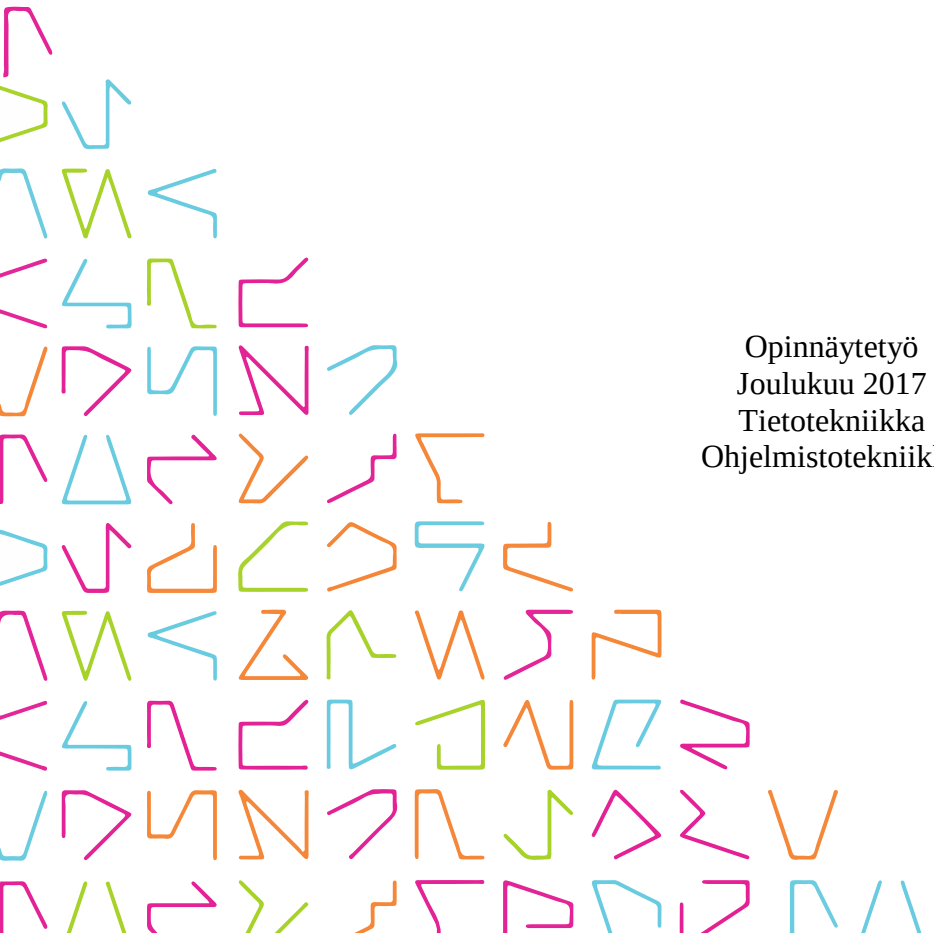
TAMPEREEN
AMMATTIKORKEAKOULU

MultiSender -mobiilisovellus

Mikko Makkonen



Opinnäytetyö
Joulukuu 2017
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

Makkonen Mikko:
MultiSender -mobiilisovellus

Opinnäytetyö 17 sivua
Joulukuu 2017

Tämä opinnäytetyö sisältää mobiiliohjelman toteuttamisen, jossa selitetty mitä funktioita on käytetty ja mitä ne tekevät. Raportti sisältää myös vähän yleistä tietoa käytetyistä kielistä ja ohjelmista.

Asiasanat: ohjemointi, mobiili

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Tietotekniikka
Ohjelmistotekniikka

MIKKO MAKKONEN:
MultiSender Mobile Application

Bachelor's thesis 17 pages
December 2017

This thesis report includes mobile application description and its functions explained. The report also includes information about used programs and coding languages.

