteellista DN-nimeä (RDN) käytetään viittamaan tiettyyn objektiin, joka sijaitsee kyseessä olevan isä-säiliön sisällä. Esimerkiksi edellisen DN-nimen RDN kuvataan näin:

cn=Administrator

OU. Nämä polut ovat yhdistelmiä etuliitteistä ja nimistä, jotka on erotettu yhtäsuuruus-merkeillä (=). Active Directoryssä yleinen Organizational Unit (OU) ilmaistaan AD-polussa seuraavalla tavalla:

cn=Matti Virtanen,ou=Revontuli IT Oy,dc=microsoft,dc=com

CN. Kaikki RDN-nimet käyttävät etuliitettä ilmaisemaan, minkä tyypin objektista on kyse. Jos tyypille ei ole omaa etuliitettä, käytetään oletuksena cn-etuliitettä (Common Name). (Richards, Allen & Lowe-Norris 2006, 17-18.) Kuviossa 2 on lista yleisimmistä objektien tyypeistä, jotka esiintyvät nimipalveluiden toteutuksissa.



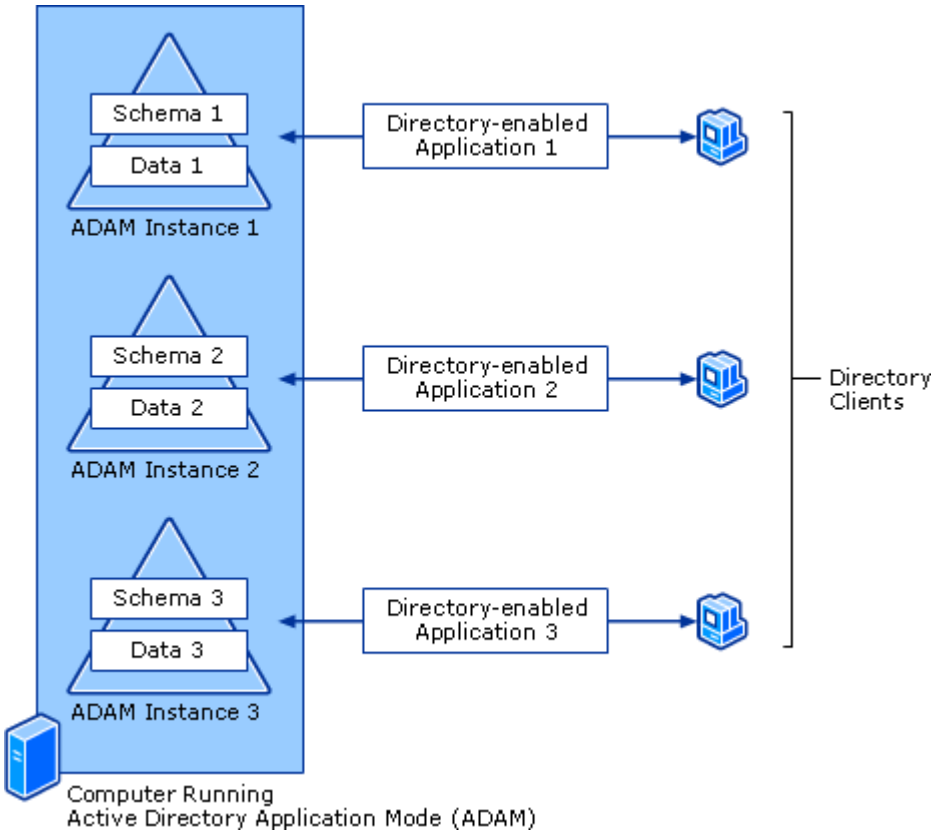
```
Wahl, et. al.              Proposed Standard                    [Page 3]

RFC 2253                LADPv3 Distinguished Names          December 1997


                  String  X.500 AttributeType
                  -------------------------------
                  CN      commonName
                  L       localityName
                  ST      stateOrProvinceName
                  O       organizationName
                  OU      organizationalUnitName
                  C       countryName
                  STREET  streetAddress
                  DC      domainComponent
                  UID     userid
```

Kuvio 2. Yleisimmät objektien tyypit nimipalveluiden toteutuksessa (Wahl, Kille & Howes 1997.)

2.4 ADAM

ADAM (Active Directory Application Mode) on kevennetty versio Active Directorysta. ADAM-ohjelmalla voidaan tehdä samoja toimintoja kuin Active Directorylla, mutta ilman tarvetta asentaa tiettyjä lisäosia, joita vaaditaan Active Directoryn käyttöä varten. ADAM voi toimia palvelimella tai suoraan Windowsin työpöydällä omana ohjelmanaan. ADAM tukee myös autentikointi- ja autorisointi-palveluja. Se voi tukea jo olemassa olevaa Windowsin tai paikallisen tietokoneen autentikointia. ADAM-ohjelmalla on myös oma autentikointi-palvelunsa, jonka voi kohdentaa turvaamaan tiettyjä ohjelmia. (Richards, Allen & Lowe-Norris 2006, 408-409.)

ADAM-palvelusta voi olla useita ilmentymiä samalla koneella. Jokaisella ilmentymällä on omat resurssinsa. Kuviossa 3 näkyy esimerkki lähiverkosta, joka käyttää ADAM-palvelua. Kuvion koneet voivat sijaita Active Directory verkkoalueen sisällä tai ulkona. (Technet Microsoft 2014.)



Kuvio 3. Esimerkki verkosta, jossa on käytössä ADAM (Technet Microsoft 2014.)

3 ASP.NET

Ennen internetiä suurin osa sovelluskehityksestä keskittyi tietokoneen työpöydältä suoritettaviin sovelluksiin. Microsoft käytti näiden sovellusten ohjelmoimiseen useimmiten Visual Basic (VB) -koodikieltä. Nämä sovellukset käyttivät tietokoneen omia resursseja toimintojen suorittamiseen. Visual Basic -ohjelmoinnissa pystyi käyttämään integroitua ohjelmointiympäristöä, jonka tarkoituksena oli helpottaa ohjelmoijien työtä. Ohjelmoijat pystyivät käyttämään ns. kontrollereita, jotka olivat valmiita malleja ohjelmissa käytettävistä osista (esim. painike-kontrolleri). Näitä kontrollereita voitiin vetää hiirellä työstettävän ohjelman ikkunaan ja ohjelmoida niiden taustalle suoritettavaa koodia. Esimerkiksi painike-kontrolleri voitiin vetää työstettävään ikkunaan ja kirjoittaa sen taustalle koodi, joka suoritettiin painettaessa kontrolleria. (Evjen, Hanselman, Muhammad, Sivakumar & Rader 2006, 1.)

Internetin tullessa 1990–luvun puolessa välissä esiintyi ongelmia. Microsoft ei pystynyt siirtämään Visual Basic -ohjelmointiympäristöä internetpohjaisten sovellusten kehittämiseen. Internetsovellukset käyttivät web-palvelimen resursseja toimintojen suorittamiseen. Lisäksi internetsovellukset loivat yhden ilmentymän ohjelmasta internetiin. Ilmentymää pystyi käyttämään kuka tahansa. Tämän ansiosta sovellukset pystyttiin päivittämään internetin kautta. Internetsovelluksiin verrattuna aiemmat työpöydältä suoritettavat ohjelmat eivät käyttäneet internetin resursseja, jonka takia ne olivat huomattavasti hitaampia. (Evjen, Hanselman, Muhammad, Sivakumar & Rader 2006, 1-2.)

Tästä syystä Microsoft kehitti **Active Server Pagesin** (ASP). ASP-mallin avulla pystyttiin kehittämään selaimen kautta suoritettavia web-sivuja. Sivuja pystyi kehittämään samalla tavalla kuin Visual Basicin ohjelmistoympäristöllä. Nämä sivut pystyivät käyttämään VB- ja JavaScript-skriptikieliä, jotka suoritetaan ennakkoon internetin kautta web-palvelimessa. ASP-mallin kehittyessä sovellukset pystyivät käyttämään yhä useampia ohjelmointikieliä. ASP-sivu saattoi sisältää HTML-, VB-, JavaScript- ja CSS-kieliä sekä monia muita kieliä. Kun Microsoft julkaisi .NET Framework version 1.0, liitettiin ASP sen osaksi. Tästä johtuen nimi muutettiin ASP.NET-muotoon. (Evjen, Hanselman, Muhammad, Sivakumar & Rader 2006, 2.)

3.1 Yleistä

ASP.NET on web-sovellusten tekoon tarkoitettu alusta joka on rakennettu .NET-viitekehyksen päälle. Tämän ansiosta ASP.NET tukee kaikkia CLR-tuettuja koodauskieliä, kuten C# ja Visual Basic. (Microsoft [Viitattu 12.10.2016].) ASP.NET-mallin versio 1.0 ilmestyi vuonna 2002. Opinnäytetyön kirjoitushetkellä uusin versio on ASP.NET 4.6, joka tulee osana Visual Studio 2015 -ohjelmointiympäristöä. (Microsoft 2015.)

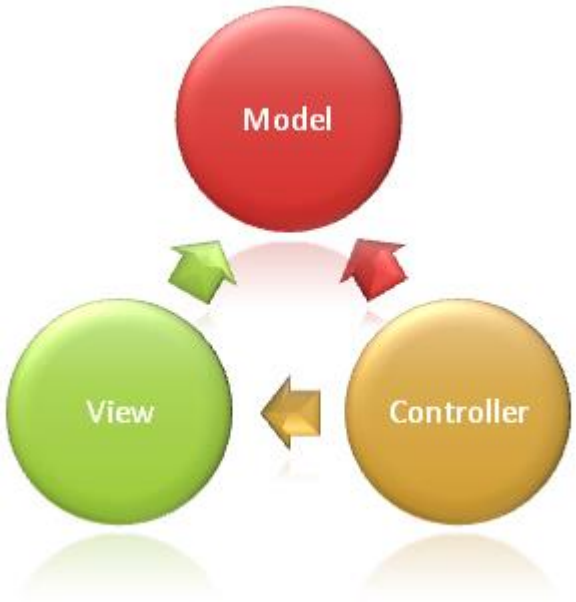
3.2 ASP.NET-tuotantomallit

ASP.NET on suunniteltu helpottamaan web-sovellusten ohjelmointia. Tästä syystä ASP.NET tarjoaa valmiita malleja web-sovellusten kehittämiseen. Nämä mallit ovat nimeltään Web Pages, Web Forms ja MVC.

Web Pages on yksinkertainen malli dynaamisten web–sovellusten luomiseen. Malli perustuu yhteen staatiseen HTML-sivuun, jonka lähdekoodiin kirjoitetaan tarvittavat toiminnot. Lähdekoodissa käytetään Razor-syntaksia dynaamisten toimintojen luomiseen. (FitzMacken 2014.)

Web Forms. Web Forms on kehittyneempi versio aiemmasta ASP-sivujen ohjelmoinnista. Sivujen teossa käytetään edelleen kontrollereita, mutta Web Forms tukee .NET-viitekehyksen koodikieliä, kuten C# ja VB.NET. (Microsoft [Viitattu 22.11.2016].)

MVC. MVC (Model View Controller) tarkoittaa arkkitehtonista mallia, jossa sovellus eritellään kolmeksi eri osaksi. Nämä osat ovat malli (Model), näkymä (View) ja kontrolleri (Controller), jotka on kuvattu kuviossa 4. (Microsoft ASP.NET Team 2009.)



Kuvio 4. MVC-malli
(Microsoft ASP.NET Team 2009.)

Malli–osalla kuvataan järjestelmän datan tallentaminen, ylläpito ja käsittely. Malli-osa on tietynlainen tietokantakuvaus ohjelmasta. **Näkymä**–osio määrittelee ohjelman käyttöliittymän ulkoasun ja kuinka tieto esitetään käyttöliittymässä. **Kontrolleri**–osa vastaanottaa käyttäjältä saadut käskyt sekä muuttaa malli- ja näkymä-osaa niiden perusteella. ASP.NET MVC käsittelee näitä ohjelman osia erikseen, mikä mahdollistaa niiden testaamisen yksitellen. ASP.NET MVC ei käytä ASP.NET-kontrollereita, vaan sivut kirjoitetaan kokonaan HTML-kielellä. (Microsoft ASP.NET Team 2009.)

