



Е-БИБЛИОТЕКА

СЕМИНАРСКА РАБОТА ПО ПРЕДМЕТОТ WEB СЕРВИСИ



Изработила: Динева
Ивана 156/2019 КТИ
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И
ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

Содржина

I.	Вовед	2
1.1.	SOAP – Simple Object Access Protocol.....	3
1.2.	REST – Representative State Transfer.....	3
II.	Главен дел.....	4
2.1.	Што користев за потребите на проектот?	4
2.2.	Вовед во проектот	5
2.3.	Како го поврзав Node.js со MongoDB Database/ базата на податоци?	6
2.3.1.	Приказ на Mongo DB со помош на MongoDB Compass	7
2.4.1.	Стандарди за безбедност на веб сервиси	8
2.5.	Имплементација на сервис за логирање и регистрирање со SOAP.....	9
2.6.	REST функции.....	11
2.6.1.	Листање на сите корисници	12
2.6.2.	Додади корисник во библиотеката	13
2.6.3.	Прикажи детали за постоечки корисник.....	14
2.6.4.	Избриши постоечки корисник	15
2.7.	Автентикација на сервисот со Passport.js	16
III.	Заклучок.....	16
3.1.	Финален изглед на мојата библиотека!.....	18

I. Вовед

За тема на проектот WEB Сервиси ја одбрав темата Библиотека. Задачи кои што требаше да ги извршам, и функционалности кои што треба да ги има библиотеката се: сервис кој ќе нуди работа со библиотека и изнајмување на книги во библиотека. Треба да се излистаат сите книги што ги има на располагање со статус: слободни, за даден корисник да се врати кои книги ги должи и слично. Треба да се додадат функционалности за додавање нови книги, изнајмување на книги од корисници, ажурирање на податоци за книгата и слично.

Области кои треба да бидат вклучени во проектот:

- Имплементација на сервис за логирање, регистрирање и одрегистрирање;
- 12 функции дополнителни (покрај тие од претходната точка) специфични за семинарската работа;
- Половина од функциите да бидат имплементирани со REST, другата половина со SOAP;
- Клиент/и за испробување на работата на сервисите;
- Аспекти на обезбедување на повикот на функциите (secure service call);
- Елаборат, код и презентација;
- Имплементација на сервисите со помош на glassfish/**nodejs**/spring.

*За имплементација на сервисите ќе користам **Node.js**.*

Е-Библиотека е систем кој ги одржува информациите за книгите присутни во библиотеката, нивните автори, членовите на библиотеката на кои им се издаваат книгите, персоналот на библиотеката и сите други. Ова е многу тешко да се организира рачно. Рачното одржување на сите овие информации е многу сложена задача. Благодарение на напредокот на технологијата, организацијата на Е-Библиотека станува многу едноставна. Управувањето со Е-Библиотеката е дизајнирано да ги компјутеризира и автоматизира операциите што се вршат над информациите за членовите, прашањата и враќањето на книгите и сите други операции. Оваа компјутеризација на библиотеката помага во многу случаи на нејзино одржување. Го намалува обемот на работа на менаџментот бидејќи најголемиот дел од извршената мануелна работа се намалува.

REST и SOAP се две технологии за веб сервиси кои го револуционизираат начинот на кој бизнисите користат API. Бидејќи API-ите се толку вообичаени, важно е редовно да ги тестирате за да пронајдете пропусти или потенцијални оперативни проблеми во рамките на веб сервисот.

1.1. SOAP – Simple Object Access Protocol

SOAP кратенка за Simple Object Access Protocol. Microsoft првично излезе со SOAP за да ги олесни сервисите за пораки на Интернет. Користи XML исклучиво за испраќање и примање податоци. Кога SOAP за прв пат беше објавен, потпирањето на XML беше голема причина да се прифати како стандард. Бидејќи XML е јазик базиран на текст добро поддржан на сите главни платформи, тој беше совршено прилагоден за испраќање податоци преку API.

