

Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

## **Google Contact API**

Seminarski rad iz Elektronskog poslovanja

**Mentor:**

Vuk Stefanović

**Studenti:**

Mirjana Rističević 0031/2017

Ivana Stanković 0061/2017

Julija Prelević 0136/2017

Beograd, 2020.

## SADRŽAJ

1. UVOD .....	3
2. Pregled korišćene literature .....	3
1. Šta je Google Contacts API? .....	3
2. Kako funkcioniše Google Contacts API .....	4
3. Najčešća primena Google API-a .....	4
4. XAMPP .....	6
3. Kreiranje aplikacije <i>My Contacts</i> .....	7
4. Zaključak .....	19
5. Literatura .....	20

## 1. UVOD

Google API je set API-a koji je Google razvio kako bi omogućio svojim korisnicima da na ovaj način povezuju svoje usluge i pojednostave korišćenje najrazličitijih pogodnosti koje Google servisi pružaju njihovim korisnicima. Osnovni primeri Google API-a su Google Maps, Google Contacts, Google Calendar, itd. Korišćenje API-a je omogućeno u najrazličitijim programskim jezicima i platformama, kao što su Java, JavaScript, Ruby, .NET, PHP, Python, itd.

S obzirom da pruža različite pogodnosti i jednostavnije korišćenje Google aplikacija, odlučili smo se za ovu temu kako bismo naučili na koji način funkcioniše Google API, koje sve funkcionalnosti obezbeđuje i kako da iste iskoristimo u našoj aplikaciji. Razvili smo bazičnu aplikaciju, koja povlači kontakte, koji se pamte na G-Mail-u i daje detaljan prikaz istih. Obezbedili smo mogućnost ažuriranja (update), brisanja (delete), dodavanja novog kontakta (create new contact). Osnovna zamisao bila je da razumemo koncept i način na koji Google Contacts API funkcioniše, kako bi nam to u daljoj budućnosti pomoglo pri korišćenju drugih Google servisa.

## 2. Pregled korišćene literature

Za izradu projekta koristili smo prvenstveno *Google API* za kontakte, koji smo implementirali u *Node JS*-u. Kako bismo obezbedili funkcionalnost, bio nam je potreban simulirani server, gde smo prilikom rada najčešće koristili open-source platformu *XAMPP*. Za dizajn aplikacije poslužili smo se primenom framework-a *Bootstrap*.

### 1. Šta je Google Contacts API?

API za Google kontakte omogućava klijentskim aplikacijama da pregledaju i ažuriraju kontakte korisnika. Kontakti se čuvaju na korisničkom Google nalogu; većina Google usluga ima pristup listi kontakata.

Klijentska aplikacija može da koristi Google contacts API za kreiranje novih kontakata, izmene ili brisanje postojećih kontakata i postavljanje upita za kontakte koji odgovaraju određenim kriterijumima.

## **2. Kako funkcioniše Google Contacts API**

API podataka o kontaktima nudi dve vrste feedova: feed kontakata i feed grupnih kontakata.

Izvori su privatni feedovi za čitanje / pisanje koji se mogu koristiti za pregled i upravljanje kontaktima / grupama korisnika. Pošto su privatni, programer im može pristupiti samo pomoću autentifikovanog zahteva. Odnosno, zahtev mora da sadrži žeton za potvrdu identiteta za korisnike čije kontakte treba preuzeti.

Reprezentacija kontaktnih podataka zavisi od projekcije. Vrednosti projekcije pokazuju koji su podaci uključeni u reprezentaciju.

API za podatke o kontaktima koristi standardne elemente Google Data API kao i elemente specifične za kontakte.

Konkretno, unos kontakata ima oblik proširene vrste kontakata, koja predstavlja osobu, mesto kao što je klub ili restoran ili organizacija.

## **3. Najčešća primena Google API-a**

Registracija korisnika obično se obavlja putem Google-a, koji omogućava korisnicima da se sigurno prijave u usluge trećih strana pomoću svog Google naloga pomoću sistema *Google sign-in*.

Ovo je trenutno dostupno unutar Android-a (operativnog sistema) ili JavaScript-a. Popularno je u Android aplikacije uključivati dugme „*Sign in with Google*“, jer ručni unos verodostojnih podataka za prijavu može oduzeti mnogo vremena zbog ograničene veličine ekrana. Pošto se korisnik obično prijavljuje na svoj Google nalog na svom mobilnom uređaju, prijava za novu uslugu sa Google-om obično je pitanje samo nekoliko klikova na dugme.