3.3 ASP.NET-kontrollerit

ASP.NET web -sovellusten luonnissa voidaan käyttää useita eri tyyppisiä kontrolleja. Nämä tyypit ovat HTML server- , validation-, user- ja web server –kontrollerit. (Microsoft [Viitattu 7.11.2016].)

HTML server -kontrollerit ovat tyypillisiä HTML-elementtejä, jotka on muokattu suorittamaan palvelimen kautta kirjoitettua koodia. Tämä mahdollistetaan lisäämällä HTML-elementtiin ’runat=”server”’-attribuutti. (Microsoft [Viitattu 7.11.2016].)

Validation-kontrollerit tarkistavat käyttäjän antamat tiedot virheiden varalta, esimerkiksi tekstikentän sisällön tarkistamisen. Jos tekstikenttä on tyhjä tai sisältää tiettyjä erikoismerkkejä, kuten kaksoispisteitä, suoritusta ei toteuteta ennen kuin tekstikentän sisältö on määritysten mukaista. (Microsoft [Viitattu 8.11.2016].)

User-kontrollerit ovat sivuja, joiden avulla ohjelmoija voi luoda omia kontrollereita. Nämä sivut voidaan liittää osaksi ASP.NET–sovellusta, ja niihin voi lisätä ohjelmoijan haluamat tyylitiedot ja muita kontrollereita. Sivujen tarkoituksena on tarjota ohjelmoijalle mahdollisuus luoda haluttuja toimintoja, jotka eivät löydy valmiina ASP.NET-mallin kontrollereista. (Microsoft [Viitattu 9.11.2016].)

Web server -kontrollerit ovat palvelinkontrolleja, jotka muodostavat pääosan ASP.NET–sovelluksen sisällöstä. Palvelinkontrollerit sisältävät laajan kirjaston yleisiä standardi-kontrollereita, kuten myös monimutkaisimpia kontrolleja, kuten kalentereita tai taulukoita. Kontrolleja luodessa kerrotaan kontrollin tyyppi, sen ID ja runat=”server”-attribuutti. (Microsoft [Viitattu 7.11.2016].) Esimerkiksi painike–kontrolleri kuvataan sovelluksen tyylitiedostossa seuraavalla tavalla:

```
<asp:button attributes runat="server" id="Button1" />
```

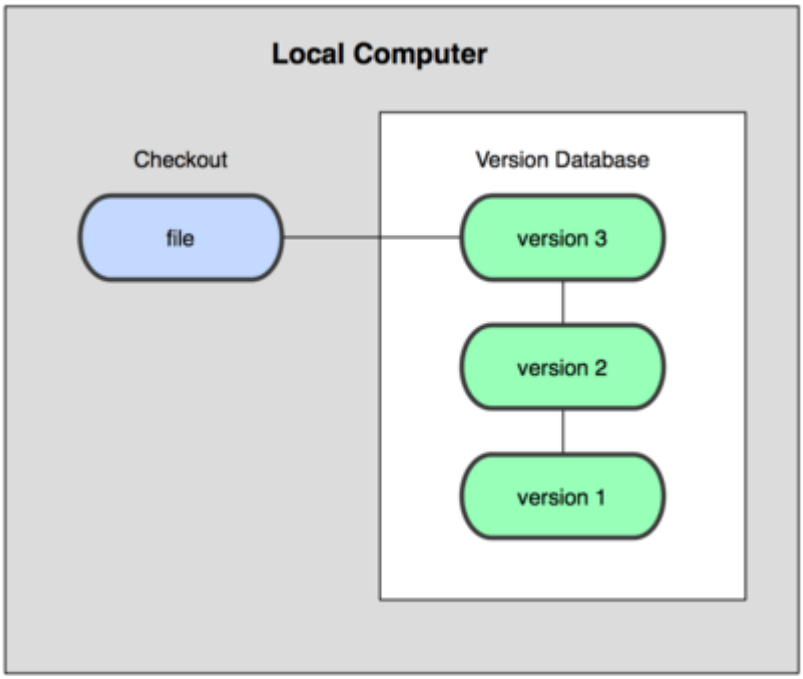
4 VERSIONHALLINTA

Esimerkiksi työraporttia tehtäessä dokumenttia muokataan monta kertaa useiden päivien aikana. Vaarana kuitenkin on, että dokumentista katoaa jonkin muutoksen takia tärkeää tietoa. Tämän estämiseksi on mahdollista käyttää versionhallintaa dokumentin varmuuskopiointia varten. Versionhallinnalla voidaan säilyttää dokumentin eri versiot joka muutoskerralla. Sillä voidaan myös saada dokumentin aiempi versio haettua menetetyn tiedon palauttamiseksi. Vaikka versionhallinta on alun perin suunniteltu ohjelmointikehitystä varten, voidaan sillä varastoida melkein mitä tahansa tiedostoja ja seurata niiden muutoksia. (Chacon & Straub 2009.) Erilaisia versionhallintajärjestelmiä ovat esimerkiksi Git, TortoiseSVN, Mercurial ja Bazaar.

4.1 Versionhallinnan vaihtoehdot

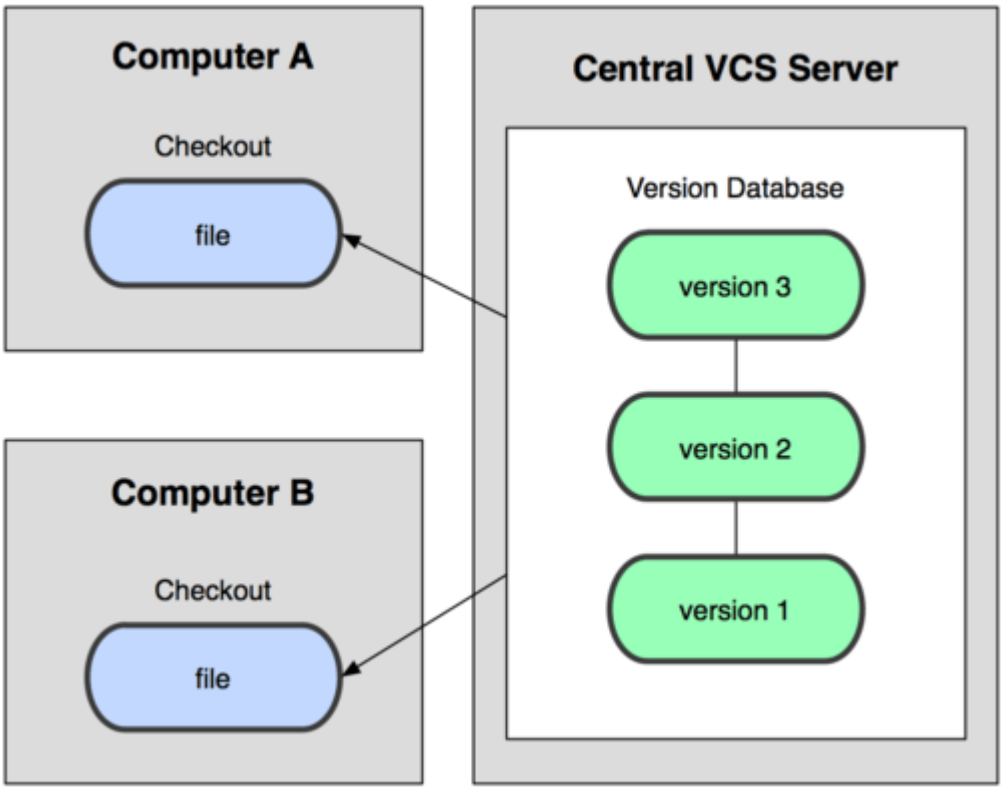
Ohjelman varmuuskopioinnin voi tehdä yksinkertaisesti kopioimalla tiedostot toiseen kansioon, mutta tämä altistaa helposti virheille. Kansiot saattavat mennä sekaisin, jolloin muokataan vääriä tiedostoja. Tätä ongelmaa varten kehitettiin paikallinen versionhallintajärjestelmä. (Chacon & Straub 2009.)

Paikallinen versionhallintajärjestelmä varastoi tiedostojen muutokset ohjelmoijan tietokoneen omaan tietokantaan. Järjestelmän avulla alkuperäisen tiedoston kopio muokataan muutosten avulla valitun version mukaiseksi. Paikallisen versionhallintajärjestelmän käytössä on kuitenkin ongelmia, kun tehdään yhteistyötä muiden kehittäjien kanssa. Kehittäjien on pakko käyttää samaa tietokonetta tai käyttää etähallintaa. (Chacon & Straub 2009.)



Kuvio 5. Paikallinen versionhallinta -diagrammi
(Chacon & Straub 2009.)

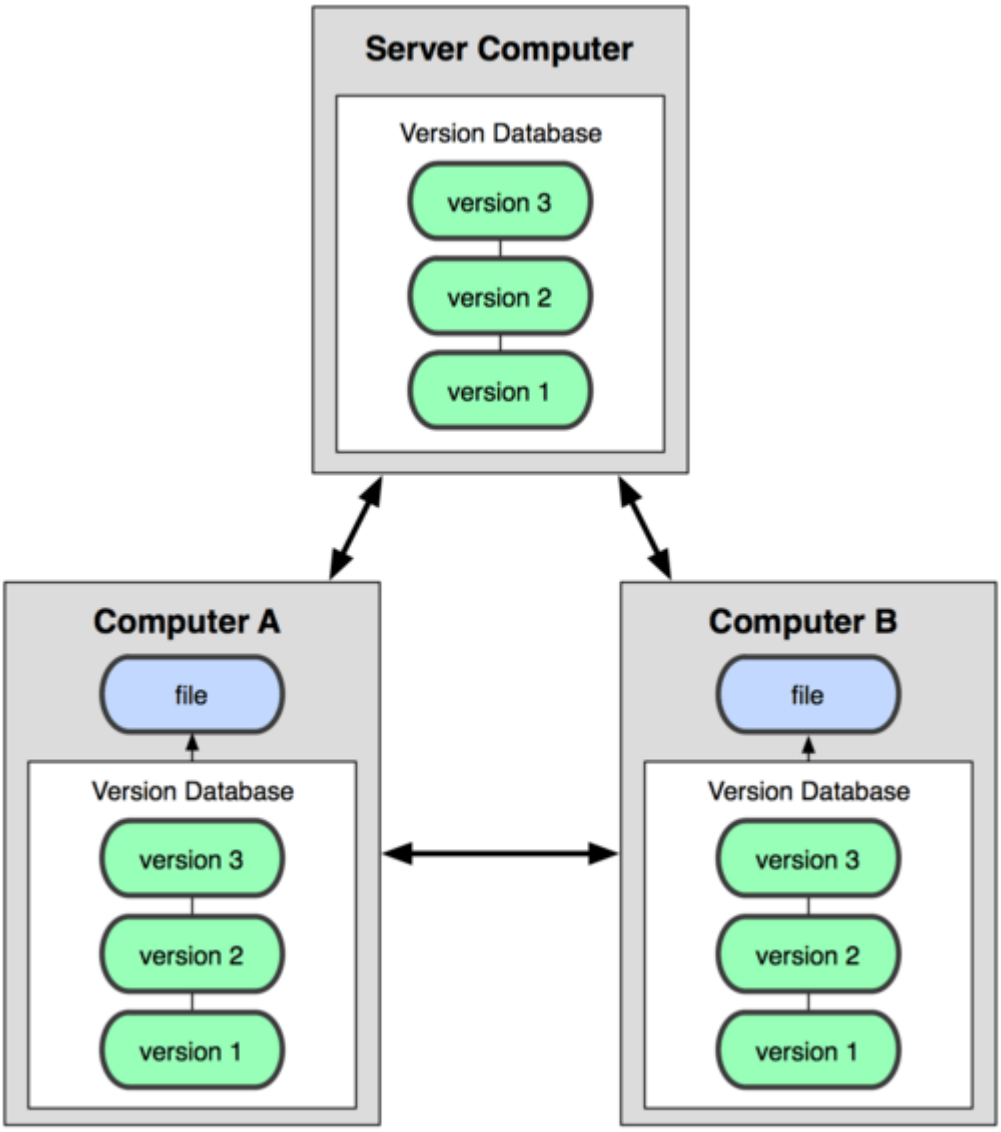
Ongelman ratkaisuksi kehitettiin keskitetyt versionhallintajärjestelmät. **Keskitetyssä versionhallinnassa** jokaisella projektilla on yksi ainoa tietokanta, joka sijaitsee omalla palvelimellaan. Palvelimen tietokanta sisältää tiedostojen eri versiot, ja ohjelmoijat voivat muokata niitä ja luoda niistä omat versionsa. Palvelimen versionhallinnan valvoja voi seurata mitä tiedostoja kukin ohjelmoija muokkaa. (Chacon & Straub 2009.)