По првичното објавување на SOAP, Microsoft го предаде развојот и одржувањето на Работната група за Интернет инженерство (IETF), каде што на крајот стана веб-стандард. Како прв популарен веб сервис, развиена е долга листа на протоколи и стандарди за употреба со SOAP, повеќето од нив почнувајќи со иницијалите WS (Web Services). Еден од најважните е WS Security, безбедносниот протокол кој ги обезбедува преносите на SOAP..

SOAP е првенствено поврзан со HTTP. Може да се користи и со други интернет протоколи, како што е SMTP. Додека SOAP имаше долг рок како широко користен протокол, со текот на времето, програмерите почнаа да сакаат нешто пофлексибилно. Не сите програмски јазици се добри во извозот на потребната XML потребна за да се направи барање за SOAP, создавајќи потреба за протокол што може да користи други формати за пораки.

1.2. REST – Representative State Transfer

Потребата за поголема флексибилност кај развивачите на веб сервиси за создавање на неколку алтернативи. Денес, REST е најпопуларен. REST е кратенка за Representative State Transfer и сега е стандарден избор за повеќето API-и. За разлика од SOAP, REST не зависи исклучиво од употребата на XML. Најчесто користен јазик со REST е JSON.

REST им дава на програмерите широка архитектонска рамка за работа. Обично се потпира на едноставна URL адреса за испраќање информации. Преку форматот на URL, REST може да користи вообичаени задачи на HTTP 1.1 како GET, POST, PUT и DELETE за да ги исполни задачите. REST поддржува HTTPS, но без строгите безбедносни барања на SOAP.

Програмерите ги нарекуваат API кои се во согласност со REST стандардите RESTful API или едноставно REST API. DreamFactory е REST API-како услуга платформа. Им помагаме на компаниите да изградат и управуваат со безбедни REST API и да ги следат и тестираат нивните решенија.

II. Главен дел

Низ овој дел ќе ги разгледаме засебно, сите побарувања кои што треба да ги вклучува проектот.

2.1. Што користев за потребите на проектот?

За потребите на проектот од Front-end технологии користев *HTML*, *CSS*, *BOOTSTRAP*, *JavaScript* и *jQuery*. Од Back-end технологии за потребите на проектот користев *MongoDB*, *Express.js* и *Node.js* како главна технологија за имплементација на проектот.

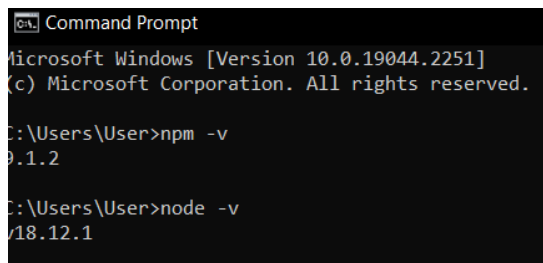
Функционалности кои може да ги изврши библиотекарот се login, logout, да ги следи сите активности на корисникот, да додава книги, да ажурира копии на веќе постоечка книга, да брише книги, да пребарува книги според категорија, наслов, автор или ISBN. Потоа може да пребарува корисници според нивното име, презиме, email или корисничко име. Да избрише account на некој корисник, да праќа известување со сите/индивидуални или филтрирани корисници. Исто така библиотекарот може да си ја промени фотографијата или лозинката, да додаде нов админ.

Функционалности кои може да ги изврши член на библиотеката се да се регистрира во библиотеката ако веќе нема постоечки профил, ако има постоечки профил да се најави во

библиотека, да побара книга, да врати книга, да пребарува книга, да додаде, едита или да избрише коментар на некоја книга, да ја промени својата профилна фотографија, да ја промени лозинката или пак да го изврши неговиот постоечки профил.

