**Звіт з лабораторної роботи №4  
з дисципліни “Бази даних та інформаційні системи”**

*Студентки групи МІТ-31  
Баш Анастасії*

Тема. Розробка Інформаційної Системи (ІС) для обробки та візуалізації даних з використанням MEAN стеку

Метою даної лабораторної роботи є Розробка повнофункціональної інформаційної системи для обробки та візуалізації даних в обраній області за допомогою MEAN стеку (MongoDB, Express.js, Angular, Node.js). Система повинна включати роботу з базою даних MongoDB, серверний API на основі Express.js та Node.js, а також клієнтську частину, реалізовану за допомогою Angular.

Бізнес процес обліку членів спортивного клубу:

Клієнти реєструються у системі, обирають абонементи та оформлюються членство у спотивному клубі. Члени клубу можуть записатись на тренування, вибравши тренера, дату та час тренування.

**Хід лабораторної роботи:**

**Завдання №1. Проектування ІС**

Моя інформаційна система є продовженням системи з минулих лабораторних робіт, система обліку абонементів та тренувань в спортивному клубі, тільки тут з в рази спрощеною реалізацією (тільки облік членів клубу, трішки з розширенішими характеристиками про них)

Функціональні вимоги для спрощеної ІС:

1. Реєстрація нових членів у клубі
2. Перегляд списку усіх наявних членів клубу
3. Оновлення інформації про наявних членів клубу, наприклад вік чи номер телефону
4. Анулювання членства або іншими словами видалення члена клубу з ІС

Спрощена ІС використовує базу даних під назвою “Gym”, що складається з однієї колекції ‘Members”. В майбутньому можна додати такі колекції, як “Trainers”, “Services”, “Memberships” та “Trainings” (тобто реалізувати таку ж структуру, що я розробляла у PosgreSQL у минулих лабораторних роботах, тільки тепер замість таблиць ми будемо використовувати колекції та JSON/BSON файли так як MongoDB є інстументом керування нереляційноими базами даних.

**Завдання №2. Розробка серверної частини**

Для початку варно налаштувати базу даних MongoDB. Для цього спочатку прописую структуру колекції, з якою буду подалі працювати.

Колекція “Members” має у собі такі характеристики про клієнтів:

* ID
* Ім’я (Name)
* Прізвище (Surname)
* Стать (Sex)
* Вік (Age)
* Контактний номер телефону (Phone)

Така структура розписана у файлі member.js, що знаходиться у файлі models:

*const* mongoose = require('mongoose');