Kuvio 6. Keskitetty versionhallinta -diagrammi
(Chacon & Straub 2009.)

Vaikka keskitetyssä versionhallintajärjestelmässä on lukuisia myönteisiä puolia verrattuna paikalliseen versioon, se sisältää riskinsä. Kaiken tiedon sijoittaminen yhdelle palvelimelle voi aiheuttaa tuhoisat seuraukset, jos varmuuskopioita ei ole tallella ja tietokanta tuhoutuu. Palvelimen huollon aikana ohjelmoijat eivät voi tehdä yhteistyötä eivätkä lisätä uusimpia tiedostoja järjestelmään. Sama ongelma ilmenee paikallisessa versionhallintajärjestelmässä. (Chacon & Straub 2009.)

Hajautettu versionhallintajärjestelmä ei kärsi paikallisen ja keskitetyn järjestelmän ongelmista. Järjestelmän käyttäjät hakevat viimeisimmän version lisäksi koko tietolähteen. Tämän ansiosta palvelimen tietokannan tuhoutuminen ei ole enää ongelma, sillä tiedot voidaan kopioida takaisin palvelimelle kenen tahansa käyttäjän tietokoneelta. Monet hajautetut järjestelmät on kehitetty toimimaan useiden etätietolähteiden kanssa, joten niiden avulla saman projektin työstäminen useiden eri kehittäjien kanssa samanaikaisesti on mahdollista. (Chacon & Straub 2009.)



Kuvio 7. Hajautettu versionhallinta -diagrammi
(Chacon & Straub 2009.)

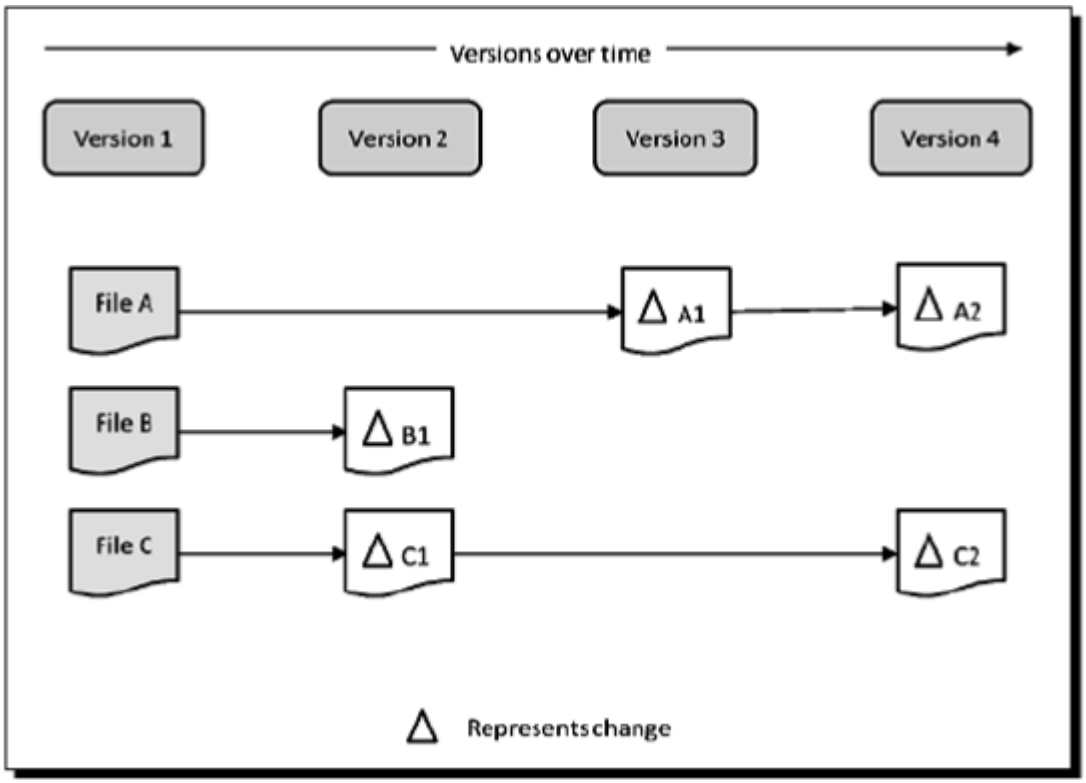
4.2 Git

Yksi suosituimmista versionhallintajärjestelmistä on Git. Sen ensimmäinen prototyyppiversio ilmestyi huhtikuussa 2005 ja sen suunnittelussa otettiin huomioon edeltävien versionhallintaohjelmien viat. Gitin kehittäjä on Linux-käyttöjärjestelmän tekijä Linus Torvalds. Gitin arkkitehtuuri on suunniteltu tukemaan sen kolmea periaatetta: jakamattomuus, tehokkuus ja turvallisuus. (Ravishankar 2013, 15.)

Jakamattomuudella tarkoitetaan sitä, että järjestelmä keskittyy täysin senhetkiseen tehtävään. Järjestelmän tulisi suorittaa toiminto täysin tai antaa

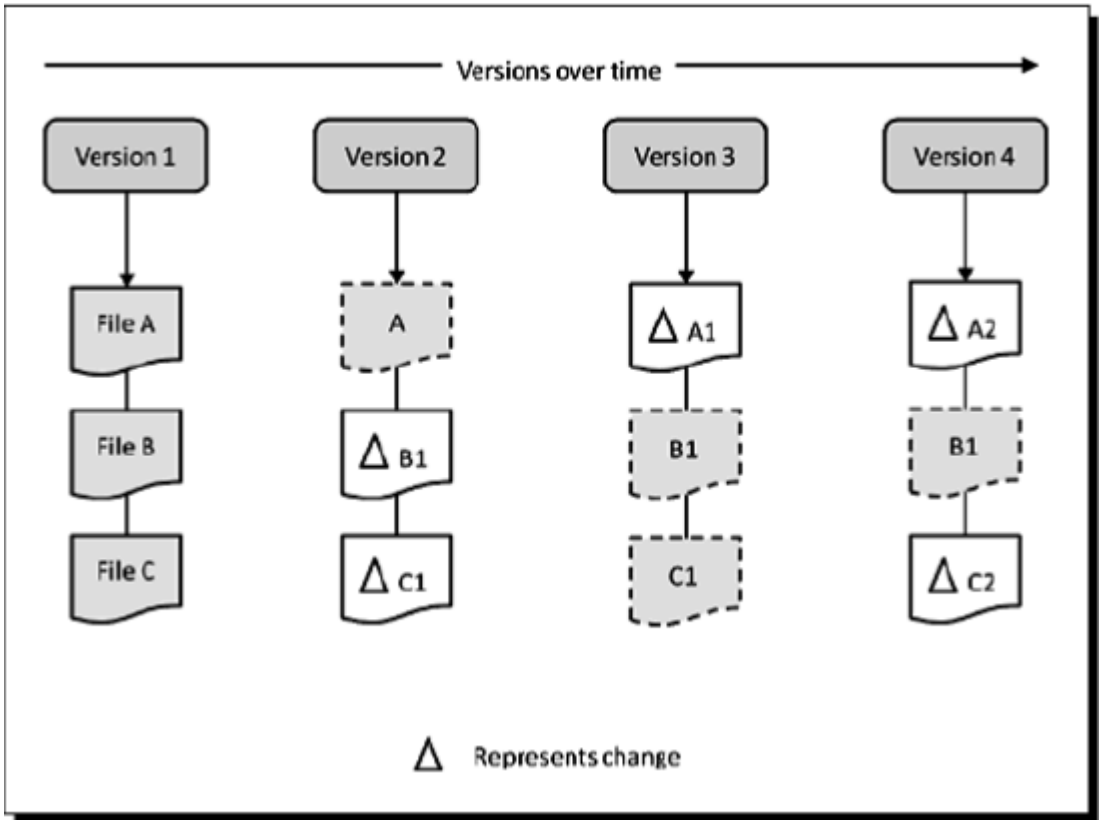
hylkäävä vastaus. Tarkoituksena on välttää osittaisia operaation suorituksia, jottei osa tiedoista katoaisi suorittamisen yhteydessä. (Ravishankar 2013, 16.)

Useimmat muut versionhallintajärjestelmät (CVS, Perforce, Bazaar ym.) näkevät käyttäjän datan kokoelmina tiedostoja ja niihin tehtyinä muutoksina. Git on **tehokas** suorittamaan miljoonien tiedostojen käsittelyn sekunneissa. Tämä perustuu Gitin kykyyn käsitellä tiedostoja. (Ravishankar 2013, 16.)



Kuvio 8. Esimerkki kuinka muut versionhallintajärjestelmät näkevät tiedostot (Ravishankar 2013, 17.)

Git ottaa tilannevedoksen (snapshot) kaikista tiedostoista, jolloin se näkee tiedostojen väliset yhteydet ja suunnittelee toimintansa niiden perusteella. Joka kerta kun ohjelmaa muokataan, Git tekee uuden tilannevedoksen. Git ei siis tallenna uusia versioita muuttumattomista tiedostoista, vaan säilyttää aiemmat tilannevedokset ja jättää viittauksen niihin uudessa versiossa. Tämän ansiosta Git pystyy suoriutumaan tehokkaasti eri toiminnoissa. (Ravishankar 2013, 17.)



Kuvio 9. Gitin tilannekuvien varastointi ajan kuluessa
(Ravishankar 2013, 17.)

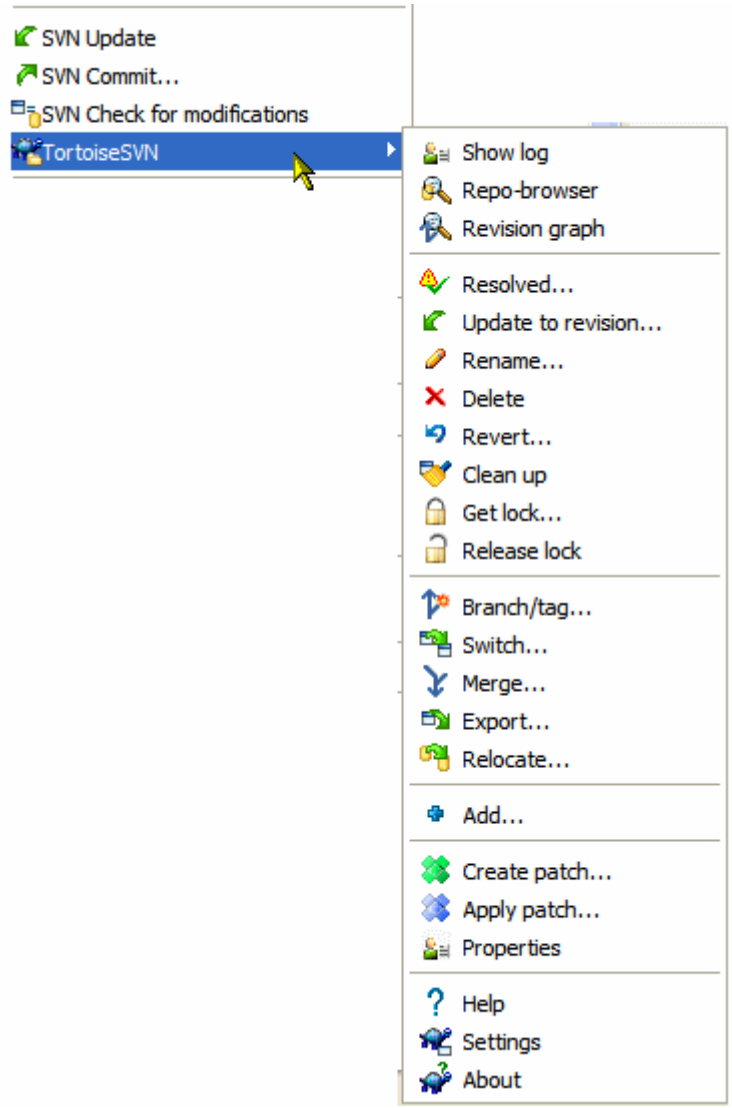
Git pitää huolen tiedostojen **turvallisuudesta** salaamalle ne tarkistussumman avulla SHA-1-tarkisteeksi. Salaamisen jälkeen niihin viitataan tarkistussumman avulla. Tarkisteen pituus on 40 heksadesimaalista (0–9 ja a–f) merkkiä. Tarkistussumma luodaan tiedoston rakenteen perusteella. Tämän ansiosta on mahdotonta muuttaa salattuja tiedostoja ilman että Git huomaisi ne. (Ravishankar 2013, 18.) Esimerkki tarkistussummasta voi olla seuraavanlainen:

24b9da6552252987aa493b52f8696cd6d3b00373

4.3 TortoiseSVN

TortoiseSVN on versionhallintaohjelma Windowsille ja perustuu Apache™ Subversionin (SVN) avoimeen lähdekoodiin. TortoiseSVN pitää kirjaa tiedostojen muutoksista. Se eroaa Gitistä, koska sen graafinen käyttöliittymä integroituu Windowsin Resurssienhallintaan. (Küng, Onken & Large [Viitattu 19.3.2016].) Täten

oman ohjelman versionhallinnan voi suorittaa suoraan Windowsin työpöydällä (kuvio 10).