Key words: programming, mobile

SISÄLLYS

LYHENTEET JA TERMIT

HTML	Hypertext Markup Language
JS	JavaScript
AJAX	Asynchronous JavaScript and XML
XML	Extensible Markup Language

1 JOHDANTO

Työn tarkoituksena oli luoda mobiilisovellus, jolla voi lukea puhelinluetteloa ja lähettää haettua tietoa serveriin. Puhelimesta saatu tieto voidaan lukea ja käyttää esimerkiksi asiakkaiden yhteenotossa. Ohjelman lopputarkoitus on antaa yrityksille helppotapa ottaa numeroita ja tietoja puhelimesta jota käyttää esimerkiksi markkinoinnissa.

2 OHJELMAT, LIITÄNNÄISET JA KIELET

Tässä osiossa mainitaan yleisimmät ohjelmat ja ohjelmointikielet mitä projektissa tuli käytettyä. Osioissa kerrotaan yleisesti mitä ne tekevät ja hieman mihin sitä on käytetty projektissa.

2.1 Visual Studio

Visual studio on ilmaiseksi käytettävä ohjelmointiympäristö, joka sisältää peruskirjastot valmiiksi monille kielille. Visual Studiolla voidaan luoda ohjelmia käyttöjärjestelmille, selaimille ja mobiilialustoille. Ohjelmointikielistä löytyvät mm. C, C++, Java, HTML/Javascript, Visual Basic, F#, C#, Node.js ja Python. Ilmaisessa versiossa on puutteita verrattuna maksulliseen versioon. Siitä puuttuvat esimerkiksi arkkitehtuuri-ohjelmia, testaustyökaluja ja jotkin koodin kääntö työkalut. Visual studio oli pää ohjelma projektin tekemiseen.

2.2 Apache Cordova

Apache Cordova on Visual Studion tuettu liitännäinen, jolla voidaan luoda mobiilisovelluksia kaikille alustoille käyttäen HTML/Javascript/CSS -koodikieliä. Cordova on avoin koodikieli ja sillä ilmainen. Tämän erillaisen koodauksen takia mikään ohjelma ei ole natiivi kyseiselle alustalle vaan hybridi. Cordovaa käytettiin projektissa sen joustavuuden takia monessakäyttöliittymässä yhtäaikaa.

2.3 JQuery

JQuery on nopea, pieni ja kattava Javascript -kirjasto. Se tuo HTML kieleen esimerkiksi dokumentin manipulointia, tapahtumanhallinnan, animaatioita sekä Ajaxin. Ajax tuo helppokäyttöisen APIn joka toimii monissa selaimissa. Ajaxin kautta saadaan helppo yhteys serveriin ja takaisin puhelimeen. JQueryä käytettiin Ajaxin helpon yhteyksien takia.

2.4 JavaScript ja HTML

JavaScript on yksi suuremmin käytetyistä verkkosivujen toteutuskielistä, jolla saa sujuvia ja hyvän näköisiä verkkosivuja. Koska kyseinen ohjelma on verkkosivupohjainen, käytetään sitä kyseisessä ohjelmassa.

HTML on alkuperäinen verkkosivujen tekemiseen tarkoitettu kieli. Se on vanhempi kuin JavaScript ja yleisesti kaikkien verkkosivujen pohja mihin lisätään muita kieliä.

3 KOODAAMINEN

Kaikki koodaaminen tapahtuu HTML ja JavaScriptiä käyttäen. Ohjelman käyttöliittymä on tehty HTML kieltä käyttäen. JavaScriptiä käytetään sisäiseen koodiin, jolla haetaan ja lähetetään tietoa puhelimesta.

Koodiesimerkeissä näkyy viittauksina mitä mikäkin funktio tekee koodissa. Toistuvissa koodin pätkissä kommentit on otettu pois. Kommenteista saadaan helposti selville mitä mikäkin kohta koodissa on tarkoitus tehdä. Yleisesti kommentteja laitetaan ainakin funktioiden alkuun jossa tulee selväksi mitä se tekee. Kommenttien merkitys koodissa ei ole suuri, mutta se auttaa muita jotka lukevat samaa koodia ymmärtämään mitä tietyt osiot tekevät. Näin ei tarvitse kysellä alkuperäiseltä tekijältä tai alkaa itse tutkimaan ja tuhlaamaan aikaa.