**Slika 2** Značaj Google API-a

Aplikacije za disk su različite veb aplikacije koje rade u Google disku pomoću API-ja za disk. Korisnici mogu da integrišu ove aplikacije u svoj *Drive* iz *Chrome* veb prodavnice što im omogućava da u potpunosti rade u *Cloud*-u. Na raspolaganju je mnogo aplikacija za kolaborativno uređivanje dokumenata (*Google docs*, *Sheets*), uređivanje slika / video zapisa, upravljanje radom ili za skiciranje dijagrama i tokova rada.

Prilagođena pretraga (*Custom Search*) omogućava veb programerima da omoguće pretragu sopstvene veb lokacije ugrađivanjem prilagođenog okvira za pretragu i korišćenjem API-ja za prilagođeno pretraživanje (*Custom Search API*). Oni ne mogu da prilagode rezultate pretrage i zaradjuju novac od oglasa prikazanih pomoću AdSense-a za pretragu.

App Engine su veb aplikacije koje se pokreću na Google App Engine-u, platformi kao servisu (PaaS), koja omogućava veb programerima da pokreću svoje veb lokacije u Google data centrima. Ove veb aplikacije ne mogu koristiti API-je za manipulisanje uslugama kao što su *TaskQueue* (distribuirana čekanja), *BigQuery* (skalabilna baza podataka zasnovana na *Dremel*-u) ili *DataStore*.

*Gadget*-i su mini-aplikacije ugrađene u *HTML*, *JavaScript*, *Adobe Flash* i *Silverlight* koje se ne mogu ugraditi u veb stranice i druge aplikacije. Ne mogu se prikazivati na više veb lokacija i proizvoda (čak i pisanje jednom dozvoljava korisnicima da ih ne mogu pokrenuti na više mesta)

#### **4. XAMPP**

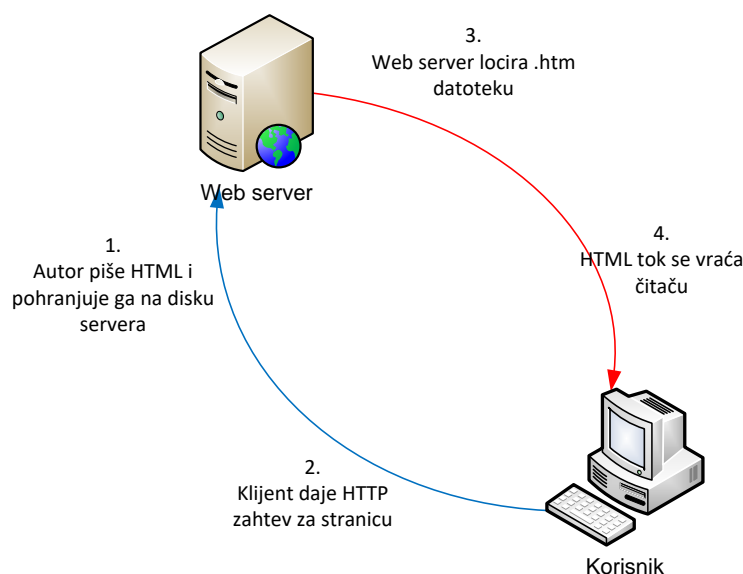
**XAMPP** (Kros-platformski Apache MySQL PHP Perl) je softver koji automatski instalira Apache web server, MySQL bazu i PHP skripting na računaru nezavisno od operativnog sistema samog računara. Ono što je bitno u ovom trenutku je Apache web server koji omogućava da se na računaru nalazi lokalni server. Na taj način na istom računaru će biti i klijent (browser) i server i preko njega se pristupa html stranicama koje se kreiraju. Takođe, XAMPP podržava i aplikacije napisane u programskom jeziku Perl.