Kuvio 10. TortoiseSVN Windowsin resurssienhallinnassa
(The TortoiseSVN Team 2016.)

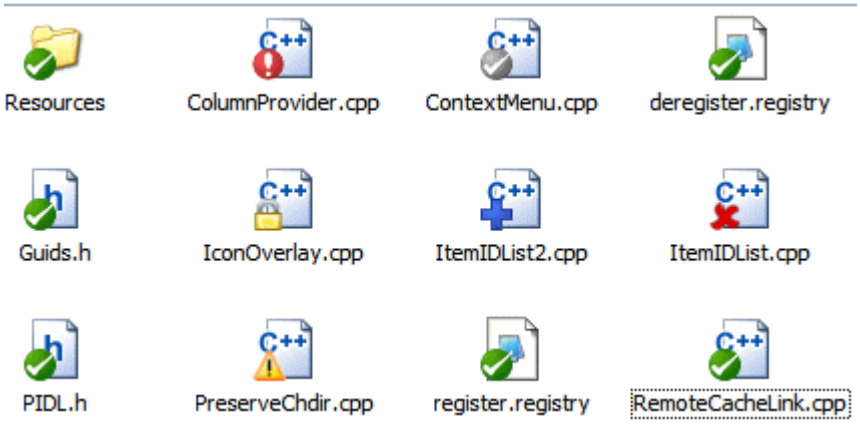
Ohjelmaa kehitetään GNU General Public Licensen (GPL) -lisenssillä eli ohjelmassa käytetty koodi on vapaasti kaikille käytössä. Myös lähdekoodi on avointa, joten oman kaupallisen version teko on mahdollista. (GNU 1991.) Lisenssi on kuvattu kuviossa 11.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
```

Kuvio 11. GNU-lisenssi
(GNU 1991)

4.3.1 TortoiseSVN-ohjelman ominaisuudet

TortoiseSVN-ohjelman graafisen käyttöliittymän ansiosta ohjelma tarjoaa tiettyjä ominaisuuksia, jotka eroavat Gitin-versionhallintaohjelmasta. (Küng, Onken & Large [Viitattu 21.3.2016].) Esimerkiksi kuviossa 12 näkyvät ohjelman kuvakepäällykset.



Kuvio 12. TortoiseSVN-kuvakepäällykset
(The TortoiseSVN Team 2016.)

Kuvakepäällykset kuvaavat tiedostojen tilan versionhallinnassa. Vihreät kuvakkeet ilmaisevat tiedostojen olevan ajan tasalla. Punaiset ristit taas tarkoittavat, että kyseiset tiedostot on poistettu uusimmasta versiosta.

TortoiseSVN voi hyödyntää myös Subversion-kirjaston ominaisuuksia, kuten versioituvia hakemistoja. Subversion pystyy tallentamaan versionhallintaan sekä tiedostot että kansiot tallentamalla koko kansiopuun muutokset. (Küng, Onken & Large [Viitattu 21.3.2016].)

5 AUTENTIKOINTI JA AUTORISOINTI

Kun kehitetään ohjelmia jotka suorittavat tärkeitä operaatioita, on niissä otettava huomioon myös tietoturva. Ensimmäisiä huomioon otettavia asioita on tieto siitä, kuka ohjelmaa käyttää. On myös tärkeää estää vieraiden käyttäjien pääsy kyseiseen ohjelmaan. Näistä toiminnoista käytetään termejä autentikointi ja autorisointi. (Apple 2012.)

Autentikoinnilla tarkoitetaan käyttäjän tai palvelun tunnistamista. Autentikoinnissa ohjelma lukee käyttäjän antaman tunnuksen ja salasanan ja vertaa niitä tietokannassa olevaan tietoon. Jos tiedot varmistuvat oikeiksi, ohjelma todentaa käyttäjän. (Apple 2012.)

Autorisoinnilla tarkoitetaan käyttäjän oikeutta suorittaa tietty toiminto. Ero autentikointiin tulee ilmi monimutkaisimmissa järjestelmissä. Esimerkiksi jos yhdellä tietokoneella on kaksi eri käyttäjää, jotka kirjautuvat sisään omilla tunnuksillaan. Kumpikin voi luoda omia tiedostojaan, mutta he eivät voi muokata toistensa tiedostoja, sillä heiltä puuttuvat oikeudet sitä varten. (Apple 2012.)

5.1 Autentikointitavat

Seuraavaksi käsitellään muutamia esimerkkejä autentikointi- ja autorisointi-palveluista.

5.1.1 Windows

Integroitu Windows-autentikointi tunnistaa käyttäjän hänen Windows-käyttäjätunnuksensa avulla. Windows-autentikointi on osa Microsoftin Internet Information Service -palvelua (IIS). IIS-palvelu on Microsoftin kehittämä palvelinohjelmistokokonaisuus, jolla voidaan perustaa verkkoa hallinnoiva palvelin. IIS pystyy tunnistamaan käyttäjät Windows-autentikoinnilla, jos kyseinen verkko käyttää Active Directory -palvelua tai Windows-tilien tunnuksia. (Microsoft [Viitattu 28.9.2016].)

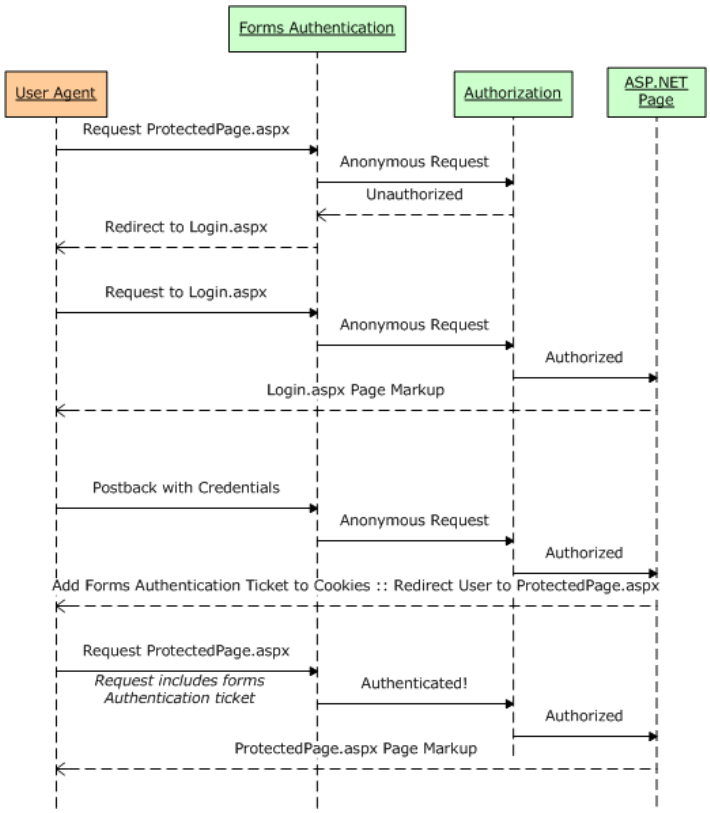
Windows-autentikointi salaa käyttäjän tunnuksen ja salasanan ennen sen lähettämistä verkon yli tietokantaan. Windows-autentikointi voi käyttää Kerberos- tai NTLM-protokollaa autentikoinnissa. Tämä autentikointi-tapa soveltuu parhaiten organisaation sisäisen tietoverkon turvaamiseen, sillä siinä web–palvelin ja verkon tietokoneet sijaitsevat samalla verkkoalueella. (Microsoft [Viitattu 28.9.2016].)

5.1.2 Forms

Forms-autentikoinnissa käyttäjä tunnistetaan ohjelman oman web form -sivun kautta, johon syötetään käyttäjätunnus ja salasana. Jos käyttäjä yrittää suorittaa kiellettyä toimintoa, hänet ohjataan automaattisesti sisäänkirjautumissivulle tunnistautumista varten. Käyttäjän antamat tiedot tarkistetaan ohjelman käyttämässä tietokannassa. (Mitchell 2008.)

Kun tunnistautuminen on vahvistettu, luodaan käyttäjää varten tiketti (Forms Authentication Ticket), joka sisältää tietoa käyttäjästä ja kertoo ohjelmalle, että käyttäjällä on tarvittavat oikeudet. Tiketti säilytetään tyypillisesti selaimen evästetiedoissa, jolloin käyttäjän ei tarvitse joka kerta kirjautua sisään saman istunnon aikana. (Mitchell 2008.)

Kuviossa 13 havainnollistetaan kuinka Forms-autentikoinnilla toimiva ASP.NET-sivu käyttäytyy, kun yritetään suorittaa tunnistamatonta toimintoa.



Kuvio 13. Forms-autentikoinnin toimintaperiaate
(Mitchell 2008.)

Kuviossa 13 käyttäjän selain yrittää päästä ASP.NET-sivulle lähettämällä pyynnön autentikoinnille. Autentikointi käsittelee pyynnön lähettämällä sen eteenpäin valvonta-prosessille, josta se saa hylkäävän vastauksen. Autentikointi siirtää käyttäjän kirjautumissivulle, jonka jälkeen prosessi toistetaan ja autorisointi antaa hyväksytyn vastauksen sekä siirtää käyttäjän ASP.NET–sivulle. (Mitchell 2008.)

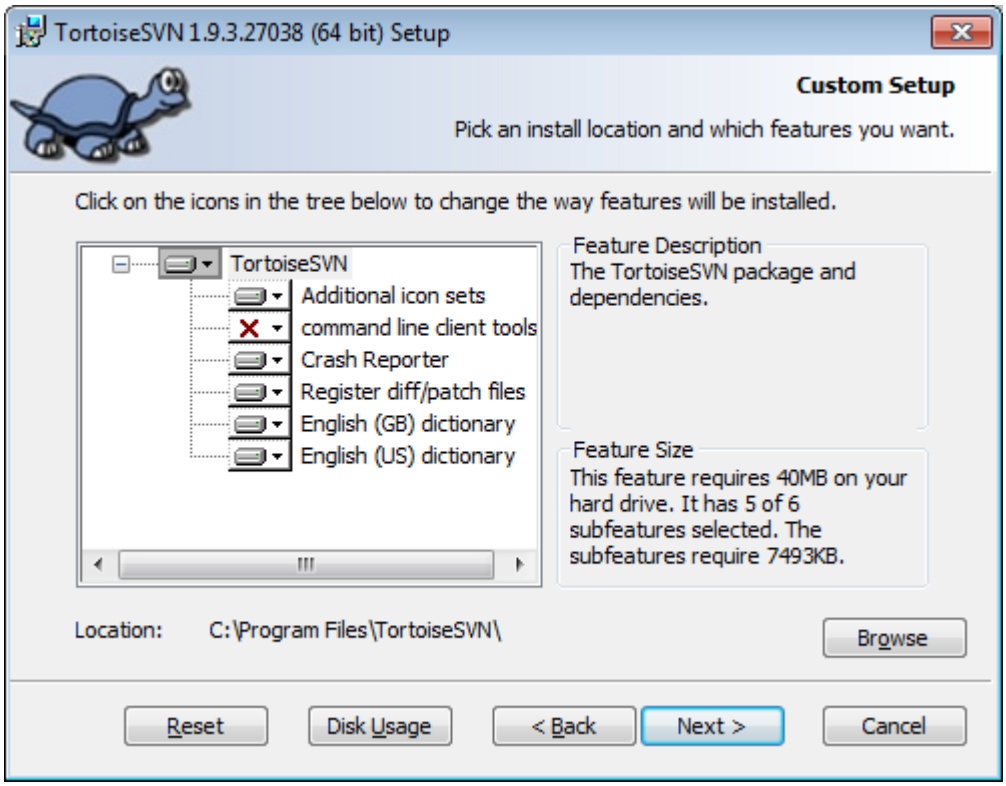
6 ASP.NET-SOVELLUKSEN TOTEUTUS