Monissa koodinpätkissä näkyy huomattavasti samaa koodia, varsinkin Ajax kutsuissa. Koska Ajax kutsut on Jqueryn valmiita funktioita ei tarvitse muuttaa kuin komentoa millä sitä kutsutaan. Tietysti kaikkien kutsujen tulos on erinlainen mitä saadaan vastauksena onnistumisesta.

3.1 Login

Kirjautuminen tapahtuu käyttäjätunnuksella ja salasanalla, jotka haetaan käyttöliittymän kahdesta tesktilaatikosta. Käyttäen Ajax POST komentoa lähetetään käyttäjätunnus ja salasana serverille Json -muodossa. Vastauksena saadaan tieto, onnistuiko kirjautuminen ja henkilökohtainen token, joka säilytetään muistissa kirjautumisen ajan.

Kirjautuessa käyttäjätunnus ja salasana muutetaan Json muotoon, jota vastaanottava laite lukee. Tämän jälkeen saadaan vastauksena, onnistuiko kirjautuminen vai ei. Jos kirjautuminen onnistuu, saadaan dataa takaisin joka tallennetaan loginid nimiseen muuttujaan. Tätä tunnusta käytettään muissa kutsuissa henkilökohtaisena tunnuksena. Tunnuksen avulla ei etsitä, haeta tai muuteta väärän henkilön tietoja.

Tärkeää kirjautumisessa ja muissa funktioissa mitä käydään läpi on nappien moninkertainen painallus. Alla olevassa koodissa on kommentoitu pätkä koodia jolla poistetaan napin ominaisuus väliaikaiseksi, jotta sitä ei voitaisi painaa useampaa kertaa ja tuoda

ongelmia. Ongelmana esimerkiksi moninkertainen painallus on datan ylikuormitus ja vastaan otettavan datan käyttö. Sillä sisään kirjautumista ei ole tarkoitettu käytettäväksi useamman kerran saman henkilön.

```
//Sisään kirjautumisen functio
function handleLogin() {
//otetaan nappi pois käytöstä kun sitä on kerran painettu ja estetään mominker-
tainen painallus
        $("#submitButton").attr("disabled", "disabled");
//tekstikentistä haettu käyttäjätunnus ja salasana
var u = $("#username").val();
var p = $("#password").val();

//lähetettävä käyttäjä tunnus ja salasana yhdistetty yhteen
var sendData = {
"email": u,
"password": p
};

//Tiedon lähetys
        $.ajax({
//ajax funktio millä tunnustetaan mitä function pitää tehdä
            type: "POST",
//osoite mihin tieto lähetetään
url: "http://api.malli.net/login",
//tieto muutetaan suoraviivaiseksi tekstiksi jsonista
            data: JSON.stringify(sendData),
//tiedon tyyppi
            contentType: "application/json",
            Accept: "application/json",
            dataType: "json",
//onnistunut kirjautuminen
            success: function (data, status) {
                alert("Status: " + JSON.stringify(status));
loginid = JSON.stringify(data);
            },
//vika ilmoitus
error: function (status) {
                alert("fail: " + JSON.stringify(status));
}
        });

//Login napin käyttö takaisin
        $("#submitButton").removeAttr("disabled");

returnfalse;
    }
```

3.2 Puhelinluottelon haku

Puhelinluettelosta haetaan tiedot ContactFindOptions() -functiolla. Lisäominaisuutena voidaan hakea kaikki nimet luettelosta tai vain ne jota halutaan. Lisäksi voidaan muokata mitä tietoja näytetään haun jälkeen. PickContact() -listaa kaikki nimet luettelosta, josta voidaan valita yksi jonka tiedot tallennetaan savedContact muuttujaan. Tätä muuttujaa käytettään tiedon lähettämiseen. Onnistuneessa haussa käytetään onSuccess() -

