Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение  
высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Отчет

Практическая работа 1.4

«Работа с SQL БД в Web-приложениях Node.js»

Выполнил

студент группы КТсо4-7 Филатова К.Р

Принял

преподаватель ИКТИБ Шкурко А.Н.

Таганрог 2024

1. Цель работы

Данная практическая работа направлена на изучение и практическое использование базовых методов доступа к БД использующих SQL в web-приложениях Node.js.

1. Задача

Необходимо реализовать простое приложение со списком дел. Приложение должно иметь следующую функциональность:

- отображение дел в виде двух списков

- запланированные и выполненные;

- добавление дела в список запланированных при помощи соответствующей формы;

- перенос дела из списка запланированных в список выполненных нажатием кнопки;

- удаление дела из списка выполненных нажатием кнопки.

Списки дел хранятся в БД. У дела не обязательно делать дополнительные атрибуты типа даты или категории. Достаточно будет просто названия. Однако, список дел должен быть доступен разным пользователям, каждый из которых имеет свое имя пользователя и пароль

1. Приложение

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

1. Исходный код

**Index.js**

const express = require('express');

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

const app = express();

const { Client } = require('pg');

var { Liquid } = require('liquidjs');

var engine = new Liquid();

app.engine('liquid', engine.express());

app.set('views', './views');

app.set('view engine', 'liquid');

app.use(express.urlencoded({ extended: true }));

const port = 3000;

const {User, getTask, addTask, completeTask, deleteTask } = require('./bd');

app.set('trust proxy', 1) // trust first proxy

  app.use(session({

    secret: '12',

    resave: false,

    saveUninitialized: true

  }));

const client = new Client({

    user: 'postgres',

    password: '1',

    database: 'sql'

});

client.connect();

function redirectLogin(req, res, next) {

    if (!req.session.userId) {

      res.redirect('/login');

    } else {

      next();

    }

  }

app.get('/', (req, res) => {

    res.render('login');

});

app.post('/login', async (req, res) => {

    const { username, password } = req.body;

    const user = await User.findOne({ where: { username, password } });

    if (user) {

        req.session.userId = user.id;

        res.redirect('/tasks');

        console.log('14')

    } else {

        res.redirect('/');

    }

});

app.post('/logout', (req, res) => {

    req.session.destroy();

    res.redirect('/');

});

app.get('/tasks', redirectLogin, async (req, res) => {

    const userId = req.session.userId;

    const tasks = await getTask(userId);

    const completedTasks = tasks.filter(task => task.completed);

    const plannedTasks = tasks.filter(task => !task.completed);

    res.render('tasks', { tasks, completedTasks, plannedTasks });

});

app.post('/addTask', redirectLogin, async (req, res) => {

    const { name } = req.body;

    const userId = req.session.userId;

    await addTask(name, userId);

    res.redirect('/tasks');

});

app.post('/completeTask/:taskId', redirectLogin, async (req, res) => {

    const { taskId } = req.params;

    await completeTask(taskId);

    res.redirect('/tasks');

});

app.post('/deleteTask/:taskId', redirectLogin, async (req, res) => {

    const { taskId } = req.params;

    await deleteTask(taskId);

    res.redirect('/tasks');

});

app.listen(port, () => {

  console.log(`Server is running on port ${port}`);

});

**Bd.js**

const { Sequelize, DataTypes } = require('sequelize');

const sequelize = new Sequelize('sql', 'postgres', '1', {

    host: 'localhost',

    dialect: 'postgres'

}

);

const Task = sequelize.define('Task', {

    name: {

        type: DataTypes.STRING,

        allowNull: false

    },

    completed: {

        type: DataTypes.BOOLEAN,

        allowNull: false,

        defaultValue: false

    },

    userId: {

        type: DataTypes.INTEGER,

        allowNull: false

    }

}, {

    tableName: 'Task',

    timestamps: false

});

const User = sequelize.define('User', {

    username: DataTypes.STRING,

    password: DataTypes.STRING

});

module.exports = { User, Task };

User.hasMany(Task, {

    foreignKey: 'userId',

    as: 'task'

});

Task.belongsTo(User,{

    foreignKey: 'userId',

});

module.exports.getTask = async (userId) => {

        const tasks = await Task.findAll({

            where: {

                userId

            }

        });

        return tasks;

};

module.exports.addTask = async (name, userId) => {

    await Task.create({

        name,

        userId

    });

};

module.exports.completeTask = async (taskId) => {

    await Task.update({ completed: true }, {

        where: {

            id: taskId

        }

    });

};

module.exports.deleteTask = async (taskId) => {

    await Task.destroy({

        where: {

            id: taskId

        }

    });

};

**Login.liquid**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Login</title>

</head>

<body>

    <h1>Login</h1>

    <form action="/login" method="post">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required>

        <br>

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required>

        <br>

        <button type="submit">Login</button>

    </form>

</body>

</html>

**Tasks.liquid**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Todo List</title>

</head>

<body>

    <h1>Todo List</h1>

    <h2>Planned</h2>

    <ul>

        {% for task in plannedTasks %}

            <li>{{ task.dataValues.name }}

                <form action="/completeTask/{{ task.dataValues.id }}" method="post">

                    <button type="submit">Completed</button>

                </form>

            </li>

        {% endfor %}

    </ul>

    <h2>Completed</h2>

    <ul>

        {% for task in completedTasks %}

            <li>{{ task.dataValues.name }}

                <form action="/deleteTask/{{ task.dataValues.id }}" method="post">

                    <button type="submit">Delete</button>

                </form>

            </li>

        {% endfor %}

    </ul>

    <h2>Add New Task</h2>

    <form action="/addTask" method="post">

        <input type="text" name="name" placeholder="Task name">

        <button>Add</button>

    </form>

    <form action="/logout" method="post">

        <button>Logout</button>

    </form>

</body>

</html>