Työ on toteutettu käyttämällä Visual Studio 2010- ja Visual Studio 2013 -ohjelmointiympäristöjä. Työssä käytettiin ASP.NET-versiota 4.0.

Työn tarkoituksena oli toteuttaa web-sivu, joka saa yhteyden Active Directoryyn joka hakee listan SeAMKin opiskelijoista hakusanan perusteella. Listasta valitun opiskelijan salasana on pystyttävä vaihtamaan joko satunnaiseksi tai ennalta määrätyksi. Lisäksi vaatimuksena oli vaihtoehto lähettää salasana tekstiviestinä. Työn alkuvaiheessa päätettiin lisätä ohjelmiston koodit Jelpparin versionhallintajärjestelmään varmuuskopiointia varten. Koska työ toteutettiin web-sovelluksena, sen tietoturva piti ottaa huomioon. Sovelluksen autentikointimenetelmäksi valittiin integroitu Windows-autentikointi.

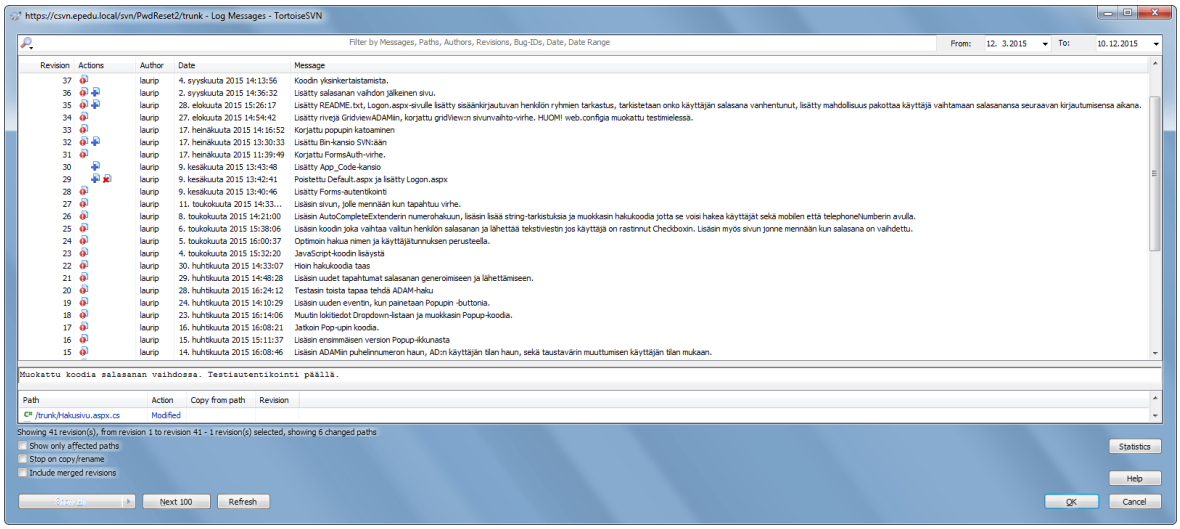
6.1 Versionhallinta

Web–sivun työstämisen ensimmäisessä vaiheessa valmisteltiin versionhallinta. Versionhallinnaksi valittiin TortoiseSVN, koska Jelpparin henkilökunnalla on se jo käytössä. Työn aloittamisen aikana otettiin käyttöön uusin versio TortoiseSVN-ohjelmasta, joka oli 1.8.10 (julkaistu 17.12.2014) (The TortoiseSVN Team [Viitattu: 11.11.2016]). TortoiseSVN sisältää asennusohjelman, jolla voidaan määrittää, mihin kansioon ohjelma asennetaan. Asennusohjelma sisältää myös muutamia lisäosia, joiden asentaminen on valinnaista. Asennusohjelma on esitetty kuviossa 14.



Kuvio 14. TortoiseSVN-ohjelman asennus ja valinnaiset lisäosat.

Sovelluksen työstämisen aikana joka muutoksen jälkeen lisättiin uusi versio sovelluksesta versionhallinnan palvelimelle. Kuviossa 15 on esitettynä osa sovelluksen versiohistoriasta.

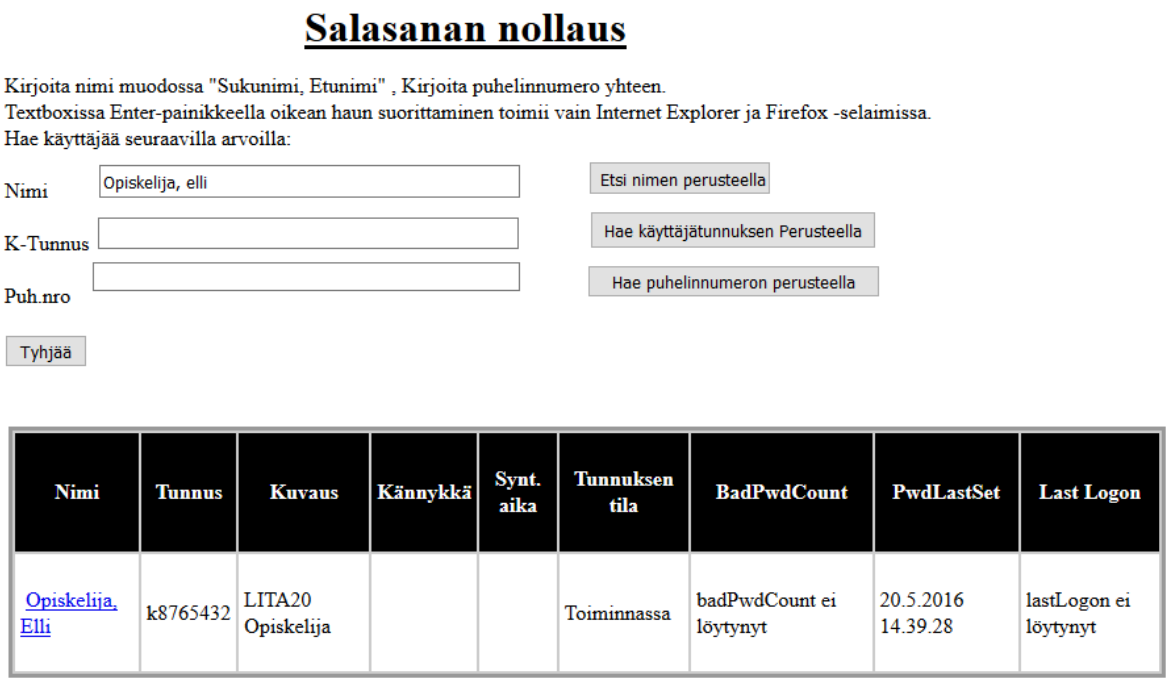


Kuvio 15. Sovelluksen versiohistoria

6.2 ASP.NET-sovelluksen kuvaus

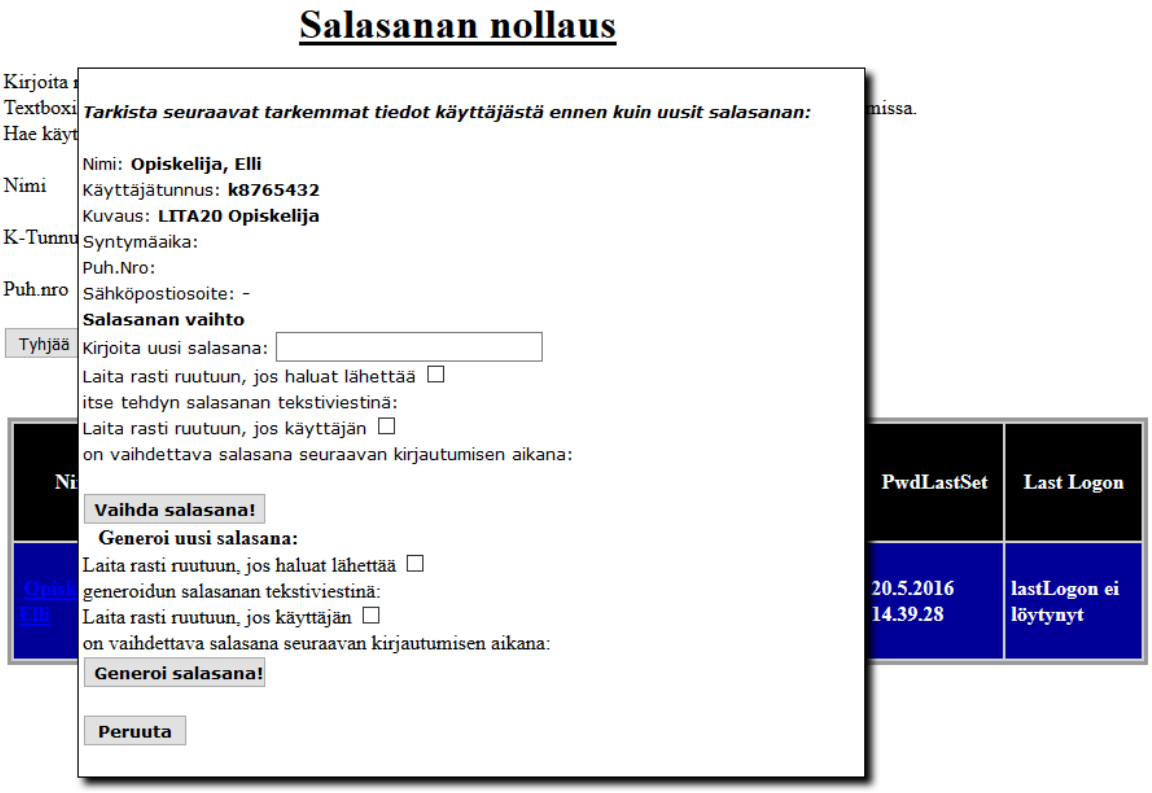
ASP.NET-sivun perustana on kaksi tiedostoa: tyyli- ja kooditiedosto. Tyylitiedostossa määritellään sivun käyttöliittymä HTML-kielellä ja web server -kontrollereilla. Kooditiedostossa määritellään kontrollereiden suoritettava koodi. Visual Studio -ohjelmointiympäristössä on mahdollista nähdä tyylitiedostosta Designer-sivu, joka luo tyylitiedostosta selaimessa näkyvän sivun. Designer-sivua voidaan muokata Visual Studiossa lisäämällä web server -kontrollereita sivulle.

ASP.NET–sivu toteutettiin C#-, HTML-, XML- ja JavaScript-koodikielillä. JavaScript–koodia syötettiin tiettyihin kontrollereihin tyylitiedostossa. Sivussa on kolme tekstikenttää ja neljä painiketta. Tekstikentät toimivat hakukenttinä, joihin kirjoitetaan haettavan opiskelijan nimi, käyttäjätunnus tai puhelinnumero. Vain yksi hakukentistä pitää täyttää, ja haku suoritetaan kyseisen hakukentän painikkeella. Haettu lista opiskelijoista näytetään sovelluksessa avautuvaan GridView-kontrolleriin. GridView-kontrolleri pystyy esittämään saatua dataa ristikkomuodossa (Microsoft [Viitattu 15.11.2016]). Tyhjää–painike tyhjentää GridView-taulukon sisällön. Kuvio 16 kuvaa valmiin sovelluksen käyttöliittymää.



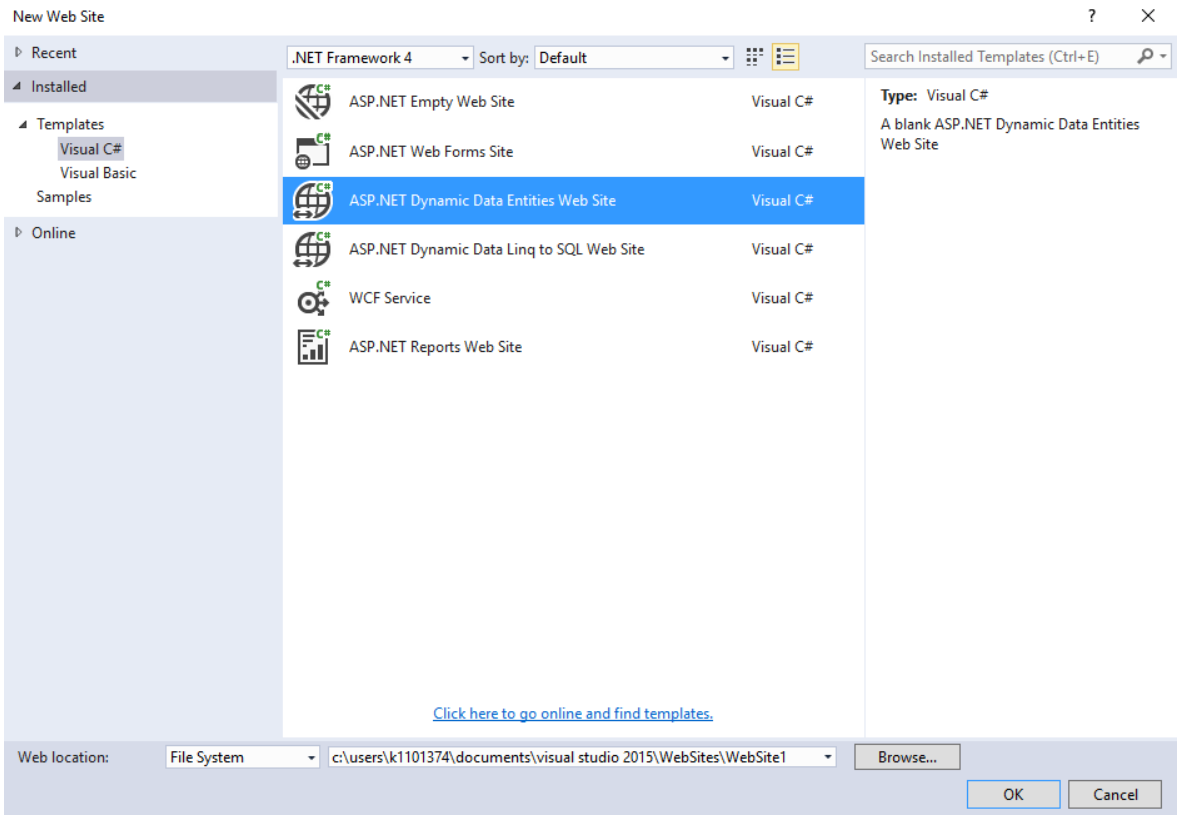
Kuvio 16. Sovelluksen käyttöliittymä

Kuviossa 16 havainnollistetaan myös testiopiskelijan haun tulos. Listassa näkyy opiskelijan nimi, käyttäjätunnus, lyhyt kuvaus opiskelijasta, puhelinnumero, syntymäaika, tunnuksen tila (lukittu tai vanhentunut), viimeisin salasanan vaihto ja viimeisin kirjautuminen. Kun valitaan tietty opiskelija, avautuu uuteen ikkunaan opiskelijan tiedot. Ikkunassa on vaihtoehdot salasanan uusimista varten (kuvio 17).



Kuvio 17. Testikäyttäjän tietojen tarkastelu

ASP.NET–sivun työstäminen aloitettiin valitsemalla Visual Studiosta ASP.NET Dynamic Data Entities Web Site –malli (kuvio 18). Malli valittiin, koska se sisälsi suurimman osan sovelluksen tarvitsemista kirjastoista. .NET-viitekehyksen versioksi valittiin 4.0. Sovellukseen piti asentaa AJAX-kirjasto, josta saatiin tarvittavat kontrollerit ennakoivan haun lisäystä varten.



Kuvio 18. ASP.NET-mallipohjan valitseminen

6.3 Active Directory -haun toteutus

Opiskelijoiden haku toteutetaan sovelluksen käyttäjän antaman hakusanan perusteella. Haku ottaa yhteyden SeAMKin Active Directory -hakemistopalveluun ja hakee listan opiskelijoista hakusanan perusteella. Sovellus ottaa myös yhteyden SeAMKin ADAM–hakemistopalveluun, josta haetaan tietyt lisätiedot haetuista opiskelijoista, kuten puhelinnumerot GridView-kontrolleriin.

Ohjelmakoodissa tutkitaan ensin, mikä hakukenttä on valittu. Hakukenttien painike-kontrollerit lähettävät kyseisen hakukentän attribuutin ja sisällön hakukoodiin. Hakukentän attribuutin avulla alustetaan hakukoodi käyttämään oikeaa hakukriteeriä. Seuraavassa koodiesimerkissä (esimerkki 1) opiskelijan nimeä varten tarkoitettu hakukenttä lähettää attribuutin ”nimi” hakukoodille, jolloin hakukoodi käyttää attribuuttia ”displayName” hakukriteerinä. ”displayName”-attribuutti on Active Directoryn resurssien attribuutti, jolla tarkoitetaan AD-palvelussa sijaitsevien käyttäjien nimiä.

