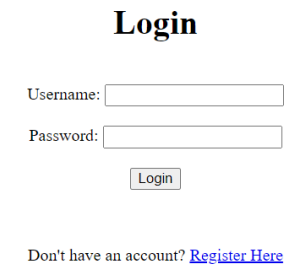
Автентикација

Домашна 2, Информациска Безбедност

Овој проект претставува едноставен автентикациски систем за најава и креирање кориснички профил на веб страна. Се состои од HTML датотеки (страна за најава, регистрација, валидација и главна по успешна најава), JSON датотека (‘databasesim.json’) со зачуваните кориснички податоци и Javascript датотека (‘server.js’) која ги запишува корисничките податоци во JSON датотеката по успешна регистрација и врши автентикација при обид на најава. За функционалноста на скриптата се користи локален сервер преку Node.js .

HTML страни

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
</head>
<body>
<center>
    <br/><br/><br/><br/><br/>
    <h1>Login</h1>
    <br/>
    <form action="/login" method="post">
      <label for="loginUsername">Username:</label>
      <input type="text" id="loginUsername" name="username" required>
      <br/><br/>
      <label for="loginPassword">Password:</label>
      <input type="password" id="loginPassword" name="password" required>
      <br/><br/>
      <button type="submit">Login</button>
    </form>
    <br/><br/>
    <p>Don't have an account? <a href="/register">Register Here</a></p>
</center>
</body>
</html>
```




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
</head>
<body>
<center>
    <br/><br/><br/><br/><br/>
    <h1>Register</h1>
    <br/>
    <form action="/register" method="post">
      <label for="registerUsername">Username:</label>
      <input type="text" id="registerUsername" name="username" required>
      <br/><br/>
      <label for="registerEmail">Email:</label>
      <input type="email" id="registerEmail" name="email" required>
      <br/><br/>
      <label for="registerPassword">Password:</label>
      <input type="password" id="registerPassword" name="password" pattern="^(?=.*\d)(?=.*[a-zA-Z]).{8,}$" title="Password must contain at least one digit and one
      <br/><br/>
      <label for="confirmPassword">Confirm Password:</label>
      <input type="password" id="confirmPassword" name="confirmPassword" required>
      <br/><br/>
      <button type="submit">Register</button>
    </form>
    <br/><br/>
    <p>Already have an account? <a href="/login">Login Here</a></p>
</center>
</body>
</html>
```

Register

Username:

Email:

Password:

Confirm Password:

Register

Already have an account? Login Here

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Main</title>
</head>
<body>
<center>
    <br/><br/><br/><br/><br/>
    <h1>Welcome</h1>
    <h3>blah blah blah</h3>
    <br/><br/>
    <a href="/login">Logout</a>
</center>
</body>
</html>
```

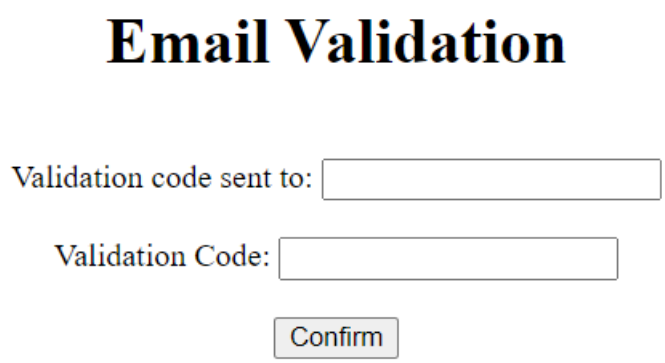
Welcome

blah blah blah

Logout

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Email Validation</title>
</head>
<body>
  <center>
    <br/><br/><br/><br/><br/>
    <h1>Email Validation</h1>
    <br/>
    <form action="/validate" method="post">
      <label for="userEmail">Validation code sent to:</label>
      <input type="text" id="userEmail" name="userEmail" value="" readonly>
      <br/><br/>
      <label for="validationCode">Validation Code:</label>
      <input type="text" id="validationCode" name="validationCode" pattern="[0-9]{12}" title="Please enter a 12-digit validation code" required>
      <br/><br/>
      <button type="submit">Confirm</button>
    </form>
  </center>

  <script>
  <!-- popolnuva readonly poleto so mailot na koj praka kod za validacija -->
    const userEmail = new URLSearchParams(window.location.search).get('email');
    const userEmailField = document.getElementById('userEmail');
    userEmailField.value = userEmail;
    document.getElementById('userEmail').textContent = userEmail;
  </script>
</body>
</html>
```



