Steg 1

Installera Node.js och Express
Verifiera installation med
$ node -v
$ express –version

Steg 2

Använd express generator för att skapa målmapp och projekt – Projektnamn: zoomba
$ express zoomba
$ cd zoomba
$ npm install

Provkör applikation med (avslutas med CTRL+C)
$ npm start
eller
$ nodemon start

Applikationen lyssnar på port 3000!
Testa med en webbläsare: http://localhost:3000 samt http://localhost:3000/users/

package.json innehåller applikationens konfigurationsinformation samt information om de moduler som ska infogas och användas i applikationen (npm install skapar katalogen node_modules ionnhållande alla moduler baserat på denna konfigurationsfil).
Vid flytt av projektet tar man normalt bort katalogen node_modules som enkelt kan återskapas med npm install.

Mer information om möjliga konfigurationsinställningar: https://docs.npmjs.com/files/package.json

```
{
 "name": "zoomba",
 "version": "0.0.0",
 "private": true,
 "scripts": {
   "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1"
  }
```

}

I detta fall behöver vi cookie-parser(sköter cookieshantering), debug(felsökning under utvecklingsfasen), http-errors(hantering av http felmeddelanden), jade(stöd för utveckling med jade) samt morgan(modul som hanterar loggning av anrop).

Filer och kataloger som är intressanta för oss i detta skede är app.js samt katalogerna public, views och routes.

Katalogstruktur:

- *zoomba/views*
  Här placerar vi hybrid HTMLsidor skapade med jade syntax.
- *zoomba/public*
  Här placerar vi HTMLsidor skapade med HTML, CSS och JS.
- zoomba/routes
  Här hittar vi de filer som anropas vid request till skapad webbtjänsten

I vårt exempelprojekt vill vi att webbtjänsten ska namnges urls.js istället för users.js.

Döp om filen users.js till urls.js!

Gå till rootkatalogen zoomba och öppna filen app.js!

```
.
:
var indexRouter = require('./routes/index');
var urlsRouter = require('./routes/urls');   ←----- Rad 8

var app = express();
:
.

app.use('/', indexRouter);
app.use('/urls', urlsRouter);       <------------- Rad 23

// catch 404 and forward to error handler
:
.
```

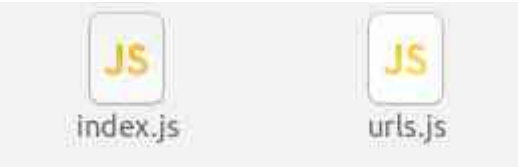
SPARA!

Provkör med npm start!

Om allt fungerar utan felmeddelande är vi redo att gå vidare med implementation av webbtjänsten.

Steg 3

Gå till katalog zoomba/routes där vi finner filerna



Filen urls,js hanterar anrop till webbtjänsten på adress http://localhost:3000/urls/
För att enkelt provköra denna implementerar vi RESTwebbtjänsten med HTTPverbet GET

Öppna urls.js och ändra till

```
var express = require('express');
var router = express.Router();

var users = [
  {_id:0,name:"Mikael Hasselmalm",url:"https://miun-se.zoom.us/my/hasselmalm/"},
  {_id:1,name:"Mattias Dahlgren",url:"https://miun-se.zoom.us/my/dahlgren/"}
];

/* GET users listing. */
router.get('/', function(req, res, next){
  res.send(users);
});

module.exports = router;
```

SPARA!

Provkör med npm start!

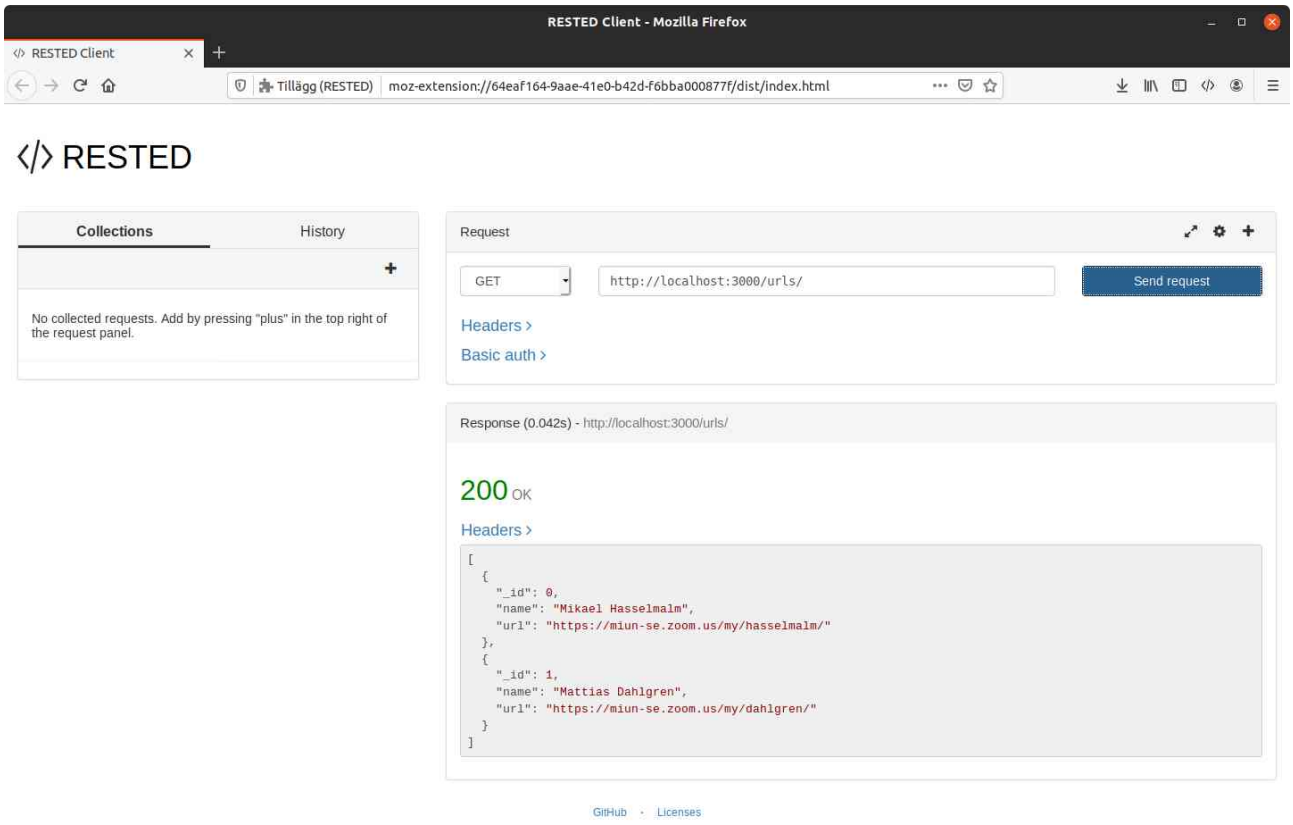
Start en webbläsare och ange adress: http://localhost:3000/urls/