```
if (attribute.Equals("nimi"))
{
    attributeForQuery = "displayName";
    string[] queryStringToSplit = receivedQueryString.Split(' ');
    queryString = queryStringToSplit[0] + " " + queryStringToSplit[1];
}
if (attribute.Equals("opiskelijaTunnus"))
{
    attributeForQuery = "sAMAccountName";
    queryString = receivedQueryString;
}
if (attribute.Equals("numero"))
{
    attributeForQuery = "mobile";
    queryString = receivedQueryString;
    queryString.Trim();
}
```

Esimerkki 1.

Ensiksi alustetaan tarvittava DataTable-taulukko. Tähän taulukkoon syötetään hakutuloksen tiedot. Taulukkoon lisätään hakua varten tarpeelliset sarakkeet, jotka nimetään Active Directoryn attribuuttien mukaan.

```
DataTable table = new DataTable();
DataRow oneRow;

table.Columns.Add("displayName");
table.Columns.Add("sAMAccountName");
table.Columns.Add("Description");
table.Columns.Add("mobile");
table.Columns.Add("schacDateOfBirth");
table.Columns.Add("userAccountControl");
table.Columns.Add("badPwdCount");
table.Columns.Add("pwdLastSet");
table.Columns.Add("lastLogon");
```

Esimerkki 2.

Seuraavaksi alustetaan tarvittavat DirectoryEntry- ja DirectorySearcher-luokat. DirectoryEntry ottaa yhteyden hakemistopalveluun ja DirectorySearcher suorittaa haun. DirectoryEntry-olioon lisätään AD-polku, joka määrittää hakemistopalvelussa haettavan objektin sijainnin. Polku koostuu IP-osoitteesta ja DC-nimikkeistä, jotka luetaan sovelluksen XML-tiedostosta. Polun lisäämisen jälkeen DirectoryEntry-olio lisätään DirectorySearcher-luokkaan.

```
DirectoryEntry AD;
DirectorySearcher getAllUsers;
DirectoryEntry ADAM;
DirectorySearcher getAllUsersADAM;
```

```
XmlDocument readXML = new XmlDocument();
readXML.Load(HttpContext.Current.Server.MapPath(@"Config.xml"));
XmlNode DC = readXML.SelectSingleNode("Configuration/DC");
XmlNode allOUs = readXML.SelectSingleNode("Configuration/searchOUs");
XmlNode ADRoot = readXML.SelectSingleNode("Configuration/ADRoot");
XmlNode ADAMIP = readXML.SelectSingleNode("Configuration/ADAM");
XmlNode ADAMRoot = readXML.SelectSingleNode("Configuration/ADAMRoot");
string empID = null;

AD = new DirectoryEntry(@"LDAP://" + DC.InnerText.ToString()
    + "/" + ADRoot.InnerText.ToString());
getAllUsers = new DirectorySearcher(AD);
```

Esimerkki 3.

Kun DirectorySearcher-olio on alustettu, määritellään haun kriteerit. Seuraavaksi kutsutaan Filter-metodia, johon lisätään tekstimuodossa hakukriteerit. Koodiesimerkissä 4 hakukriteeriksi valitaan AD-palvelun käyttäjät eli opiskelijat, joiden ”displayName”-attribuutti vastaa hakukentän tekstiä.

```
getAllUsers.Filter = "(& (objectCategory=user)(" + attributeForQuery
    + "=*" + queryString + "*)(!pwdLastSet=0))";
```

Esimerkki 4.

Seuraavaksi määritellään, mitkä attribuutit haetaan hakukriteerin suodattamista käyttäjistä.

```
getAllUsers.PropertiesToLoad.Add("displayName");
getAllUsers.PropertiesToLoad.Add("sAMAccountName");
getAllUsers.PropertiesToLoad.Add("Description");
getAllUsers.PropertiesToLoad.Add("badPwdCount");
getAllUsers.PropertiesToLoad.Add("pwdLastSet");
getAllUsers.PropertiesToLoad.Add("lastLogon");
getAllUsers.PropertiesToLoad.Add("employeeID");
getAllUsers.PropertiesToLoad.Add("userAccountControl");
getAllUsers.PropertiesToLoad.Add("mobile");
getAllUsers.PropertiesToLoad.Add("distinguishedName");
```

Esimerkki 5.

DirectorySearcher-olion FindAll-metodilla haku suoritetaan, ja saadut hakutulokset käydään läpi. Foreach-lause käy läpi jokaisen hakutuloksen ja tarkistaa löytyykö haetut attribuutit. Hakutulokset otetaan Searchresult-oliosta ja lisätään DataTablen riviin. Rivi koostuu esimerkissä 2 alustetuista sarakkeista ja se sisältää yhden opiskelijan tiedot.

```
foreach (SearchResult oneResult in getAllUsers.FindAll())
```

```
                {
oneRow = table.NewRow();

                if (oneResult.Properties.Contains("displayName"))
                {
oneRow["displayName"] = oneResult.Properties["displayName"][0].ToString();
                }
                else oneRow["displayName"] = "Displayname ei löytynyt";
              if (oneResult.Properties.Contains("employeeID"))
                {
                     empID = oneResult.Properties["employeeID"][0].ToString();
                }
```

Esimerkki 6.

Kun Active Directorysta saadut hakutulokset on käyty läpi, suljetaan AD-yhteys ja aloitetaan ADAM-haku. ADAM-palvelusta haetaan opiskelijoiden syntymäajat ja puhelinnumerot. Hakukriteerinä käytetään opiskelijan ”employeeID”-attribuuttia, joka alustettiin esimerkissä 6.

```
getAllUsers.Dispose();
                    AD.Close();


                    ADAM = new DirectoryEntry(@"LDAP://" +
ADAMIP.InnerText.ToString() + "/" + ADAMRoot.InnerText.ToString());

                    getAllUsersADAM = new DirectorySearcher(ADAM);

                    getAllUsersADAM.Filter = ("(uid=" + empID + ")");
                    getAllUsersADAM.SearchScope = SearchScope.Subtree;
                    getAllUsersADAM.PropertiesToLoad.Add("schacDateOfBirth");
                    getAllUsersADAM.PropertiesToLoad.Add("mobile");

                    try
                    {
                      foreach (SearchResult ADAMResult in getAllUsersADAM.FindAll())
                        {
```

Esimerkki 7.

Lopuksi katkaistaan ADAM-yhteys ja lisätään tiedot DataTablen riviin. DataTable siirretään GridView-kontrolleriin.