2.2. Вовед во проектот

За потребите на проектот инсталирано ми е Node.js. Правам проверка на инсталацијата со помош на наредбите `node -v` & `npm -v`.



```

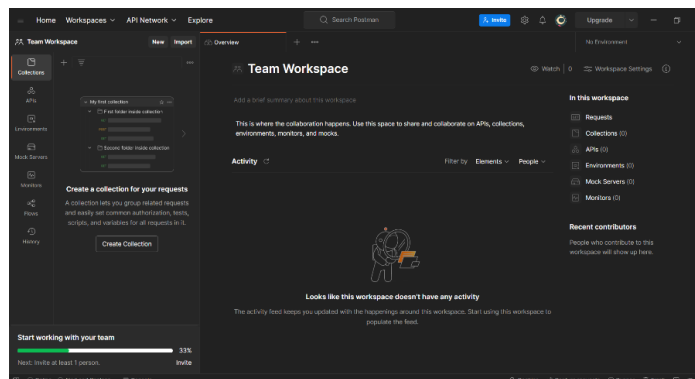
Command Prompt
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>npm -v
9.1.2

C:\Users\User>node -v
v18.12.1
  
```

Слика 1: Верзии на инсталација на Node.js

Postman успешно го инсталирав за потребите на REST за проектот.



Слика 2: Postman

За креирање на проектот иницијално почнувам со наредбите `mkdir e_library` & `cd e_library`. Потоа иницијализирам пакети потребни за работа со nodejs со помош на наредбата `npm init`. И инсталирам express со помош на наредбата `npm install express`.

```

C:\Users\User>mkdir e_library
C:\Users\User>cd e_library
C:\Users\User\e_library>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (e_library) e_library
version: (1.0.0) 1.0.0
description: library
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: Ivana
license: (ISC)
about to write to C:\Users\User\e_library\package.json:
{
  "name": "e_library",
  "version": "1.0.0",
  "description": "library",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Ivana",
  "license": "ISC"
}
Is this OK? (yes) yes
C:\Users\User\e_library>npm install express
added 57 packages, and audited 58 packages in 2s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

```

Слика 3: Иницијално креирање на проектот

2.3. Како го поврзав Node.js со MongoDB Database/ базата на податоци?

MongoDB е NoSQL база на податоци што се користи за складирање на големи количини на податоци без традиционална табела со релациони бази на податоци. Наместо редови и колони, MongoDB користи колекции и документи за складирање на податоци. Колекциите се состојат од збир од документи, и документот се состои од парови клуч-вредност кои се основна единица на податоци во MongoDB.

За да ја поврземе Node.js апликацијата со MongoDB, треба да ја повикаме библиотеката наречена *Mongoose*,

```
const mongoose = require("mongoose");
```

После тоа, треба да го повикаме методот за поврзување на *Mongoose* со помош на кодот,

```
mongoose.connect("mongodb://localhost:27017/collectionName", {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
```

Потоа треба да дефинираме шема. Шема е структура која дава информации за тоа како податоците се складираат во колекцијата.

2.3.1. Приказ на Mongo DB со помош на MongoDB Compass

Collections				
activities				
<u>Storage size:</u> 36.86 kB	<u>Documents:</u> 7	<u>Avg. document size:</u> 247.00 B	<u>Indexes:</u> 1	<u>Total index size:</u> 36.86 kB
books				
<u>Storage size:</u> 36.86 kB	<u>Documents:</u> 6	<u>Avg. document size:</u> 1.02 kB	<u>Indexes:</u> 1	<u>Total index size:</u> 36.86 kB
issues				
<u>Storage size:</u> 32.77 kB	<u>Documents:</u> 3	<u>Avg. document size:</u> 289.00 B	<u>Indexes:</u> 1	<u>Total index size:</u> 32.77 kB
sessions				
<u>Storage size:</u> 36.86 kB	<u>Documents:</u> 1	<u>Avg. document size:</u> 210.00 B	<u>Indexes:</u> 2	<u>Total index size:</u> 53.25 kB
users				
<u>Storage size:</u> 36.86 kB	<u>Documents:</u> 3	<u>Avg. document size:</u> 1.40 kB	<u>Indexes:</u> 1	<u>Total index size:</u> 36.86 kB