Скрипта за автентикација

```
const express = require('express');
const session = require('express-session');
const bcrypt = require('bcrypt');
const fs = require('fs');
const path = require('path');

// kreira express aplikacija
const app = express();
app.listen(3000);    // da moze da prima baranja (localhost, porta 3000)
app.use(express.urlencoded({ extended: true }))    // za parsiranje na baranjata
app.use(express.static('public'));   // pristap do html vo 'public' folderot
app.set('view engine', 'ejs');

// go cita json failot so korisnickite informacii
const dataPath = path.join(__dirname, 'databasesim.json');
let userData = [];
let validationCodes = {};

const roles = { owner: 'owner', admin: 'admin', user: 'user', };

try {
  const data = fs.readFileSync(dataPath, 'utf-8');
  userData = JSON.parse(data);
}
catch (err) { console.error(err); }

app.use(session({
  secret: 'mcxedonc43ffdcmef9paa',   // taen kluc
  resave: true,
  saveUninitialized: true
}));

app.get('/login', (req, res) => {res.sendFile(path.join(__dirname, 'public', 'login.html'));});
app.get('/register', (req, res) => {res.sendFile(path.join(__dirname, 'public', 'register.html'));});
app.get('/validate', (req, res) => {res.sendFile(path.join(__dirname, 'public', 'validate.html'));});
app.get('/main', (req, res) => {res.render('main', { user: req.session.user });});
```

```
function delay(req, res, next) {
  setTimeout(next, 3000);
}

// funkcionalnost na register.html
app.post('/register', delay, (req, res) => {
  const { username, email, password, confirmPassword } = req.body;

  if (password !== confirmPassword) {
    res.send("Passwords do not match. Please make sure both passwords are the same.");
    return;
  }

  // dali username ili email veke postojat
  const userExists = userData.some((user) => user.username === username);
  const emailExists = userData.some((user) => user.email === email);

  if (userExists) res.send('Username already exists. Please choose another.');
  else if (emailExists) res.send('Email already exists. Please choose another.');
  else {
    bcrypt.genSalt(10, (err, salt) => {      // generira salt
      if (err) throw err;
      bcrypt.hash(password, salt, (err, hash) => {     // hashira lozinkata
        if (err) throw err;

        const validationCode = Math.floor(100000000000 + Math.random() * 900000000000);   // generira random 12 cifren kod
        validationCodes[email] = validationCode;
        sendEmail(email, validationCode);

        // zacuvuva informaciite za noviot profil vo userData
        if(username === 'alex') userData.push({ username, email, role: 'owner', password: hash, salt });
        else userData.push({ username, email, role: 'user', password: hash, salt });        // mu dava uloga na noviot korisnik

        res.redirect(`/validate?email=${encodeURIComponent(email)}`);         // go prenesuva korisnikot na validate.html
      });
    });
  }
});
```

```
// validacija na email
app.post('/validate', delay, (req, res) => {
  const { userEmail, validationCode } = req.body;

  if (validationCodes[userEmail] && validationCodes[userEmail] == validationCode) {
    const userIndex = userData.findIndex((user) => user.email === userEmail);
    const user = userData.find((user) => user.email === userEmail);

    if (userIndex !== -1) {
      fs.writeFileSync(dataPath, JSON.stringify(userData, null, 2), 'utf-8');   //ako e uspesna validacijata, gi zapisuva informaciite vo databazata

      req.session.user = { username: user.username, role: user.role };
      res.render('main', { user: req.session.user });    //ako e uspesna, zapisuva informaciite vo databazata i prenesuva korisnikot na main.html
    }
    else res.send('Validation failed. Please try again.');
  }
  else res.send('Validation failed. Please try again.');
});

// simulacija na prakanje verifikaciski kod
function sendEmail(email, validationCode) {
  console.log(`Sending validation email to ${email} with code: ${validationCode}`);
}

// funkcionalnost na login.html
app.post('/login', delay, (req, res) => {
  const { username, password } = req.body;

  //dali vnesenite username i password odgovaraat na postoecki profil
  const user = userData.find((u) => u.username === username);
  if (user) {
    bcrypt.compare(password, user.password, (err, result) => {
      if (result) {
          req.session.user = { username, role: user.role };
          res.render('main', { user: req.session.user });    // ako da, go prenesuva korisnikot na main.html
      }
      else res.send('Incorrect username or password. Please try again.');
    });
  }
  else res.send('Incorrect username or password. Please try again.');
});
```

