**Лабораторна робота №4**

Mongoose

**Мета роботи:**

Ознайомитись з mongoose.

**Результат виконання:**

Лістинг 1 - app.js

const mongoose = require("mongoose");

const express = require("express");

const Schema = mongoose.Schema;

const app = express();

const jsonParser = express.json();

const userScheme = new Schema({ name: String, age: Number }, { versionKey: false });

const User = mongoose.model("User", userScheme);

app.use(express.static(\_\_dirname + "/public"));

mongoose.set("strictQuery", false);

mongoose.connect("mongodb://127.0.0.1:27017/usersdb", { useUnifiedTopology: true, useNewUrlParser: true })

    .then(() => {

        app.listen(55555, function () {

            console.log("Сервер очікує підключення...");

        });

    })

    .catch(err => {

        console.error("Error connecting to MongoDB:", err);

    });

app.get("/api/users", function (req, res) {

    User.find({})

        .then(users => {

            res.send(users);

        })

        .catch(err => {

            console.error("Error fetching users:", err);

            res.status(500).send("Internal Server Error");

        });

});

app.get("/api/users/:id", function (req, res) {

    const id = req.params.id;

    User.findOne({ \_id: id })

        .then(user => {

            if (!user) {

                return res.status(404).send("User not found");

            }

            res.send(user);

        })

        .catch(err => {

            console.error("Error fetching user:", err);

            res.status(500).send("Internal Server Error");

        });

});

app.post("/api/users", jsonParser, function (req, res) {

    if (!req.body) return res.sendStatus(400);

    const userName = req.body.name;

    const userAge = req.body.age;

    const user = new User({ name: userName, age: userAge });

    user.save()

        .then(savedUser => {

            res.send(savedUser);

        })

        .catch(err => {

            console.error("Error saving user:", err);

            res.status(500).send("Internal Server Error");

        });

});

app.delete("/api/users/:id", function (req, res) {

    const id = req.params.id;

    User.findByIdAndDelete(id)

        .then(deletedUser => {

            if (!deletedUser) {

                return res.status(404).send("User not found");

            }

            res.send(deletedUser);

        })

        .catch(err => {

            console.error("Error deleting user:", err);

            res.status(500).send("Internal Server Error");

        });

});

app.listen(3001, function () {

    console.log("Сервер очікує підключення...");

});

app.put("/api/users", jsonParser, function (req, res) {

    if (!req.body) return res.sendStatus(400);

    const id = req.body.id;

    const userName = req.body.name;

    const userAge = req.body.age;

    const newUser = { age: userAge, name: userName };

    User.findByIdAndUpdate(id, newUser, { new: true })

        .then(updatedUser => {

            res.send(updatedUser);

        })

        .catch(err => {

            console.error("Error updating user:", err);

            res.status(500).send("Internal Server Error");

        });

});

Лістинг 2 – index.html

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width" />

    <title>User list</title>

    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet" />

</head>

<body>

    <h2>User list</h2>

    <form name="userForm">

        <input type="hidden" name="id" value="0" />

        <div class="form-group">

            <label for="name">Name:</label>

            <input class="form-control" name="name" />

        </div>

        <div class="form-group">

            <label for="age">Age:</label>

            <input class="form-control" name="age" />

        </div>

        <div class="panel-body">

            <button type="submit" class="btn btn-sm btn-primary">Save</button>

            <a id="reset" class="btn btn-sm btn-primary">Reset</a>

        </div>

    </form>

    <table class="table table-condensed table-striped table-bordered">

        <thead>

            <tr>

                <th>Id</th>

                <th>Name</th>

                <th>Age</th>

                <th></th>

            </tr>

        </thead>

        <tbody>

        </tbody>

    </table>

    <script>

        // Получение всех пользователей

        async function GetUsers() {

            // отправляет запрос и получаем ответ

            const response = await fetch("/api/users", {

                method: "GET",

                headers: { "Accept": "application/json" }

            });

            // если запрос прошел нормально

            if (response.ok === true) {

                // получаем данные

                const users = await response.json();

                let rows = document.querySelector("tbody");

                users.forEach(user => {

                    // добавляем полученные элементы в таблицу

                    rows.append(row(user));

                });

            }

        }

        // Получение одного пользователя

        async function GetUser(id) {

            const response = await fetch("/api/users/" + id, {

                method: "GET",

                headers: { "Accept": "application/json" }

            });

            if (response.ok === true) {

                const user = await response.json();

                const form = document.forms["userForm"];

                form.elements["id"].value = user.\_id;

                form.elements["name"].value = user.name;

                form.elements["age"].value = user.age;

            }

        }

        // Добавление пользователя

        async function CreateUser(userName, userAge) {

            const response = await fetch("api/users", {

                method: "POST",

                headers: { "Accept": "application/json", "Content-Type": "application/json" },

                body: JSON.stringify({

                    name: userName,

                    age: parseInt(userAge, 10)

                })

            });

            if (response.ok === true) {

                const user = await response.json();

                reset();

                document.querySelector("tbody").append(row(user));

            }

        }

        // Изменение пользователя

        async function EditUser(userId, userName, userAge) {

            const response = await fetch("api/users", {

                method: "PUT",

                headers: { "Accept": "application/json", "Content-Type": "application/json" },

                body: JSON.stringify({

                    id: userId,

                    name: userName,

                    age: parseInt(userAge, 10)

                })

            });

            if (response.ok === true) {

                const user = await response.json();

                reset();

                document.querySelector("tr[data-rowid='" + user.\_id + "']").replaceWith(row(user));