Слика 4: Приказ на табели кои се креирани

library.users

1

DOCUMENTS

INDEXED

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

Filter

Type a query: { field: 'value' }

Reset

Find

More Options

ADD DATA

EXPORT COLLECTION

1 - 3 of 3

```

_id: ObjectId('63a0e32f021be1de044e3c')
joined: 2022-12-19T22:06:34.202+00:00
image: ""
violationFlag: false
fines: 0
isDefin: false
firstName: "Angela"
lastName: "Dineva"
username: "dinevaa"
email: "dinevaangela33@yahoo.com"
gender: "Female"
address: "Ivan agovski 4a"
bookIssueInfo: Array
salt: "29809f5b18b7ad41da774f05766472af13d25e0674b34aa1ddb6ca7c110828fb"
hash: "63feb469b3e24db8b7837b71b9bb0862139656e99dc9dcd2f8d9237561b2cac8dd9701-"
__v: 2

```

Слика 5: Приказ на табелата корисници-users

library.books

DOCUMENTS

INDEX

Documents

Aggregations

Sohema

Explain Plan

Indexes

Validation

Filter

Type a query: { field: 'value' }

Reset

Find

More Options

ADD DATA

EXPORT COLLECTION

1 of 6

Navigation icons

```

      _id: ObjectId('639dd126b693ac1ce8f2fd1')
    > comments: Array
      title: "Life of pi"
      author: "Yann Martel"
      ISBN: "124578693625"
      category: "Roman"
      stock: 10
      description: "<p><em><strong>Life of Pi</strong></em>&nbsp;&nbsp;&nbsp;is a Canadian&nbsp;&nbsp;&nbsp;philos_"
      _v: 0
    }
  ]
}

      _id: ObjectId('639dd1ce6b93ac1ce8f2f6fa')
    > comments: Array
      title: "Message in a Bottle"
      author: "Nicholas Sparks"
      ISBN: "124578699876"
      category: "Roman"
      stock: 14
      description: "<p>&ldquo;I am lost without you. I am soulless, a drifter without a ho_"
    }
  ]
}

```

Слика 7: Приказ на табелата книги- books

2.4. Аспекти на обезбедување на повикот на функциите (secure service call)

WS Security е стандард кој се однесува на безбедноста кога податоците се разменуваат како дел од веб сервисот. Ова е клучна карактеристика во SOAP што го прави многу популарен за креирање веб сервиси.

Безбедноста е важна карактеристика во секој веб сервис. Бидејќи сите веб сервиси се изложени на Интернет, секогаш постои можност за безбедносна закана за веб сервисите. Оттука, кога развиваме веб сервиси, секогаш се препорачува да се осигураме дека сервисот е дизајниран и развиен имајќи ја предвид безбедноста.

SOAP обезбедува дополнителен слој наречен WS Security за обезбедување дополнителна безбедност кога се прават повици до веб сервиси. WS Security може да се повика со едноставно корисничко име или лозинка или може да се користи со бинарни сертификати за автентикација. Бидејќи телото на SOAP е шифрирано, ќе може да се дешифрира само од веб серверот што го хостира веб сервисот. Ова е поради тоа како е дизајниран SOAP.

Да претпоставиме дека ако пораката е предадена на серверот на базата на податоци во барање HTTP, таа не може да се дешифрира бидејќи базата на податоци нема соодветни механизми за тоа. Само кога барањето навистина ќе стигне до веб-серверот како SOAP, тој ќе може да ја дешифрира пораката и да го испрати соодветниот одговор назад до клиентот.

2.4.1. Стандарди за безбедност на веб сервиси

Стандардот WS-Security се врти околу тоа што безбедносната дефиниција е вклучена во заглавието на SOAP. Ингеренциите во SOAP заглавието се управуваат на 2 начини.

Прво, тој дефинира посебен елемент наречен UsernameToken. Ова се користи за предавање на корисничкото име и лозинката на веб сервисот.

Другиот начин е да користи бинарен токен преку BinarySecurityToken. Ова се користи во ситуации во кои се користат техники за шифрирање како Kerberos или X.509.

1. Може да се испрати барање од клиентот на веб сервисот до сервисот за безбедносни токени. Оваа услуга може да биде среден веб сервис која е специјално изградена за да обезбеди кориснички имиња/ лозинки или сертификати на вистинскиот веб сервис SOAP.
2. Безбедносниот токен потоа се пренесува до клиентот на веб сервисот.
3. Клиентот на веб сервисот потоа го повикува веб сервисот, но, овој пар, осигурувајќи дека безбедносниот токен е вграден во SOAP пораката.
4. Веб сервисот ја разбира SOAP пораката со токенот за автентикација и потоа може да ја контактира услугата за безбедносен токен за да види дали безбедносниот токен е автентичен или не.