*const* MemberSchema = new mongoose.Schema({

    id: *Number*,

    name: *String*,

    surname: *String*,

    sex : *String*,

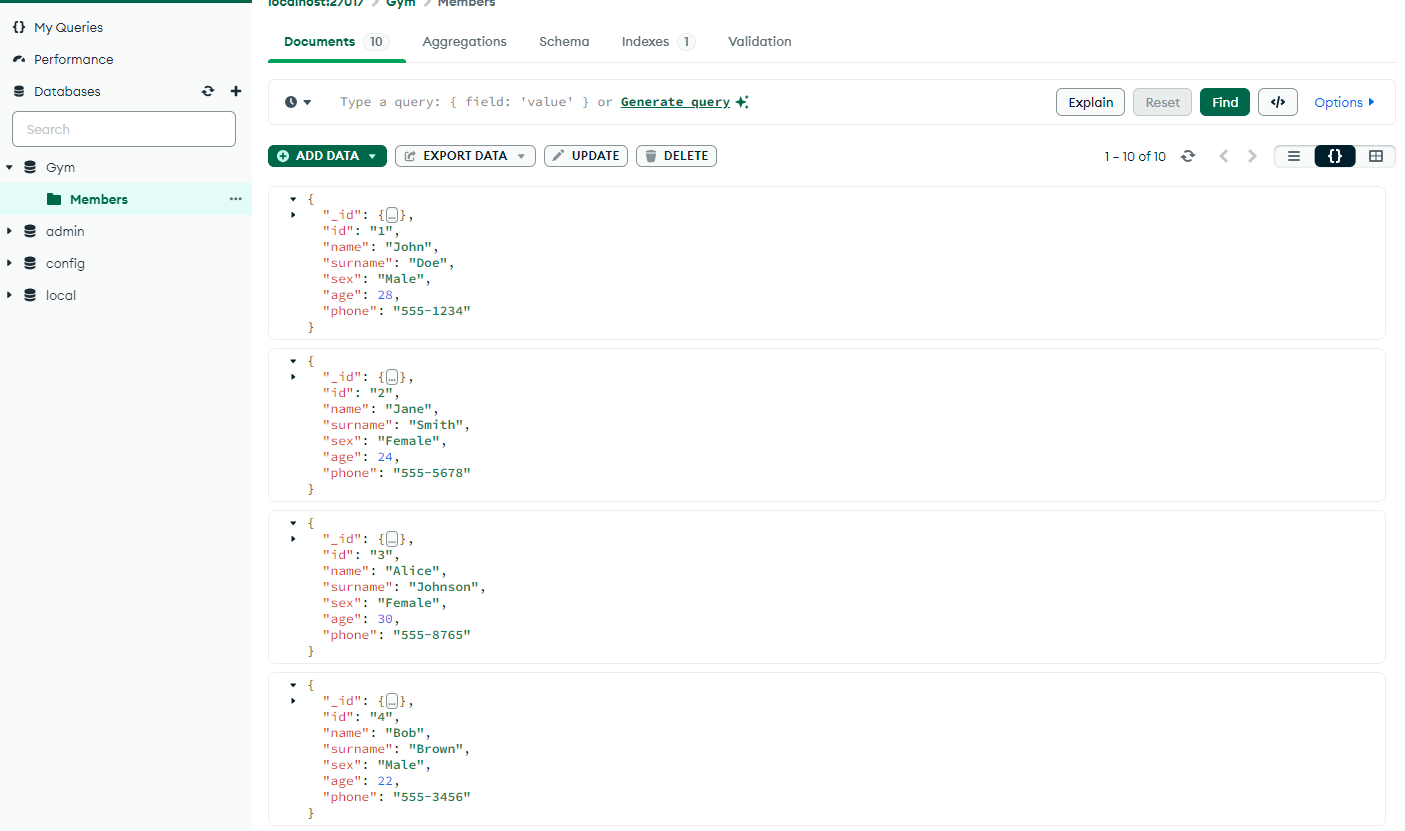
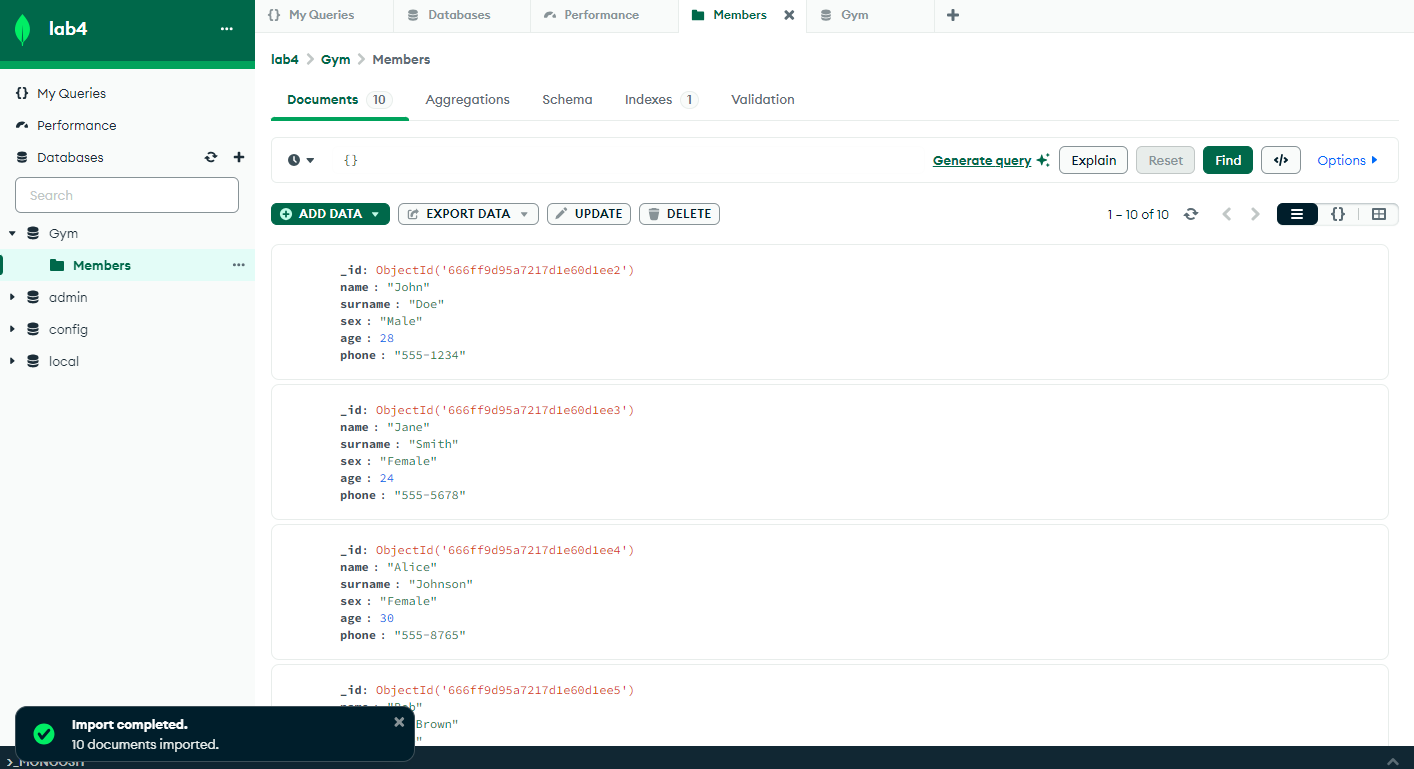
    age: *Number*,