funktiota, joka esittää haetun tuloksen ruudulle. Esimerkissä ruudulle tulee näkyviin sukunimi, etunimi ja tunnus.

```
var savedContact;

//Etsii halutulla nimella tai numerolla puhelinluettelon
function handlefind() {
//hakee tekstilaatikosta kirjotetun tekstin
var name = document.getElementById("name").value;
var options = new ContactFindOptions();
//käyttää kirjoitettua nimen etsinässä. Tyhjänä hakee kaikki tiedot
        options.filter = name;
//hyväksyy monta saman nimistä henkilöä
        options.multiple = true;
//mitä tietoja haetaan
var fields = ["displayName", "name", "organizations"];
//ilmoittaa haun tuloksen
        navigator.contacts.find(fields, onSuccess, onError, options);
};

//Hakee kaikki tiedot ja listaa ne. näistä valitaan yksi joka tallennetaan
function saveContact() {
        navigator.contacts.pickContact(function (contact) {
//tallentaa valitun henkilön tiedot
            savedContact = contact
alert('tallennettu:' + JSON.stringify(savedContact));
        }, function (err) {
            alert('Error: ' + err);
        });
    }


function onSuccess(contacts) {

for (var i = 0; i < contacts.length; i++) {
            alert(
"Family Name: " + contacts[i].name.familyName + "\n" +
"Given Name: " + contacts[i].name.givenName + "\n" +
"Id: " + contacts[i].name.id + "\n"
);

        }

    };
```

3.3 Tietojen haku serveriltä

Tietojen hakeminen tapahtuu Ajax GET -functiolla. Vastauksena saadaan haettujen henkilöiden tiedot Json -muodossa. Tämä tieto tallennetaan getdata -muuttujaan mistä tieto on saatavilla käyttäjälle. Jos halutaan hakea vain yhden henkilön tiedot. Uutena muuttujana käytetään tokenia, joka on aikaisemmin tallennettu loginid muuttuja sisään-kirjautumisesta. Tällä tunnuksella varmistetaan että haettu tieto kuuluu sinulle eikä jol-lekkin muulle.

```
var getdata;
//hakee serveriltä tallennetut tiedot
function handleget() {
        $("#getButton").attr("disabled", "disabled");
var token = JSON.parse(loginid);

        $.ajax({
            type: "GET",
            url: "http://api.malli.net/accounts/",
            contentType: "application/json",
            Accept: "application/json",
            dataType: "json",
            headers: {
"X-Access-Token": token
            },
            success: function (data, status) {
alert("Status: " + JSON.stringify(status));
                getdata = data;            },

            error: function (status) {
// error handler
                alert("error: " + JSON.stringify(status));
            }
        });
        $("#getButton").removeAttr("disabled");
returnfalse;
}
```

3.4 Tietojen Lisääminen, muuttaminen ja poistaminen

Tietojen lisääminen tapahtuu json -muodossa. Lähetyskutsuna käytetään Ajax POST -funktiota. Tiedot otetaan käyttöliittymän tekstilaatikoista. Tiedot lähetetään yksi kerrallaan. Alemmassa koodissa on malliesimerkki, miltä täytetty henkilötieto näyttäisi. Kaikkia tietoja ei ole pakko täyttää lähetyksessä. Kun lähetys onnistuu, saadaan vastauksena tieto tapahtumasta. sendData -muuttujan sisältö voidaan korvata myös handlefind() -funktion tuloksella savedContact -muuttujalla.

```
//lisää serverille tietoa
function handlepost() {
        $("#postButton").attr("disabled", "disabled");

var sendData = savedContact;
//esimerkki lähetettävästä tiedosta
var sendData = {
"firstName": "Tom",
"lastName": "Tester",
"nickName": "Awesome-Tom",
```

```
"phone": "+35840123132",
"email": "tom.tester@dummy.org",
"address": "Street 12 A 456",
"zip": "00010",
"city": "Helsinki",
"companyName": "Acme Ltd.",
"website": "http:\/\/dummy.org",
"note": "This is created at API TESTS"
        };

var token = JSON.parse(loginid);


        $.ajax({
            type: "POST",
            url: "http://api.malli.net/accounts",
            data: JSON.stringify(sendData),
            contentType: "application/json",
            Accept: "application/json",
            dataType: "json",
            headers: {
"X-Access-Token": token
            },
            success: function (data, status) {

                alert("Status: " + JSON.stringify(status));
            },

            error: function (status) {
// error handler
                alert("fail: " + JSON.stringify(status));
            }
        });

        $("#postButton").removeAttr("disabled");
returnfalse;
}
```