Поголу кодот го прикажува форматот на делот за автентикација кој е дел од WSDL документот. Според кодот, SOAP пораката ќе содржи 2 дополнителни елементи, еден е корисничко име а другиот е лозинка.

```
<xs:element name="UsernameToken">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Username"/>
      <xs:element ref="Password" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Id" type="xs:ID"/>
  </xs:complexType></xs:element>
```

2.5. Имплементација на сервис за логирање и регистрирање со SOAP

Иницијално за да користиме SOAP, при имплементација на сервис за Библиотека користиме `var soap = require('soap');`

За да можеме да го вчитаме WSDL- Web Service Definition Language документот користиме `var xml = fs.readFileSync('service.wsdl', 'utf8');`

```
app.listen(port, function () {
  console.log('Listening on port ' + port);
  var wsdl_path = "/wsdl";
```

```

    soap.listen(app, wsdl_path, serviceObject, xml);
    console.log("Check http://localhost:" + port + wsdl_path + "?wsdl to see if the service is
    working");
  });

```

За да користиме модул во Node.js апликацијата, прво треба да го вклучиме.

```

const express = require('express');
const session = require('express-session');
const path = require('path');

```

Треба да се поврземе со базата на податоци. И да додадеме код за иницијализација на express. `const app = express();`

Следно, треба да ги поврземе модулите што ќе ги користиме со express.

```

app.use(session({ secret: 'secret', resave: true, saveUninitialized: true }));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'static')));

```

Следно, треба да ја декларираме рутата за најавување што ќе ја даде login.html датотеката на клиентот, користејќи GET барање.

```

app.get('/', function(request, response) { response.sendFile(path.join(__dirname +
'/login.html')); });

```

Следно, треба да додадеме нива рута која ќе го автентифицира корисникот.

```

app.post('/auth', function(request, response) {
  let username = request.body.username;
  let password = request.body.password;
  if (username && password) {
    connection.query('SELECT * FROM accounts WHERE username = ? AND password = ?',
    [username, password], function(error, results, fields) {
      if (error) throw error;
      if (results.length > 0) {
        request.session.loggedin = true;
        request.session.username = username;

```

```

        response.redirect('/home');
    } else { response.send('Incorrect Username and/or Password!'); }
    response.end(); });
} else { response.send('Please enter Username and Password!');
    response.end(); } });

```

Конечно можеме да ја креираме домашната рута што ќе го даде корисничкото име на корисникот.

```

app.get('/home', function(request, response) {
    if (request.session.loggedin) {
        response.send('Welcome back, ' + request.session.username + '!');
    } else {
        response.send('Please login to view this page!'); }
    response.end(); });

```

За крај, нашиот Node.js сервер треба да слуша на порта, ја користиме портата 5000.

```
app.listen(5000);
```

2.6. REST функции

Ако претпоставиме дека имаме JSON based database од корисници, ако ги имаме следниве корисници во датотеката users.json:

```

{ "user1" : {
    "name" : "ivana",
    "password" : "password1",
    "profession" : "librarian",
    "id": 1 },
  "user2" : {
    "name" : "mario",
    "password" : "password2",
    "profession" : "user",
    "id": 2 } }

```

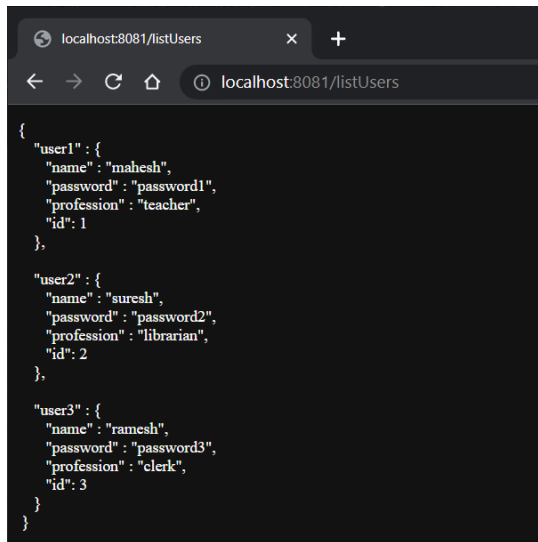
URI	HTTP Метод	POST Body	Резултат
listUsers	GET	Празно	Прикажи листа од сите корисници
addUser	POST	JSON стринг	Додади детали на нов корисник
deleteUser	DELETE	JSON стринг	Избриши веќе постоечки корисник
:id	GET	Празно	Прикажи детали за корисник