    phone: *String*

});

*module*.*exports* = mongoose.model('Member', MemberSchema);

Заповнюю базу даних, імпортуючи їх з JSON-файлу з згенерованими даними за допомогою mockaroo.com. (10 елементів)



Тепер на початку роботи з базою даних у мене буде 10 стартових об’єктів.

**Запитт до бази даних**

**1)** db.Members.find()

[

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee2"

    },

*"id"*: "1",

*"name"*: "John",

*"surname"*: "Doe",

*"sex"*: "Male",

*"age"*: 28,

*"phone"*: "555-1234"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee3"

    },

*"id"*: "2",

*"name"*: "Jane",

*"surname"*: "Smith",

*"sex"*: "Female",

*"age"*: 24,

*"phone"*: "555-5678"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee4"

    },

*"id"*: "3",

*"name"*: "Alice",

*"surname"*: "Johnson",

*"sex"*: "Female",

*"age"*: 30,

*"phone"*: "555-8765"

  },

db.Members.find({ sex: "Male" })

{

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee2"

    },

*"id"*: "1",

*"name"*: "John",

*"surname"*: "Doe",

*"sex"*: "Male",

*"age"*: 28,

*"phone"*: "555-1234"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee5"

    },

*"id"*: "4",

*"name"*: "Bob",

*"surname"*: "Brown",

*"sex"*: "Male",

*"age"*: 22,

*"phone"*: "555-3456"

  },

db.Members.find({ age: { $gt: 25 } })

[

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee2"

    },

*"id"*: "1",

*"name"*: "John",

*"surname"*: "Doe",

*"sex"*: "Male",

*"age"*: 28,

*"phone"*: "555-1234"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee4"

    },

*"id"*: "3",

*"name"*: "Alice",

*"surname"*: "Johnson",

*"sex"*: "Female",

*"age"*: 30,

*"phone"*: "555-8765"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee6"