```
              }
              getAllUsersADAM.Dispose();
                     ADAM.Close();

                     table.Rows.Add(oneRow);
                 }
                 gridViewADAM.DataSource = table;
                 gridViewADAM.DataBind();
```

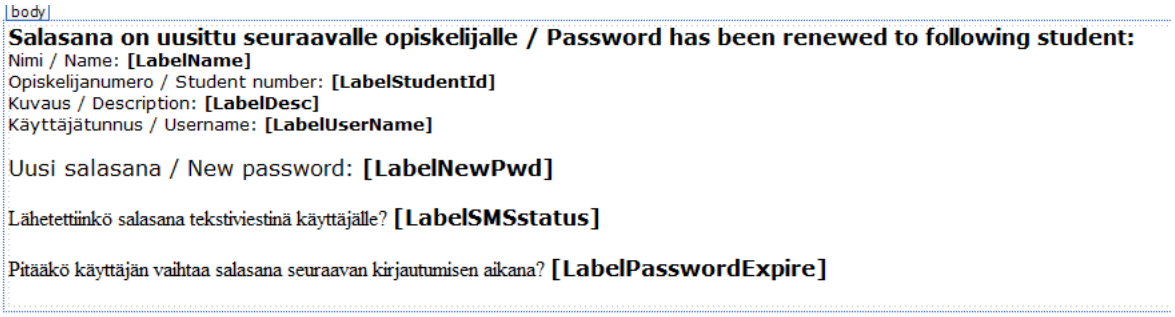
Esimerkki 8.

GridView-kontrolleri esittää haetut tiedot taulukkomuodossa (kuvio 16). Jos tietyn opiskelijan tunnukset ovat vanhentuneet, esitetään kyseisen opiskelijan rivi punaisena.

6.4 Salasanan uusinta

Opiskelijoiden haun jälkeen tiedot esitetään GridView-kontrollerissa. Opiskelijan nimeä painamalla avautuu ikkuna, joka näyttää valitun opiskelijan tiedot ja valinnat salasanan uusintaa varten. Salasanan voi määrittää itse tai luoda satunnaisgeneraattorilla. On myös mahdollista valita vaihtoehto, jossa salasana lähetetään tekstiviestinä kyseiselle opiskelijalle.

Salasanan vaihdossa suoritetaan uusi haku, jolla haetaan kyseisen opiskelijan DN-nimi Active Directorysta. Opiskelijan DN-nimen avulla suoritetaan salasanan vaihto DirectoryEntry-olion avulla. Ohjelmakoodi DN-nimen hausta ja salasanan vaihdosta kuvataan liitteessä 1. Salasanan vaihdon jälkeen sovellus siirtyy salasanan vaihdon tulosten tarkastelusivulle, jossa näytetään opiskelijan tiedot sekä uusi salasana.



Kuvio 19. ASP.NET Designer -sivu salasanan vaihdon tuloksesta

6.5 Ennakoiva haku

Kun kirjoitetaan opiskelijan nimeä hakukenttään, ennakoiva haku suorittaa hakutoimintoa taustasuorituksena ja tuo listan opiskelijoista hakukenttään. Tämän haun avulla voidaan täsmentää haluttu opiskelija hakutuloksista.

Ennakoiva haku käyttää AJAX Control Toolkitin versiota 15.1.1. AJAX Control Toolkit on ASP.NET-sovellukseen lisättävä kirjasto kontrollereita, jotka täydentävät sovelluksen käyttöliittymää (NuGet 2016). Kirjasto sisältää AutoCompleteExtender-kontrollerin, jonka avulla ennakoiva haku mahdollistetaan. Nämä kontrollerit ottavat sovelluksen hakukenttiin kirjoitetut tekstit ja siirtävät ne taustalle suoritettavan hakukoodin käsiteltäviksi. Kuviossa 20 kuvataan AutoCompleteExtender tyylitiedostossa.

```
<body style="height: 615px">
    <form id="form1" runat="server">
         <asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="true" />
        <br />
         <asp:Label ID="Label6" runat="server" style="z-index: 1; left: 581px;
            top: 18px; position: absolute; width: 251px; font-size: xx-large;
            font-weight: 700; text-decoration: underline" Text="Salasanan nollaus"></asp:Label>
         <br />
    <div class="style1" draggable="true">
            <cc1:AutoCompleteExtender ID="AutoCompleteExtenderAD" runat="server" ServiceMethod="GetUserListFromAD"
                        ServicePath="WebService.asmx" TargetControlID="tbNimi">
                    </cc1:AutoCompleteExtender>
        <cc1:AutoCompleteExtender ID="AutoCompleteExtender2" runat="server" ServiceMethod="GetUserListFromADByAccountName"
                        ServicePath="WebService.asmx" TargetControlID="tbKayttajatunnus">
                    </cc1:AutoCompleteExtender>
        <cc1:AutoCompleteExtender ID="AutoCompleteExtender3" runat="server" ServiceMethod="GetUserListFromADByPhoneNumber"
                        ServicePath="WebService.asmx" TargetControlID="tbPuhelinnro">
                    </cc1:AutoCompleteExtender>
            Kirjoita nimi muodossa &quot;Sukunimi, Etunimi&quot; , Kirjoita puhelinnumero yhteen.

        <br />
```

Kuvio 20. AutoCompleteExtender-kontrollerit

Ensiksi kontrolleria käyttävä ikkuna alustetaan ScriptManager–kontrollerilla, joka hallinnoi AJAX–kontrollereiden toimintaa. Seuraavaksi lisätään AutoCompleteExtender–kontrollerit. Kontrollerit lisätään jokaista sovelluksen hakukenttää varten, ja TargetControlID–attribuuttiin kirjoitetaan kyseisen hakukentän tunniste eli ID. ServicePath–attribuutilla täsmennetään kooditiedosto, jossa hakukoodit sijaitsevat. Kontrollerien ServiceMethod–attribuutilla täsmennetään, mikä kooditiedoston metodi suoritetaan kyseisen hakukentän perusteella.

Taustalla suoritettava hakukoodi sijaitsee tiedostossa WebService.asmx. Hakukoodit ovat metodeja, jotka suorittavat haun sovelluksen hakukentästä saadun tekstin perusteella. Haku suoritetaan heti, kun hakukenttään kirjoitetaan tekstiä. Haetut opiskelijat sijoitetaan Array–taulukkoon, joka lähetetään takaisin sovellukselle. Ennakoivan haun nimen perusteella suoritettava koodi on kuvattuna liitteessä 2.

6.6 Salasanan lähetys tekstiviestinä

Salasanan vaihdon yhteydessä voi valita vaihtoehdon, jolla uusi salasana lähetetään opiskelijalle tekstiviestinä. Tekstiviestin lähettää Jelpparin oma palvelin, johon on ohjelmoitu Jelpparin toimesta tekstiviestejä lähettävä sovellus. Tähän palvelimeen otetaan SQL-yhteys, jonka jälkeen luodaan SQL-komentona

tekstiviestinä lähetettävä viesti. Koodi SQL-komennon luomiseen kuvataan liitteessä 3.

7 YHTEENVETO JA POHDINTA

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa ASP.NET-sovellus, joka pystyy hakemaan ja uusimaan SeAMKin opiskelijoiden salasanoja. Työssä tarkasteltiin versionhallintaa ja sen käyttöä osana sovelluksen kehitystä. Osana web-sovellusten kehitystä kerrottiin myös tietoturvasta.

Työn alussa käsiteltiin hakemistopalvelujen historia ja AD-palvelun rakenne. Työssä kerrottiin myös ASP.NET-sovelluskehyksen historiasta ja sen toimintamalleista. Työssä perehdyttiin myös versionhallintaan, kuten Git- ja TortoiseSVN-versionhallintaohjelmiin. Lisäksi käytiin läpi eri autentikointi- ja autorisointi-tapoja sovellusten kehityksessä, kuten Windows- ja Forms-autentikoinnit. Työn lopuksi kehitettiin sovellus, joka käyttää ASP.NET-sovelluskehystä.

Tämän työn tekijällä oli vain vähän kokemusta ASP.NET-sovellusten kehityksestä, joten haasteita oli monia. Testaamisessa huomattiin jälkikäteen, että tietyt toiminnot eivät onnistuneet paikallisessa ympäristössä, vaan sovellus piti siirtää SeAMKin tietohallinnon palvelimelle. Näiden toimintojen testaus jäi vähälle, sillä testaus piti ulkoistaa tietohallinnon työntekijöille, joilla oli kiireitä muiden projektien kanssa. Sovelluksesta saatiin kuitenkin hyvä mallipohja jatkokehitystä varten.

Loppujen lopuksi työstä sai todella paljon kokemusta hakemistopalveluista, versionhallinnasta ja ASP.NET-sovellusten kehittämisestä. Työssä saatiin toteutettua web-sovellus, joka hakee SeAMKin opiskelijoiden tiedot hakusanan perusteella Active Directorystä.

ASP.NET-sovelluksilla voidaan luoda ohjelmia jotka suorittavat monimutkaisia toimintoja helpolla käyttöliittymällä. Tämän ansiosta sovelluksen käyttäjiltä ei vaadita erityistä tietoteknistä osaamista. ASP.NET-sovellukset ovat edelleen kehityksessä, joten näiden sovellusten suosio varmasti kasvaa.

LÄHTEET

Apple. 2012. Authentication and Authorization. [www-dokumentti]. Apple Inc. [Viitattu 13.10.2016]. Saatavissa: https://developer.apple.com/library/prerelease/content/documentation/Security/Conceptual/Security_Overview/AuthenticationAndAuthorization/AuthenticationAndAuthorization.html#//apple_ref/doc/uid/TP30000976-CH2-SW1

Chacon, S. & Straub, B. 2009. Getting Started – About Version Control. [Verkkokirja]. Software Freedom Conservancy. [Viitattu 9.10.2016]. Saatavissa: https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control

Evjen, B., Hanselman, S., Muhammad, F., Sivakumar, S. & Rader, D. 2006. Professional ASP.NET 2.0. Indianapolis: Wiley Publishing, Inc.

FitzMacken, T. 2014. Introducing ASP.NET Web Pages - Getting Started. [www-dokumentti]. Microsoft Corporation. [Viitattu 1.11.2016]. Saatavissa: https://www.asp.net/web-pages/overview/getting-started/introducing-aspnet-web-pages-2/getting-started

GNU. 1991. GNU General Public License. [www-dokumentti]. Free Software Foundation. [Viitattu 20.3.2016]. Saatavissa: http://www.gnu.org/licenses/gpl-2.0.html#SEC1

Jelppari. Ei päiväystä. Jelppari. [www-dokumentti]. Seinäjoen koulutuskuntayhtymän ja Seinäjoen ammattikorkeakoulun Tietohallinnon Helpdesk-palvelu Jelppari. [Viitattu 18.5.2016]. Saatavissa: http://jelppari.epedu.fi/fi/Etusivu

Küng, S., Onken, L. & Large, S. Ei päiväystä. What is TortoiseSVN? [www-dokumentti]. The TortoiseSVN Team. [Viitattu 19.3.2016]. Saatavissa: https://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-preface.html#tsvn-preface-about

Küng, S., Onken, L. & Large, S. Ei päiväystä. What is TortoiseSVN? [www-dokumentti]. The TortoiseSVN Team. [Viitattu 21.3.2016]. Saatavissa: https://tortoisesvn.net/docs/release/TortoiseSVN_en/tsvn-preface-features.html Haettu 20.5.2016

Microsoft. 2015. Visual Studio 2015 Update 1. [www-dokumentti]. Microsoft Corporation. [Viitattu 13.10.2016]. Saatavissa: https://www.visualstudio.com/en-us/news/releasenotes/vs2015-update1-vs#dotnet

Microsoft. 2000. Windows 2000 Security Technical Overview. [www-dokumentti]. Microsoft Corporation. [Viitattu 29.10.2016]. Saatavissa: https://msdn.microsoft.com/en-us/library/bb742513.aspx

Microsoft ASP.NET Team. 2009. ASP.NET MVC Overview. [www-dokumentti]. Microsoft Corporation. [Viitattu 25.10.2016]. Saatavissa: https://www.asp.net/mvc/overview/older-versions-1/overview/asp-net-mvc-overview

Microsoft. Ei päiväystä. ASP.NET. [www-dokumentti]. Microsoft Corporation. [Viitattu 12.10.2016]. Saatavissa: https://www.asp.net/aspnet

Microsoft. Ei päiväystä. ASP.NET Web Server Controls Overview. [www-dokumentti]. Microsoft Corporation. [Viitattu 7.11.2016]. Saatavissa: https://msdn.microsoft.com/en-us/library/zsyt68f1.aspx

Microsoft. Ei päiväystä. ASP.NET Validation Controls. [www-dokumentti]. Microsoft Corporation. [Viitattu 8.11.2016]. Saatavissa: https://msdn.microsoft.com/en-us/library/debza5t0.aspx

Microsoft. Ei päiväystä. ASP.NET User Controls. [www-dokumentti]. Microsoft Corporation. [Viitattu 9.11.2016]. Saatavissa: https://msdn.microsoft.com/en-us/library/y6wb1a0e.aspx

Microsoft. Ei päiväystä. Gridview control. [www-dokumentti]. Microsoft Corporation. [Viitattu 15.11.2016]. Saatavissa: https://msdn.microsoft.com/en-us/library/cc295223.aspx

Microsoft. Ei päiväystä. Introduction to ASP.NET Web Forms. [www-dokumentti]. Microsoft Corporation. [Viitattu 22.11.2016]. Saatavissa: https://www.asp.net/web-forms/what-is-web-forms

Microsoft. Ei päiväystä. Windows Authentication <windowsAuthentication>. [www-dokumentti]. Microsoft Corporation. [Viitattu: 28.9.2016]. Saatavissa: http://www.iis.net/configreference/system.webserver/security/authentication/windowsauthentication

Mitchell, S. 2008. Security Basics and ASP.NET Support (C#). [www-dokumentti]. Microsoft Corporation. [Viitattu: 29.9.2016] Saatavissa: http://www.asp.net/web-forms/overview/older-versions-security/introduction/security-basics-and-asp-net-support-cs

NuGet. 2016. [www-dokumentti]. .NET Foundation. [Viitattu 14.11.2016]. Saatavissa: https://www.nuget.org/packages/AjaxControlToolkit/

Ravishankar, S. 2013. Git: Version Control for Everyone Beginner's Guide. [Verkkokirja]. Birmingham: Packt Publishing Ltd. [Viitattu 8.6.2016] Saatavana ProQuest ebrary e-kirjakokoelmasta.

Richards, J., Allen, R. & Lowe-Norris, A.G. 2006. Active Directory, Third Edition. 3.painos. California: O'Reilly Media, Inc.

Talvivaara, J. Ei päiväystä. Verkon nimi- ja hakemistopalvelut 2. Nimipalvelut. [www-dokumentti]. VirtuaaliAMK-verkosto. [Viitattu 13.5.2016]. Saatavissa: http://www2.amk.fi/mater/tietotekniikka/nimipalvelut/2_nimipalvelut.html

Talvivaara, J. Ei päiväystä. Verkon nimi- ja hakemistopalvelut 3. Hakemistopalvelut. [www-dokumentti]. VirtuaaliAMK-verkosto. [Viitattu 14.5.2016]. Saatavissa: http://www2.amk.fi/mater/tietotekniikka/nimipalvelut/3_hakemistopalvelut.html

Talvivaara, J. Ei päiväystä. Verkon nimi- ja hakemistopalvelut 8. Active Directory. [www-dokumentti]. VirtuaaliAMK-verkosto. [Viitattu 15.5.2016]. Saatavissa: http://www2.amk.fi/mater/tietotekniikka/nimipalvelut/8_activedirectory.html

Technet Microsoft. 2014. What Is Active Directory Application Mode? [www-dokumentti]. Microsoft Corporation. [Viitattu 27.10.2016]. Saatavissa: https://technet.microsoft.com/en-us/library/cc738377(v=ws.10).aspx

The TortoiseSVN Team. 2016. TortoiseSVN Screenshots. [www-dokumentti]. The TortoiseSVN Team. [Viitattu 18.3.2016]. Saatavissa: https://tortoisesvn.net/screenshots.html

The TortoiseSVN Team. Ei päiväystä. Project Status – Older releases. [www-dokumentti]. The TortoiseSVN Team. [Viitattu: 11.11.2016]. Saatavissa: https://tortoisesvn.net/status.html

Wahl, M., Kille, S. & Howes, T. 1997. Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names. [www-dokumentti]. Netscape Communications Corp.[Viitattu 29.10.2016]. Saatavissa: https://www.ietf.org/rfc/rfc2253.txt

LIITTEET

LIITE 1. Koodi opiskelijan DN-nimen hausta ja salasanan vaihdosta