2.6.1. Листање на сите корисници

Во датотеката `server.js` го додаваме следниот код, ако сакаме да ги излистаме сите корисници кои се претплатени на датотеката

```
var express = require('express');
var app = express();
var fs = require("fs");
app.get('/listUsers', function (req, res) {
  fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
    console.log( data );
    res.end( data );  });
})
var server = app.listen(5000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

За да пристапиме до ова API го користиме URL-то: `localhost:5000/listUsers` и HTTP метод: GET.



Слика 8: Излез за листање на сите корисници

2.6.2. Додади корисник во библиотеката

За да додадеме нов корисник во библиотеката, го додаваме следниот код во датотеката `server.js`

```

var express = require('express');
var app = express();
var fs = require("fs");
var user = {
  "user4": {
    "name": "leo",
    "password": "password4",
    "profession": "teacher",
    "id": 4 } }
app.post('/addUser', function (req, res) { // First read existing users.
  fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
    data = JSON.parse( data );
    data["user4"] = user["user4"];
    console.log( data );
    res.end( JSON.stringify(data));  });
var server = app.listen(5000, function () {

```

```
var host = server.address().address
var port = server.address().port
console.log("Example app listening at http://%s:%s", host, port) })
```

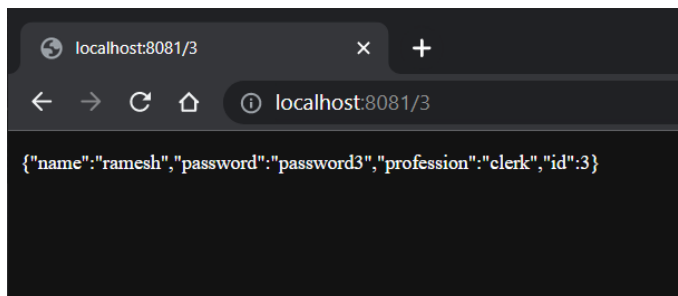
За да пристапиме до ова API го користиме URL-то: localhost:5000/addUser и HTTP метод: POST.

2.6.3. Прикажи детали за постоечки корисник

Ако сакаме да ни бидат прикажани деталите за некој корисник, го додаваме следниот код во server.js датореката

```
var express = require('express');
var app = express();
var fs = require("fs");
app.get('/:id', function (req, res) { // First read existing users.
  fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
    var users = JSON.parse( data );
    var user = users["user" + req.params.id]
    console.log( user );
    res.end( JSON.stringify(user));  });
})
var server = app.listen(5000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port) })
```

За да пристапиме до ова API го користиме URL-то: localhost:5000/2 и HTTP метод: GET. /1, /2, ... во зависност од тоа за кој корисник сакаме да ни бидат прикажани деталите.



Слика 9: Излез за приказ на детали за постоечки корисник

2.6.4. Избриши постоечки корисник

Ако сакаме да избришеме веќе постоечки корисник од библиотеката, го додаваме следниот код во `server.js` датотеката