#### **Fukcionisanje veze između klijenta i servera na internetu:**

Pristup statičkim stranicama na veb-u teče po sledećem redosledu (**Error! Reference source not found.**):

1. Autor veb sajta kreira statičku veb stranicu i čuva je u određenom folderu na serveru
2. Klijent – veb browser – daje HTTP zahtev za datom stranicom. Zahtev se zadaje kada korisnik računara unese odgovarajući URL u veb browser.
3. Veb server prihvata zahtev klijenta i locira traženu html stranicu

4. Tražena stranica se vraća veb čitaču koji je prikazuje korisniku.



**Slika 1** Statičke veb stranice i način pristupa

Pri korišćenju XAMPP-a, folder na serveru za smeštanje sajta sa stranicama neće biti na nekom udaljenom računaru, već će postojati lokalni server koji se dobija instalacijom XAMPP-a. Na taj način kada se kuca odgovarajući URL u web browser, podaci o toj stranici će se čitati iz odgovarajućeg foldera o kome će biti reči nakon objašnjenja instalacije XAMPP-a. Taj folder se nalazi na korisnikovom računaru.

### 3. Kreiranje aplikacije *My Contacts*

Osnovna zamisao *My Contacts* aplikacije bila je korišćenje mogućnosti uvida u kontakte koji se čuvaju na Google mejl (*G-Mail*) nalogima, a generišu putem mobilnih telefona. Na ovaj način, korisnici mogu dodavati nove kontakte (opcija *Create new contact*), brisati (*delete*) ili ažurirati postojeće kontakte (*update*). Kako bismo olakšali pristup kontaktima i njihovom uređivanju, osmislili smo veoma jednostavnu aplikaciju koja može funkcionisati i bez mobilnog telefona, jednostavnim prijavljivanjem putem mejl adrese korisnika.

Prikaz programskog koda *My Contacts* aplikacije:

```

<!DOCTYPE html>
<html>

<head>
  <title>My contacts</title>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <link rel="stylesheet" href="style.css">
  <link href="https://fonts.googleapis.com/css?family=Amatic+SC&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="ionicons.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>

<body>
  <nav class="navbar navbar-inverse">
    <a class="navbar-brand" href="#">My contacts</a>
    <ul class="nav navbar-nav">
      <li><button type="button" class="btn" id="authorize_button" style="display: none;">Log In</button></li>
      <li><button type="button" class="btn" id="signout_button" style="display: none;">Sign Out</button></li>
    </ul>
  </nav>
  <div id="content" class="container">
    <div class="mb-3">
      <button type="button" onclick="openCreateModal()" class="btn btn-primary">Create New Contact</button>
    </div>
    <table class="table table-striped" id="contacts_table">
      <thead>
        <tr>
          <th>Display Name</th>
          <th>Phone Number</th>
          <th>Email</th>
          <th></th>
          <th></th>
        </tr>
      </thead>
      <tbody></tbody>
    </table>
  </div>
  <div id="myModal" class="modal fade" role="dialog">

```



```

    <div class="modal-dialog">
      <!-- Modal content-->
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-
dismiss="modal">&times;</button>
          <h4 class="modal-title">Contact details</h4>
        </div>
        <div class="modal-body">
          <div class="form-group">
            <label for="firstName">First name:</label>
            <input type="text" class="form-
control" id="firstName">
          </div>
          <div class="form-group">
            <label for="lastName">Last name:</label>
            <input type="text" class="form-
control" id="lastName">
          </div>
          <div class="form-group">
            <label for="mobile">Mobile number:</label>
            <input type="text" class="form-control" id="mobile">
          </div>
          <div class="form-group">
            <label for="emailAddress">Email address:</label>
            <input type="email" class="form-
control" id="emailAddress">
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default" data-
dismiss="modal">Cancel</button>
          <button id="updateBtn" class="btn btn-
primary" onclick="updateConnection()" type="button">Submit</button>
        </div>
      </div>
    </div>
    <script type="text/javascript">
      // Client ID and API key from the Developer Console
      var CLIENT_ID = '72502538284-
v9qn9ff6rkvgubvqgd1ehi9e2js2ul5d.apps.googleusercontent.com';
      var API_KEY = 'AIzaSyDa-ShNTBN2lFqXulejqLMzLNYsn-M-wFY';

      // Array of API discovery doc URLs for APIs used by the quickstart
      var DISCOVERY_DOCS = ["https://www.googleapis.com/discovery/v1/apis/peop
le/v1/rest"];