            }

        }

        // Удаление пользователя

        async function DeleteUser(id) {

            const response = await fetch("/api/users/" + id, {

                method: "DELETE",

                headers: { "Accept": "application/json" }

            });

            if (response.ok === true) {

                const user = await response.json();

                document.querySelector("tr[data-rowid='" + user.\_id + "']").remove();

            }

        }

        // сброс формы

        function reset() {

            const form = document.forms["userForm"];

            form.reset();

            form.elements["id"].value = 0;

        }

        // создание строки для таблицы

        function row(user) {

            const tr = document.createElement("tr");

            tr.setAttribute("data-rowid", user.\_id);

            const idTd = document.createElement("td");

            idTd.append(user.\_id);

            tr.append(idTd);

            const nameTd = document.createElement("td");

            nameTd.append(user.name);

            tr.append(nameTd);

            const ageTd = document.createElement("td");

            ageTd.append(user.age);

            tr.append(ageTd);

            const linksTd = document.createElement("td");

            const editLink = document.createElement("a");

            editLink.setAttribute("data-id", user.\_id);

            editLink.setAttribute("style", "cursor:pointer;padding:15px;");

            editLink.append("Edit");

            editLink.addEventListener("click", e => {

                e.preventDefault();

                GetUser(user.\_id);

            });

            linksTd.append(editLink);

            const removeLink = document.createElement("a");

            removeLink.setAttribute("data-id", user.\_id);

            removeLink.setAttribute("style", "cursor:pointer;padding:15px;");

            removeLink.append("Delete");

            removeLink.addEventListener("click", e => {

                e.preventDefault();

                DeleteUser(user.\_id);

            });

            linksTd.append(removeLink);

            tr.appendChild(linksTd);

            return tr;

        }

        // сброс значений формы

        document.getElementById("reset").click(function (e) {

            e.preventDefault();

            reset();

        })

        // отправка формы

        document.forms["userForm"].addEventListener("submit", e => {

            e.preventDefault();

            const form = document.forms["userForm"];

            const id = form.elements["id"].value;

            const name = form.elements["name"].value;

            const age = form.elements["age"].value;

            if (id == 0)

                CreateUser(name, age);

            else

                EditUser(id, name, age);

        });

        // загрузка пользователей

        GetUsers();

    </script>

</body>

<style>

    body {

        padding: 10px

    }

    .panel-body {

        padding-left: 0px;

    }

</style>

</html>

В результаті отримали додаток, який працює з БД mongoDB

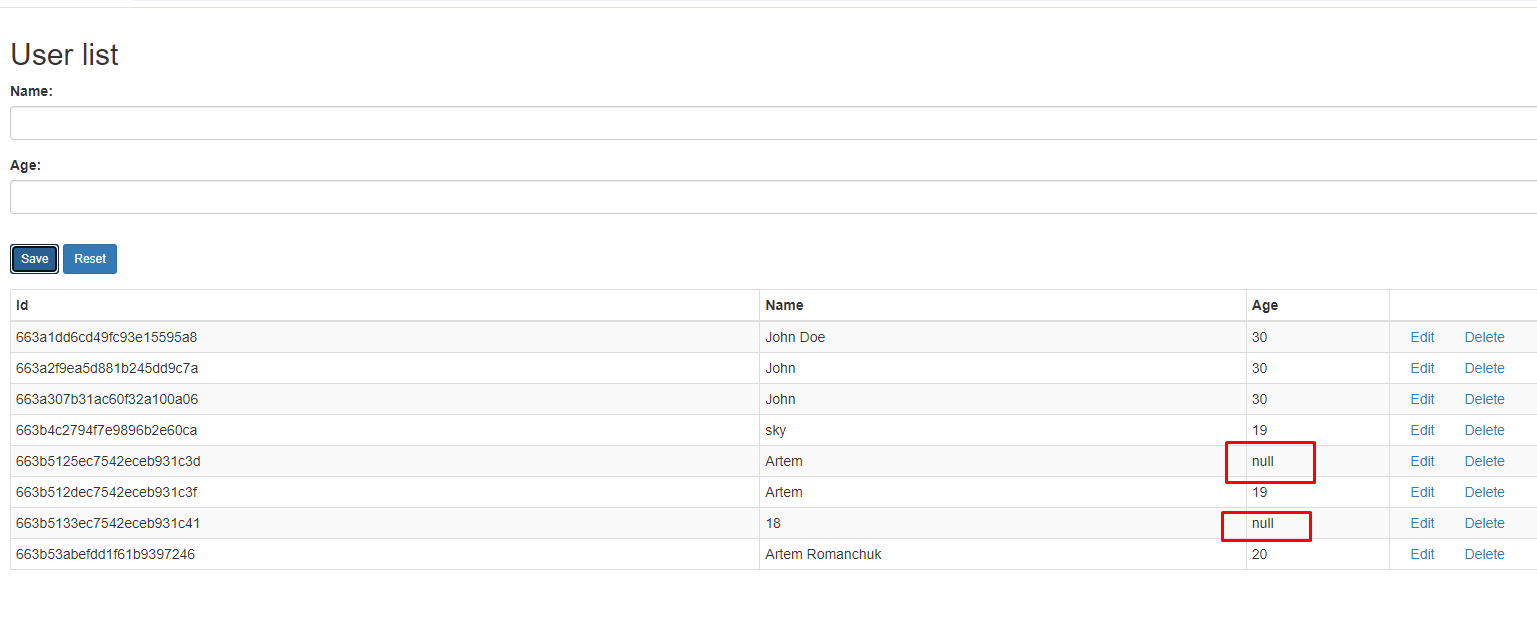


Рисунок 1 – робочий додаток

**Висновок:** Ознайомились з mongoose. Створили додаток для роботи з БД.