```
var express = require('express');
var app = express();
var fs = require("fs");
var id = 2;
app.delete('/deleteUser', function (req, res) { // First read existing users.
  fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
    data = JSON.parse( data );
    delete data["user" + 2];
    console.log( data );
    res.end( JSON.stringify(data));  });
})
var server = app.listen(5000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port) })
```

За да пристапиме до ова API го користиме URL-то: `localhost:5000/deleteUser` и HTTP метод: DELETE.

2.7. Автентикација на сервисот со Passport.js

Автентикација е процес на одредување дали некој или нешто е, всушност, кој или што вели дека е. Технологијата за автентикација обезбедува контрола на пристапот за системите со проверка дали акредитациите на корисникот се совпаѓаат со ингеренциите во базата на податоци на овластени корисници или во серверот за автентикација на податоци. Притоа, автентикацијата гарантира безбедни системи, безбедни процеси и безбедност на информациите на претпријатието.

Passport е популарен, модуларен софтвер за автентикација за Node.js. Со него, автентикацијата може лесно да се интегрира во која било апликација базирана на Node и Express. Библиотеката Passport обезбедува повеќе од 500 механизми за автентикација, вклучувајќи OAuth, JWT и едноставна автентикација базирана на корисничко име и лозинка.

Користењето на Passport.js го олеснува интегрирањето на повеќе од еден тип на автентикација во сервисот.

III. Заклучок

За потребите на предметот WEB Сервиси, изработив проект на тема Е-Библиотека, со помош на REST и SOAP. Наизменично користев и SOAP и REST, земајќи ги во обзир ситуациите каде што единиот има предност пред другиот.

SOAP се препорачува за enterprise апликации, high-security апликации, дистрибуирана средина, financial сервиси, payment gateways, телекомуникациски сервиси итн. Додека REST се препорачува за public API-а за веб сервиси, мобилни сервиси и социјални мрежи.

Кога имаме лимитирани ресурси и пропусен опсег подобро е да се користи REST, бидејќи SOAP пораките се потешки и конзумираат многу поголем пропусен опсег. Кога мрежниот пропусен опсег е ограничување, препорачано е да се користи REST. Исто така REST се препорачува во состојба на statelessness, кога нема потреба да се одржува

состојбата на информацијата од еден request до друг. Но, кога ни треба proper information flow, од еден request до друг, SOAP е покомпатибилен.

Кога треба да кешираме многу request-и тогаш се користи REST, во ситуации кога клиентот може да го побара истиот ресурс повеќе пати. Ова ќе го зголеми бројот на request-и кои се пратени до серверто. Со имплементирање на кеширање, најфреквентните queries results може да бидат складирани на intermediate локација, па кога клиентот ќе побара ресурс прво ќе го провери кешот, ако ресурсите се во кешот нема да пристапува до серверот. Кеширањето ги намалува пристапите до серверот.

Кога ни треба ease of coding, REST е многу полесен од SOAP.

SOAP користиме кога ни е потребно асинхронно процесирање и subsequent invocation. Кога на клиентот му е потребна гарантиран левел на доверливост и сигурност, тогаш дефинитивно SOAP.

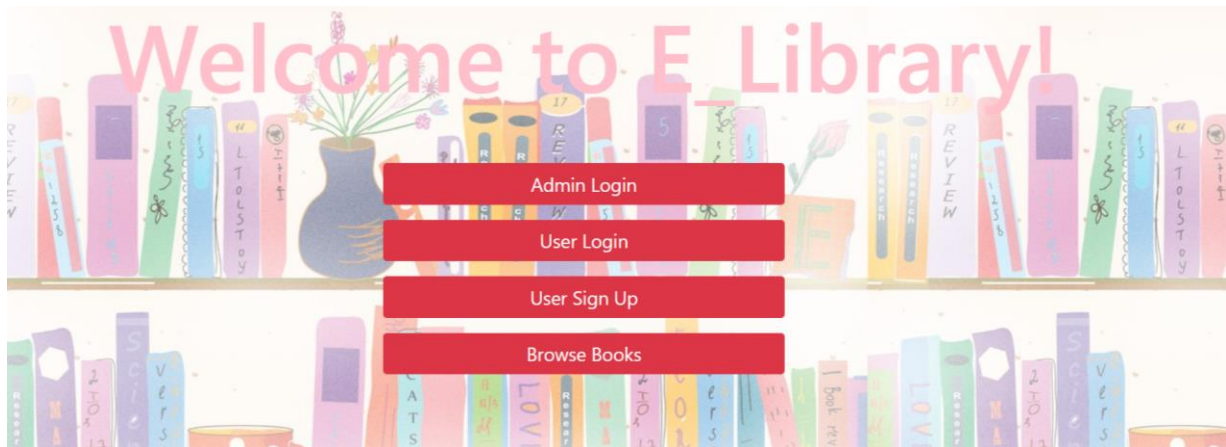