```

```
// Authorization scopes required by the API; multiple scopes can be
// included, separated by spaces.
var SCOPES = "https://www.google.com/m8/feeds/";

var authorizeButton = document.getElementById('authorize_button');
var signoutButton = document.getElementById('signout_button');

var personObj = {};

// append row to the HTML table
function appendRow(person) {
    var tbl = document.getElementById('contacts_table').getElementsByTag
Name('tbody')[0], // table reference
    row = tbl.insertRow(tbl.rows.length);

    if (person.names && person.names.length > 0) {
        createCell(row.insertCell(), person.names[0].displayName, '');
    } else {
        createCell(row.insertCell(), "No display name found for connecti
on.", '');
    }
    if (person.phoneNumbers && person.phoneNumbers.length > 0) {
        createCell(row.insertCell(), person.phoneNumbers[0].value, '');
    } else {
        createCell(row.insertCell(), "No phone number found for connecti
on.", '');
    }
    if (person.emailAddresses && person.emailAddresses.length > 0) {
        createCell(row.insertCell(), person.emailAddresses[0].value, '')
;
    } else {
        createCell(row.insertCell(), "No email address found for connect
ion.", '');
    }

    createCellBtn(row.insertCell(), 'Details', 'btn btn-
primary', showDetails, person);
    createCellBtn(row.insertCell(), 'Delete', 'btn btn-
danger', deleteConnection, person);
}

// create DIV element and append to the table cell
function createCell(cell, text, style) {
    var div = document.createElement('div'), // create DIV element
        txt = document.createTextNode(text); // create text node
    div.appendChild(txt); // append text node to the DIV
```

```

        div.setAttribute('class', style); // set DIV class attribute
        div.setAttribute('className', style); // set DIV class attribute for
        IE (!)
        cell.appendChild(div); // append DIV to the table cell
    }

    function createCellBtn(cell, text, style, functionRef, handlingObj) {
        var element = document.createElement("button");
        element.innerHTML = text;
        element.setAttribute('class', style);
        cell.appendChild(element);
        element.addEventListener("click", function() {
            functionRef(handlingObj);
        });
    }

    /**
     * On load, called to load the auth2 library and API client library.
     */
    function handleClientLoad() {
        gapi.load('client:auth2', initClient);
    }

    /**
     * Initializes the API client library and sets up sign-in state
     * listeners.
     */
    function initClient() {
        gapi.client.init({
            apiKey: API_KEY,
            clientId: CLIENT_ID,
            discoveryDocs: DISCOVERY_DOCS,
            scope: SCOPES
        }).then(function() {
            // Listen for sign-in state changes.
            gapi.auth2.getAuthInstance().isSignedIn.listen(updateSigninStatu
s);

            // Handle the initial sign-in state.
            updateSigninStatus(gapi.auth2.getAuthInstance().isSignedIn.get()
);

            authorizeButton.onclick = handleAuthClick;
            signoutButton.onclick = handleSignoutClick;
        }, function(error) {
            appendPreText(JSON.stringify(error, null, 2));
        });
    }

```

```
/**
 * Called when the signed in status changes, to update the UI
 * appropriately. After a sign-in, the API is called.
 */
function updateSigninStatus(isSignedIn) {
  if (isSignedIn) {
    authorizeButton.style.display = 'none';
    signoutButton.style.display = 'block';
    listConnectionNames();
  } else {
    authorizeButton.style.display = 'block';
    signoutButton.style.display = 'none';
  }
}

/**
 * Sign in the user upon button click.
 */
function handleAuthClick(event) {
  gapi.auth2.getAuthInstance().signIn();
}

/**
 * Sign out the user upon button click.
 */
function handleSignoutClick(event) {
  gapi.auth2.getAuthInstance().signOut();
}

function appendPreText(message) {
  var pre = document.getElementById('content');
  var textContent = document.createTextNode(message + '\n');
  pre.appendChild(textContent);
}

function onLoadMoreContacts() {
  var loadMoreElem = document.getElementById('load_more');
  loadMoreElem.remove();
}

function appendNextPrevElement(token) {
  var pre = document.getElementById('content');
  var element = document.createElement("button");
  element.setAttribute("id", "load_more");
  element.setAttribute('class', 'btn btn-success');
  element.innerHTML = "Load more contacts";
}
```

```

        pre.appendChild(element);

        element.addEventListener("click", function() {
            listConnectionNames("", token);
        });
    }

    function openCreateModal() {
        personObj = {};
        $('#myModal').modal('show');
        $.clearFormFields('#myModal');
    }

    $.clearFormFields = function(area) {
        $(area).find('input[type="text"],input[type="email"]').val('');
    };

    function showDetails(person) {
        personObj = person;
        $('#myModal').modal('show');
        document.getElementById("firstName").value = person.names ? person.names[0].givenName : '';
        document.getElementById("lastName").value = person.names ? person.names[0].familyName : '';
        document.getElementById("mobile").value = person.phoneNumbers ? person.phoneNumbers[0].value : '';
        document.getElementById("emailAddress").value = person.emailAddresses ? person.emailAddresses[0].value : '';
    }

    function deleteConnection(resource) {
        gapi.client.people.people.deleteContact({
            "resourceName": resource.resourceName
        }).then(function(response) {
            listConnectionNames();
        });
    }

    function updateRequest(person, phone, fname, lname, email) {
        gapi.client.people.people.updateContact({
            "resourceName": person.resourceName,
            "updatePersonFields": "phoneNumbers,names,emailAddresses",
            "etag": person.etag,
            "phoneNumbers": [{
                "value": phone
            }],
            "names": [{

