Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий комплекс
«Інститут прикладного системного аналізу»

Відділення другої вищої та післядипломної освіти

ЛАБОРАТОРНА РОБОТА №1

Виконала: студентка 4-го курсу
групи ІС-зп62
Слобода К.О.

Київ – 2019

Мета роботи: освоїти основні поняття технологій програмування за допомогою використання node.js.

За допомогою node.js та фреймворку express було створено сайт «Сімейне дерево» (**папка lab2**), з декількома сторінками:

server.js

```
"use strict";
const ch              = require('./helpers/cheerio_helper');
const express         = require('express');
const MongoClient = require('mongodb').MongoClient;
const bodyParser    = require('body-parser');
const db              = require('./config/db');
const app             = express();
const port            = 3000;

app.use(bodyParser.urlencoded({ extended: true }));
app.set('view engine', 'ejs');
app.use('/public', express.static('public'));

MongoClient.connect(db.url, { useNewUrlParser: true }, (err, client) => {
  if (err) throw err;
  let db = client.db('familyTreeApp');
  require('./routes')(app, db, ch);
  app.listen(port, () => {
    console.log('We are live on ' + port);
  });
});
```

routes/index.js

```
const treesRoutes = require('./trees_routes');
const personsRoutes = require('./persons_routes');
const articlesRoutes = require('./articles_routes');
module.exports = function(app, db, ch) {
  treesRoutes(app, db, ch);
  personsRoutes(app, db, ch);
  articlesRoutes(app, db, ch);
};
```

routes/trees_routes.js

```
"use strict";
let ObjectID = require('mongodb').ObjectID;
module.exports = function(app, db, ch) {
  // Get main page
  app.get('/', (req, res) => {
    res.render('index');
  });

  // Create tree
```

```
app.post('/', (req, res) => {
  if (!req.body) return res.sendStatus(400);
  const tree = { name: req.body.name };
  db.collection('trees').insertOne(tree, (err, result) => {
    if (err) throw err;
    res.redirect('/trees');
  });
});

// Get trees
app.get('/trees', (req, res) => {
  db.collection('trees').find({}).toArray((err, items) => {
    if (err) throw err;
    res.render('trees', { items: items });
  });
});

// Get tree
app.get('/tree/:id', (req, res) => {
  const tdetails = { '_id': new ObjectID(req.params.id) };
  const pdetails = { 'treeId': req.params.id };
  db.collection('trees').findOne(tdetails, (err, tree) => {
    if (err) throw err;
    db.collection('persons').find(pdetails).toArray((err, persons) => {
      if (err) throw err;
      persons.sort(function(p1, p2) { return new Date(p1.dateOfBirth) -
                                    new Date(p2.dateOfBirth); });
      res.render('tree', { tree: tree, persons: persons });
    });
  });
});

// Delete tree
app.post('/delete-tree', (req, res) => {
  const id = req.body.id;
  const tdetails = { '_id': new ObjectID(id) };
  const pdetails = { 'treeId': new ObjectID(id) };
  db.collection('persons').deleteMany(pdetails, (err, item) => {
    if (err) throw err;
  });
  db.collection('trees').deleteOne(tdetails, (err, item) => {
    if (err) throw err;
    res.json(id);
  });
});

// Get tree for Edit page
app.get('/edit-tree/:id', (req, res) => {
  const details = { '_id': new ObjectID(req.params.id) };
```

```
    db.collection('trees').findOne(details, (err, tree) => {
      if (err) throw err;
      res.render('tree-edit', { tree: tree });
    });
  });

  // Update tree
  app.post('/edit-tree/:id', (req, res) => {
    const id = req.params.id;
    const details = { '_id': new ObjectID(id) };
    const tree = { name: req.body.name };
    db.collection('trees').update(details, tree, (err, result) => {
      if (err) throw err;
      res.redirect('/trees');
    });
  });
};
```

routes/persons_routes.js

```
"use strict";
let ObjectID = require('mongodb').ObjectID;
module.exports = function(app, db, ch) {
  // Get person
  app.get('/person/:id', (req, res) => {
    const wdetails = { 'treeId': req.query.treeId, 'gender': 1 }
    const mdetails = { 'treeId': req.query.treeId, 'gender': 0 }
    db.collection('persons').find(wdetails).toArray((err, women) => {
      if (err) throw err;
      db.collection('persons').find(mdetails).toArray((err, men) => {
        if (err) throw err;

        women = women.filter((n) => { return n._id != req.params.id; });
        men = men.filter((n) => { return n._id != req.params.id; });
        women.unshift({ _id: 0, firstName: "None" })
        men.unshift({ _id: 0, firstName: "None" })

        if (req.params.id == 0) {
          res.render('person', { person: { _id: 0, treeId: req.query.treeId },
                                 women: women,
                                 men: men });
        } else {
          const details = { '_id': new ObjectID(req.params.id) };
          db.collection('persons').findOne(details, (err, person) => {
            if (err) throw err;
            res.render('person', { person: person, women: women, men: men });
          });
        }
      });
    });
```

```
});

// Update person
app.post('/edit-person/:id', (req, res) => {
  const person = { firstName: req.body.firstName,
                   lastName: req.body.lastName,
                   middleName: req.body.middleName,
                   dateOfBirth: req.body.dateOfBirth,
                   dateOfDeath: req.body.dateOfDeath,
                   gender: req.body.gender,
                   motherId: req.body.motherId,
                   fatherId: req.body.fatherId,
                   treeId: req.body.treeId };

  if (req.params.id == 0) {
    db.collection('persons').insertOne(person, (err, result) => {
      if (err) throw err;
      res.redirect('/tree/' + req.body.treeId);
    });
  } else {
    const id = req.params.id;
    const details = { '_id': new ObjectID(id) };
    db.collection('persons').update(details, person, (err, result) => {
      if (err) throw err;
      res.redirect('/tree/' + req.body.treeId);
    });
  }
});

// Delete person
app.get('/delete-person/:id/:treeId', (req, res) => {
  const id = req.params.id;
  const treeId = req.params.treeId;
  const details = { '_id': new ObjectID(id) };
  db.collection('persons').deleteMany(details, (err, item) => {
    if (err) throw err;
    res.redirect('/tree/' + treeId);
  });
});

// Get person info
app.get('/get-person-info/:id', (req, res) => {
  const details = { '_id': new ObjectID(req.params.id) };
  db.collection('persons').findOne(details, (err, person) => {
    if (err) throw err;

    let searchTerm = encodeURI(person.firstName + "+" + person.lastName);
    let searchUrl = 'https://www.google.com/search?q=' + searchTerm + '&tbm=nws';
```

```
    ch.getPersonInfo(searchUrl).then(result => {
      res.render('person-info', { links: result });
    }).catch(e => {
      res.render('person-info', { links: [{ text: e }] });
    });
  });
 });
};
```

Висновки:

Під час виконання лабораторної роботи був створений сайт «Сімейне дерево», з можливістю створити своє сімейне дерево та додати туди членів родини.