Овој код е пример на Node.js апликација користејќи Express framework за креирање сервер и bcrypt библиотеката за хаширање и проверка на лозинки. Кодот е дел од систем за регистрација и најава на корисници, користејќи JSON датотека како база на податоци.

Прво ги вчитува потребните библиотеки:

- express: За креирање на веб сервер и обработка на HTTP барања;
- express-session: Ги зачувува податоците за сесијата на страната на серверот;
- bcrypt: За хаширање на лозинки и проверка на истите;
- fs: За читање и пишување во датотеки;
- path: Помошен модул за работа со патеки.

Потоа креира Express апликација, насочува серверот на порта 3000 за слушање и примање на барања. Ја чита и претвара JSON датотеката со корисничките податоци во објект за понатамошно користење. Ги дефинира рутите до соодветните страни и на крај, им ги дефинира функционалностите.

Рута за регистрација (/register):

1. Се вчитуваат параметрите за корисничкото име, е-маилот и лозинката;
2. Се проверува дали Password и Confirm password се исти;
3. Се проверува дали веќе постои корисник со исто корисничко име или е-маил;
4. Ако е во ред, лозинката се хашира со bcrypt и корисничките податоци (со salt) се додаваат во низата userData, се симулира праќање на маил со валидациски код и корисникот се префрла на validate.html

Рута за валидација (/validate):

1. Се споредуваат внесениот код со валидацискиот пратен на маилот прикажан на емаил страната
2. Ако е во ред, се запишува новорегистрираниот корисник во JSON датотеката и го пренесува на main.html (main.ejs)

Рута за најава (/login):

1. Се вчитуваат параметрите за корисничкото име и лозинката;
2. Се проверува дали постои корисник со исто корисничко име, и ако постои, се споредува лозинката со bcrypt;
3. Ако лозинката е валидна, корисникот се префрла на main.html (main.ejs)

Содржина на JSON датотеката


```
[
  {
    "username": "alex",
    "email": "alex@gmail.com",
    "role": "owner",
    "password": "$2b$10$Y77rkRgyMDGXp.OPoFcpuO/UBUgTsrXlDhlJ/evoAfwirvmc2FAiS",
    "salt": "$2b$10$Y77rkRgyMDGXp.OPoFcpuO"
  },
  {
    "username": "admin",
    "email": "admin@gmail.com",
    "role": "admin",
    "password": "$2b$10$843S4Ob.2ojfwK.vJQE1puea0ng3Ic78FEWni9J4oAb4gDSUGh4PC",
    "salt": "$2b$10$843S4Ob.2ojfwK.vJQE1pu"
  },
  {
    "username": "user1",
    "email": "user1@gmail.com",
    "role": "user",
    "password": "$2b$10$w/nINOdNdWcTjWShnx/BCec8kQ7MRnHokhAS0qctrwM9FMhqxjrFu",
    "salt": "$2b$10$w/nINOdNdWcTjWShnx/BCe"
  }
]
```