Tietojen muuttaminen serverissä tapahtuu samalla tavalla kuin sinne lisääminen. Lähetyskutsuna käytetään Ajax PUT -metodia. Muuttaessa lähetetään kaikki tiedot uudestaan ja yliajetaan vanha. Tiedot lähetetään Json -muodossa. Jotta oikean henkilöntietoja muutetaan, käytetään henkilötiedon serveritunnusta. Kyseisen tunnuksen näkee, kun hakee serveriltä tiedon henkilöstä. Tässäkin voidaan käyttää savedContact -tallennettua muuttujaa, jos haluaa päivittää tietoja. Palautteena saadaan tieto, menikö tiedot perille. Tämän jälkeen voidaan vielä varmistaa contactSearch() funktiolla menikö tiedot varmasti perille.

```
//päivitä tietoja
function handleupdate() {
        $("#updateButton").attr("disabled", "disabled");

var sendData = savedContact;
//esimerkki
var sendData = {
```

```
"firstName": "Jerry",
"lastName": "Tester",
"nickName": "Awesome-Tom",
"phone": "+35840123132",
"email": "Jerry.tester@dummy.org",
"address": "Kultalampi 8",
"zip": "33710",
"city": "Tampere",
"companyName": "Acme Ltd.",
"website": "http:\/\/dummy.org",
"note": "This is created at API TESTS"
        };

var token = JSON.parse(loginid);

$.ajax({
            type: "PUT",
//esimerkki haku osoitteesta, jossa on päivitettävän henkilön tunnus osoitteen
lopussa
url: "http://api.malli.net/accounts/2202",
            data: JSON.stringify(sendData),
            contentType: "application/json",
            Accept: "application/json",
            dataType: "json",
            headers: {
"X-Access-Token": token
            },
            success: function (data, status) {

                alert("Status: " + JSON.stringify(status)
            },

            error: function (status) {
// error handler
                alert("fail: " + JSON.stringify(status));
            }
        });

        $("#updateButton").removeAttr("disabled");

returnfalse;
}
```

Henkilön poistaminen tapahtuu tunnuksen avulla, joka lisätään osoitteeseen samalla tavalla kuin tietojen päivityksessä. Lähetys kutsuna käytetään Ajax DELETE. Vastauksena saadaan, että kyseinen henkilö on poistettu.

```
//poistaa henkilöitä
function handledelete() {
        $("#deleteButton").attr("disabled", "disabled");

var token = JSON.parse(loginid);

        $.ajax({
            type: "DELETE",
            url: "http://api.malli.net/accounts/2203",
```

```
            contentType: "application/json",
            Accept: "application/json",
            dataType: "json",
            headers: {
"X-Access-Token": token
            },
            success: function (data, status) {

                alert("Status: " + JSON.stringify(status));
            },

            error: function (status) {
// error handler
                alert("fail: " + JSON.stringify(status));
            }
        });

        $("#deleteButton").removeAttr("disabled");

returnfalse;
}
```

4 POHDINTA

Opin uusia asioita, kuten HTML ja Javascript -kielet, joita en ollut ennen käyttänyt perusasioita lukuun ottamatta. Suurimpana ongelmana oli laiskuus käynnistää Visual Studio ja alkaa koodaamaan, jonka vuoksi projekti venyi huomattavasti. Otin lopuksi itseäni niskasta kiinni ja kävin koko projektin uudestaan lävitse ja sain aikaiseksi jotain hyväksyttävää. Tulevaisuutta varten pitää ottaa kantaa siihen, että etätyö on tällä hetkellä erittäin vaikeaa koska eksyn muihin asioihin liian helposti. Toisena asiana on aikarajat jotka olisi pakollisia, jotta olisi suurempi paine saada asiat valmiiksi oikeaan aikaan eikä kaksi vuotta myöhemmin.

LÄHTEET

Apache Cordova -dokumentti
https://cordova.apache.org

HTML, Javascript -malleja
www.w3schools.com

Yleistä tietoa ohjelmistoista ja kielistä
en.wikipedia.org