Steg 4

För att kunna provköra verben POST, PUT och DELETE måste vi använda ett verktyg för att provköra och felsöka skapade webbtjänster.

POST – Create - Skapa ny
GET – Read - Hämta data
PUT – Update - Lägg till /Uppdatera
DELETE – Delete - Ta bort data

Ett exempel på ett sådant verktyg är tillägget RESTED för webbläsaren FIREFOX.



Steg 5

Implementera de HTTPverb vi behöver. Detta sker som tidigare i filen zoomba/routes/urls.js

```
var express = require('express');
var router = express.Router();

var users = [
  {_id:0,name:"Mikael Hasselmalm",url:"https://miun-se.zoom.us/my/hasselmalm/"},
  {_id:1,name:"Mattias Dahlgren",url:"https://miun-se.zoom.us/my/dahlgren/"}
];

/*********************************************************
*  GET user listing.
*********************************************************/
router.get('/', function(req, res, next) {
  res.send(users);
});

/*********************************************************
* POST/Create new user
*********************************************************/
router.post('/', function(req, res, next){
  // Get current id
  let max = 0;
  users.forEach(function(elem){
    if(elem._id > max) max = elem._id;
  });

  // Create new post
  let newU = {};
  newU._id = max+1;
  newU.name = req.body.name;
  newU.url = req.body.url;

  users.push(newU);

  res.send(newU);
});

/*********************************************************
* DELETE uniq user
*********************************************************/
router.delete('/:id', function(req, res, next){
  var id = req.params.id;
        var del=-1;

        for(var i=0; i < users.length; i++){
    if(users[i]._id == id) del = i; // Find the array index that holds _id = id
    console.log(i);
        }
        if(del>=0) status=users.splice(del, 1); // Delete element and rearrange the array indexes

//      res.contentType('application/json');
        res.send(users);
});

module.exports = router;
```

Steg 6

Som du kanske har sett så laddas all data om vid start av programmet vilket inte är att föredra. Genom att lagra all information i en fil istället så kan vi spara vår data mellan omstarter av applikationen.

Skapa en ny katalog med namn json. (zoomba/json).

Skapa i katalogen en fil med namn urls.json.
Innehållet i denna fil ska vara []

Programkoden i filen zoomba/router/urls.js måste ändras för att arbeta mot fil istället för minne.

```
.
:
var router = express.Router();

const fs = require('fs');
let rawdata = fs.readFileSync('json/urls.json'); // Wait for the data to be retrieved

let users = JSON.parse(rawdata);
.
:
/**********************************************************
* POST/Create new user
**********************************************************/
router.post('/', function(req, res, next){
.
:
  users.push(newU);
  /*********************************************
  * Update user file
  *********************************************/
  fs.writeFile('json/urls.json', JSON.stringify(users), 'utf8', function(err) {
      if (err) throw "Couldn't write to file!"+err;
    }
  );
  res.send(newU);
});

/**********************************************************
* DELETE uniq user
**********************************************************/
router.delete('/:id', function(req, res, next){
  if(del>=0) status=users.splice(del, 1); // Delete element and rearrange the array indexes

//  res.contentType('application/json');
  fs.writeFile('json/urls.json', JSON.stringify(users), 'utf8', function(err) {
    if (err) throw "Couldn't write to file!"+err;
  }
  );
```

```
  res.send(users);
});
module.exports = router;
```

Steg 7

Under detta moment ska vi skapa ett enklare webbgränssnitt som arbetar mot vår RESTwebbtjänst
Nedan ser vi en förklaring av vår webbtjänst ”urls” och hur vi använder denna.

VERB	REQUEST	BODY	ANSWER
POST	http://localhost:3000/urls/	JSON data name url	JSON object defining created data { _id : int , name : string , url : string }
GET	http://localhost:3000/urls/		JSON array showing all defined objects
DELETE	http://localhost:3000/urls/:id		JSON array showing all defined objects

För att skapa de HTMLsidor som behövs för att slutföra detta projekt flyttar vi till katalogen med namnet public.
I denna katalog ska vi skapa en fil som heter index.html

Vi har nu en fungerande lösning till problemet!