```
    public enum objectClass
    {
        user, group, computer
    }
    public enum returnType
    {
        distinguishedName, ObjectGUID
    }

    public string GetObjectDistinguishedName(string usersAMAccountName)
    {
        XmlDocument readXML = new XmlDocument();
        readXML.Load(HttpContext.Current.Server.MapPath(@"Config.xml"));
        XmlNode DC = readXML.SelectSingleNode("Configuration/DC");
        XmlNode ADRoot = readXML.SelectSingleNode("Configuration/ADRoot");

        string distinguishedName = string.Empty;
        string connectionPrefix = "LDAP://" + DC.InnerText.ToString() + "/" +
ADRoot.InnerText.ToString();
        DirectoryEntry entry = new DirectoryEntry(connectionPrefix);
        DirectorySearcher mySearcher = new DirectorySearcher(entry);

        mySearcher.Filter = "(&(objectClass=user)(| (cn = " + usersAMAccountName +
")(sAMAccountName = " + usersAMAccountName + ")))";

        SearchResult result = mySearcher.FindOne();

        if (result == null)
        {
            throw new NullReferenceException
            ("unable to locate the distinguishedName for the user " +
            usersAMAccountName + " in the " + DC.InnerText.ToString() + "/" +
ADRoot.InnerText.ToString() + " domain");
        }
        DirectoryEntry directoryObject = result.GetDirectoryEntry();

        distinguishedName = "LDAP://" + directoryObject.Properties["distin-
guishedName"].Value;

        entry.Close();
        entry.Dispose();
        mySearcher.Dispose();
        return distinguishedName;
    }


    public void ResetPassword(string userDistinguishedName, string password)
    {
        DirectoryEntry uEntry = new DirectoryEntry(userDistinguishedName);
        uEntry.Invoke("SetPassword", new object[] { password });
        uEntry.Properties["LockOutTime"].Value = 0;

        uEntry.Close();
    }
```

LIITE 2. Ennakoivan haun metodi

```
[WebMethod]
    public string[] GetUserListFromAD(string prefixText)
    {
        ArrayList allUsers = new ArrayList();
        DirectoryEntry AD;
        DirectorySearcher getAllUsers;

        XmlDocument readXML = new XmlDocument();
        readXML.Load(HttpContext.Current.Server.MapPath(@"Config.xml"));
        XmlNode DC = readXML.SelectSingleNode("Configuration/DC");
        XmlNode allOUs = readXML.SelectSingleNode("Configuration/searchOUs");

        bool amountOfWords = prefixText.ToLowerInvariant().Contains(',');

        string searchFilter = "displayName";

        foreach (XmlNode oneOU in allOUs)
        {
            AD = new DirectoryEntry(@"LDAP://" + DC.InnerText.ToString() + "/" +
oneOU.InnerText.ToString());
            getAllUsers = new DirectorySearcher(AD);

            getAllUsers.Filter = "(& (objectCategory=user)(" + searchFilter + "=*" +
prefixText + "*)(userAccountControl=512)(!pwdLastSet=0))";
            getAllUsers.SearchScope = SearchScope.Subtree;
            getAllUsers.PropertiesToLoad.Add("displayName");
            getAllUsers.PropertiesToLoad.Add("Description");

            try
            {
                foreach (SearchResult oneResult in getAllUsers.FindAll())
                {
                    string userToAdd;

                    if (!allUsers.Contains(oneResult.Properties[searchFil-
ter][0].ToString()))
                    {
                        userToAdd = oneResult.Properties[searchFil-
ter][0].ToString();

                        if (oneResult.Properties.Contains("Description"))
                        {
                            userToAdd += " " + oneResult.Properties["Descrip-
tion"][0].ToString();
                        }

                        allUsers.Add(userToAdd);
                    }
                }
                getAllUsers.Dispose();
                AD.Close();
            }
            catch (DirectoryServicesCOMException e)
            {
```

```
            getAllUsers.Dispose();
            AD.Close();
        }
    }

    allUsers.Sort();

    string[] array = new string[allUsers.Count];
    allUsers.CopyTo(array);

    return array;
```

LIITE 3. Koodi tekstiviestin lähetyksestä SQL-komentona

```
public bool SendSMSwithPwd(string passwordReceived, bool isCheckBoxChecked, bool is-
CheckBoxChangePwdChecked)
    {
                // Viesti tekstiviestiin
                string messageToEncode = "Hei! Salasanasi on vaihdettu! Uusi salasa-
nasi on: " + passwordReceived
             + " Terveisin Jelppari!";
                String s_url_encoded = HttpUtility.UrlEncode(messageToEncode);

                XmlDocument readXMLSMS = new XmlDocument();
                readXMLSMS.Load(HttpContext.Current.Server.MapPath(@"SMScon-
fig.xml"));

                XmlNode _sqlsrv = readXMLSMS.SelectSingleNode("SendSMS/SMSGate-
waySrv");
                XmlNode _sqldb = readXMLSMS.SelectSingleNode("SendSMS/SMSGate-
wayDB");
                XmlNode _email = readXMLSMS.SelectSingle-
Node("SendSMS/emailSenderAddress");
                XmlNode _smtpsrv = readXMLSMS.SelectSingleNode("SendSMS/smtpSrv");
                XmlNode _logfile = readXMLSMS.SelectSingleNode("SendSMS/logFile");
                XmlNode _max = readXMLSMS.SelectSingleNode("SendSMS/maxMessages");

                string userRealName = null;
                string lastSqlIdentity = null;

                SqlConnection conn = new SqlConnection("Data Source=" + _sqlsrv.In-
nerText.ToString()
+ ";Initial Catalog=" + _sqldb.InnerText.ToString() + ";Integrated Security=SSPI;");
                conn.Open();

                SqlCommand command;
                SqlDataReader ident;

                userRealName = getAttributeTest(User.Identity.Name.ToString().Sub-
string(User.Identity.Name.ToString().LastIndexOf(@"\") + 1), "cn");

                string numberMobile = Session["mobile"].ToString();

                if (IsValidPhone(numberMobile))
                {
                    // First, insert new outgoing entry
                    command = new SqlCommand("INSERT INTO Outgoing (Number, Message)
VALUES ('" + numberMobile + "', '" + s_url_encoded + "')", conn);
                    command.ExecuteNonQuery();
                    command.Dispose();

// Second get inserted identity (SELECT SCOPE_IDENTITY())
                    command = new SqlCommand("SELECT SCOPE_IDENTITY() FROM Out-
going", conn);
                    ident = command.ExecuteReader();

                    while (ident.Read())
                    {
                        lastSqlIdentity = ident[0].ToString();
                    }
                    ident.Dispose();
```

```
                    command.Dispose();

                    // Third, make log about send
                    command = new SqlCommand("INSERT INTO OutgoingLogs (senderID,
senderName, DateTime, Number, Message, OutgoingId, Successful) VALUES ('"
                        + User.Identity.Name.ToString() + "', '" + userRealName +
"', GetDate(), '" + numberMobile + "', '" + s_url_encoded + "', " + lastSqlIdentity
+ ", 0)", conn);
                    command.ExecuteNonQuery();
                    command.Dispose();

                    //Lokiviesti omaan tekstitiedostoon:
                    createLogMessage(Session["userSAMAccountName"].ToString(), is-
CheckBoxChecked, isCheckBoxChangePwdChecked);

                    searchByName = false;
                    searchByID = false;
                    searchByPhone = false;

                    conn.Close();

                    Session.Add("SMSstatus", "Kyllä");
                    return true;
                }
                else
                {
                    //Numeroja ei ole tai ne eivät toimi = Ei lähetetä tekstiviestiä
                    Session.Add("SMSstatus", "Epäonnistui - Puhelinnumeroa ei
saatu.");
                    searchByName = false;
                    searchByID = false;
                    searchByPhone = false;
                    conn.Close();
                    return false;
                }
```