Кога имаме потреба од formal means of communication – SOAP. Кога и клиентот и серверот имаат договор за exchange формат, тогаш SOAP ни дава поригидна спецификација.

Користи SOAP во stateful операции. Ако апликацијата има побарување дека состојбата треба да се одржува од еден request до друг, користи SOAP.

Предизвици во SOAP => Еден од предизвиците е WSDL документот, кој е оној кој му кажува на клиентот сите операции кои што можат да бидат изведени од веб сервисот. Овој документ ги содржи сите информации кои што се користат во SOAP пораките и кои се сите операции кои се достапни преку веб сервисот. Друг предизвик е големината на SOAP пораката, која се префрла од клиентот до серверот. Големи пораки кои користат SOAP и места каде пропусната моќ е ограничување, се проблем.

Предизвици во REST => Недостаток на безбедност. REST не ни обезбедува сигурност. И затоа REST е многу соодветен за јавно достапни URLs. Кога треба доверливи податоци да бидат пратени помеѓу клиент и сервер, REST е најлошиот механизам кој може да се користи. Следно е недостатокот на состојба. Повеќето веб апликации побаруваат stateful механизам.

3.1. Финален изглед на мојата библиотека!



Слика 10: Добредојдовте во мојата библиотека!

[Home](#)
[Book Inventory](#)
[Users](#)
[Add Books](#)
[Browse Books](#)

Welcome dinevai ▾

Recent User Activities		
Info	Category	Date Posted
dinevai issued To the Lake: A Balkan Journey of War and Peace	Issue	Wed Dec 21 2022
dinevaa issued Message in a Bottle	Issue	Mon Dec 19 2022
dinevaa issued It Starts with Us	Issue	Mon Dec 19 2022
dinevai issued Life of Pi	Issue	Mon Dec 19 2022
dinevai returned Life of Pi	Return	Mon Dec 19 2022
trajkovad issued Message in a Bottle	Issue	Sat Dec 17 2022
trajkovad returned Message in a Bottle	Return	Sat Dec 17 2022

[First](#)
[1](#)
[Last](#)

Books

6

View

Users


2

Users

Слика 11: Најава како админ во библиотеката

[Home](#)
[Browse Books](#)
Welcome dinevaa

[+ Issue Book](#)
[Renew/Return Book](#)



Name : Angela Dineva
 Email : dinevaangela33@yahoo.com
 Book Issued : 2
 Flagged : false
 Joined : Mon Dec 19 2022

Recent Activities

Info	Category	Date
You have issued It Starts with Us	Issue	Issue : Mon Dec 19 2022 Return : Mon Dec 26 2022
You have issued Message in a Bottle	Issue	Issue : Mon Dec 19 2022 Return : Mon Dec 26 2022

[First](#)
[1](#)
[Last](#)

Слика 12: Најава како корисник на библиотеката

[Home](#)
[Browse Books](#)
Welcome dinevaa

Life of Pi

Author : Yann Martel

Category : Roman

In stock : 10

[Issue](#)
[Details](#)

Message in a Bottle

Author : Nicholas Sparks

Category : Roman

In stock : 14

[Issued!](#)
[Return/Renew](#)
[Details](#)

To the Lake: A Balkan Journey of War and Peace

Author : Kapka Kassabova

Category : Biography

In stock : 4

[Issue](#)
[Details](#)

Remnants of Another Age

Author : Nikola Madzirov

Category : Poetry

In stock : 20

[Issue](#)
[Details](#)

In Search of Lost Time

Author : Marcel Proust

Category : Novel

In stock : 25

[Issue](#)
[Details](#)

It Starts with Us

Author : Colleen Hoover

Category : Romance

In stock : 19

[Issued!](#)
[Return/Renew](#)
[Details](#)

[First](#)
[1](#)
[Last](#)

Слика 13: Приказ на сите книги достапни во библиотеката