```

```
        "givenName": fname,
        "familyName": lname
    }],
    "emailAddresses": [{
        "value": email
    }]
}).then(function(response) {
    $('#myModal').modal('hide');
    listConnectionNames();
});
}

function createNewRequest(phone, fname, lname, email) {
    gapi.client.people.people.createContact({
        "phoneNumbers": [{
            "value": phone
        }],
        "names": [{
            "givenName": fname,
            "familyName": lname
        }],
        "emailAddresses": [{
            "value": email
        }]
    }).then(function(response) {
        $('#myModal').modal('hide');
        listConnectionNames();
    });
}

function updateConnection() {
    var person = personObj;
    var phoneNumber = document.getElementById('mobile').value;
    var firstName = document.getElementById('firstName').value;
    var lastName = document.getElementById('lastName').value;
    var email = document.getElementById('emailAddress').value;

    if (person && person.resourceName) {
        updateRequest(person, phoneNumber, firstName, lastName, email);
    } else {
        createNewRequest(phoneNumber, firstName, lastName, email);
    }
}

function listConnectionNames(name, pageToken) {
    var namesFilter = name ? name : "";
    var query = {
```

```

        'givenName': namesFilter
    }
    gapi.client.people.people.connections.list({
        'resourceName': 'people/me',
        'pageSize': 20,
        'personFields': 'names,addresses,emailAddresses,phoneNumbers,photos',
        'pageToken': pageToken,
        'query': {
            "givenName": "Jane"
        },
        'sortOrder': 'LAST_MODIFIED_DESCENDING'
    }).then(function(response) {
        var connections = response.result.connections;

        if (pageToken) {
            onLoadMoreContacts();
        } else {
            $("tbody").empty();
        }

        if (connections.length > 0) {
            for (i = 0; i < connections.length; i++) {
                appendRow(connections[i])
            }
            if (response.result.nextPageToken) {
                var loadMoreElem = document.getElementById('load_more');
                if (loadMoreElem) {
                    loadMoreElem.remove();
                }
                appendNextPrevElement(response.result.nextPageToken);
            }
        } else {
            appendPreText('No connections found.');
```

Programski deo koda u CSS-u:

```
*{  
    font-family: "Amatic SC";  
    font-size: 30px;  
}  
  
body{  
    background-color: tan;  
}  
  
.navbar-inverse{  
    background-color: dimgray;  
    border-color: dimgray;  
}  
  
.navbar-inverse .navbar-brand{  
    color: rgba(255, 255, 255, 0.644);  
    margin-left: 20px;  
    font-size: 30px;  
}  
  
.navbar-inverse .navbar-brand:hover{  
    color: white;  
    transition-duration: 0.6s;  
}  
  
.btn{  
    font-size: 20px;  
}  
  
li #authorize_button{  
    margin: 5px auto 5px 1250px;  
    font-size: 22px;  
}  
  
li #authorize_button:hover{  
    background-color: rgba(255, 255, 255, 0.521);  
    transition-duration: 0.4s;  
}  
  
li #signout_button{  
    margin: 8px auto 8px 1250px;  
    font-size: 22px;  
}  
  
li #signout_button:hover{  
    background-color: rgba(255, 255, 255, 0.521);  
    transition-duration: 0.4s;
```



```
}

.btn-danger{
    background-color: indianred;
    border: none;
}

.btn-danger:hover{
    background-color: rgba(205, 92, 92, 0.5);
    border: none;
    transition-duration: 0.5s;
}

.container{
    background-color: tan;
}

.mb-3{
    margin-top: 5px;;
}

.btn-primary{
    background-color: rgb(32, 178, 170);
    border:none;
}

.btn-primary:hover{
    background-color:rgba(32, 178, 170,0.4) ;
    transition-duration: 0.5s;
}

.btn-default{
    transition-duration: 0.4s;
}

.form-control{
    font-style: italic;
    font-size: 25px;
}

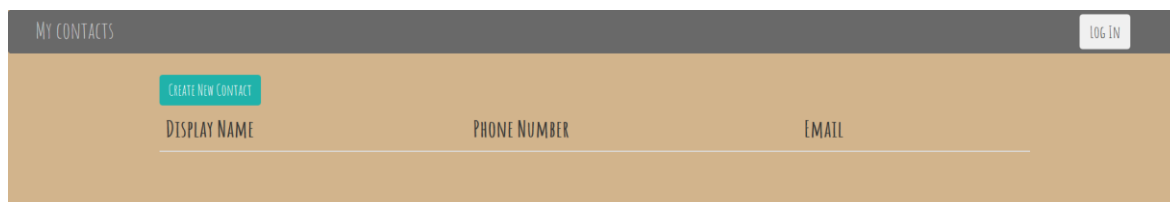
.modal-title{
    font-size: 30px;
}

.btn-success{
    background-color: mediumseagreen;
}

.btn-success:hover{
    background-color: rgba(60, 179, 114, 0.61);
    transition-duration: 0.4s;
    border: none;
}
```

```
}
```

Pre nego što se uloguje, korisniku se prikazuje:



Display Name	Phone Number	Email
--------------	--------------	-------

Nakon što korisnik unese svoju mejl adresu, izlistaće mu se 20 kontakata poslednje modifikovanih, koje je memorisao na mejlu:



Display Name	Phone Number	Email		
VLAJKO	+381 60 5404016	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
SMARTPOINT	+381 62 667358	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
TATA 064	+381 64 5695702	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
MARIJANA ANDRIJANINA	+381 65 6816117	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
IVICA MILADINović	065 2222871	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
NIKOLA SIćOVIć	+381 65 9190398	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
SINIŠA STANIVUK FON	+381 69 2609935	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete
SINIŠA STANIVUK	+381 69 2609935	NO EMAIL ADDRESS FOUND FOR CONNECTION.	Details	Delete

Kreiranje novog kontakta:

The screenshot displays the 'My Contacts' application interface. On the left, a sidebar lists contacts: 'VLADKO', 'SMARTPOINT', 'TATA OGA', 'MARIJANA ANDRIJANINA', 'IVICA MILADINOVIC', 'NIKOLA SICOVIC', 'SINISA STANIVUK FON', and 'SINISA STANIVUK'. The main area shows a modal titled 'CONTACT DETAILS' for editing the contact 'SINISA STANIVUK'. The modal contains input fields for 'FIRST NAME:', 'LAST NAME:', 'MOBILE NUMBER:', and 'EMAIL ADDRESS:'. Below the modal, a message states 'No email address found for connection.' The right sidebar shows buttons for 'DETAILS' and 'DELETE' for each contact.

Prilikom klika na dugme *details*:

[illegible]

## 4. Zaključak

Temu kreiranja aplikacije koja koristi Google API za kontakte smo izabrali jer nam sa jedne strane pruža mogućnost da naučimo nešto što nam u daljem radu može veoma značiti prilikom izrade nekih funkcionalnosti koje je potrebno implementirati, a sa druge strane i raznovrsne mogućnosti za učenje kroz rad na ovom projektu. Radom na ovom projektu smo naučili nove načine realizacije određenih ideja, kao i nove alate koji su nam doprineli u tome. Nas ova oblast prvenstveno i zanima i smatramo da nam je izrada ovog rada pružila odličnu osnovu za njeno dalje izučavanje.

## 5. Literatura

- [1] <https://developers.google.com/products/develop>
- [2] <https://www.w3schools.com/bootstrap/>
- [3] <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [4] <https://stackoverflow.com/questions/37726602/using-google-api-to-modify-google-contacts>
- [5] <https://getbootstrap.com/docs/4.4/layout/overview/>