    },

*"id"*: "5",

*"name"*: "Charlie",

*"surname"*: "Williams",

*"sex"*: "Male",

*"age"*: 27,

*"phone"*: "555-6543"

  },

db.Members.find({ name: { $regex: "^J" } })

[

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee2"

    },

*"id"*: "1",

*"name"*: "John",

*"surname"*: "Doe",

*"sex"*: "Male",

*"age"*: 28,

*"phone"*: "555-1234"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee3"

    },

*"id"*: "2",

*"name"*: "Jane",

*"surname"*: "Smith",

*"sex"*: "Female",

*"age"*: 24,

*"phone"*: "555-5678"

  }

]

db.Members.find().sort({ age: 1 })

[

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee5"

    },

*"id"*: "4",

*"name"*: "Bob",

*"surname"*: "Brown",

*"sex"*: "Male",

*"age"*: 22,

*"phone"*: "555-3456"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1eeb"

    },

*"id"*: "10",

*"name"*: "Hannah",

*"surname"*: "Rodriguez",

*"sex"*: "Female",

*"age"*: 23,

*"phone"*: "555-3210"

  },

  {

*"\_id"*: {

*"$oid"*: "666ff9d95a7217d1e60d1ee3"

    },

*"id"*: "2",

*"name"*: "Jane",

*"surname"*: "Smith",

*"sex"*: "Female",

*"age"*: 24,

*"phone"*: "555-5678"

  },

db.Members.aggregate([

    { $group: { \_id: "$sex", count: { $sum: 1 } } }

])

[

  {

*"\_id"*: "Male",

*"count"*: 5

  },

  {

*"\_id"*: "Female",

*"count"*: 5

  }

]

db.Members.aggregate([

    { $group: { \_id: "$sex", averageAge: { $avg: "$age" } } }

])

[

  {

*"\_id"*: "Female",

*"averageAge"*: 26.4

  },

  {

*"\_id"*: "Male",

*"averageAge"*: 26.6

  }

]

db.Members.aggregate([

    { $group: { \_id: null, minAge: { $min: "$age" } } }

])

[

  {

*"\_id"*: null,

*"minAge"*: 22

  }

]

Наступник кроком буде розробка самої серверної частини та реалізація API за допомогою Express.js та Node.js для доступу та управління даними (CRUD операції).

Планую реалізувати такі CRUD операції:

1. GET: отриманння усюго списку членів клубу, виведення інофрмації прпо члена клубу за його ID
2. POST: додавання нового члена клубу до БД
3. PUT: оновлення інформаціх про члена клубу за його ID
4. DELETE: видалення члена клубу з БД

Проте спочатку для взаємодії з БД необхідно створити сервер та API для нього.

Для цього створюю файл server.js та налаштовую в ньому створення сервера під портом 4000 та підключення його до моєї локальної БД:

*const* express = require('express');

*const* mongoose = require('mongoose');

*const* path = require('path');

*const* bodyParser = require('body-parser');

*const* api = require('./routes/api')

*const* PORT = 4000;

*const* app = express();

app.use(express.static(path.join(\_\_dirname, 'routes')));

mongoose.connect('mongodb://localhost:27017/Gym', { useNewUrlParser: true, useUnifiedTopology: true })

    .then(() *=>* {

        console.log('MongoDB Connected');

    })

    .catch(*err* *=>* console.error('Error connecting to MongoDB:', *err*));

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

app.use(bodyParser.json());

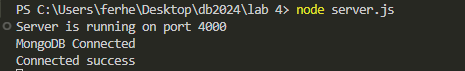
app.use('/api', api);

app.listen(PORT, () *=>* {

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

});

Перевіряю підключення:



Все працює!

Далі створюю папку routes, де у файлі api.js буду прописувати усі API для планових CRUD операцій.

Загальні налаштування та підключення до БД:

*const* express = require('express');

*const* router = express.Router();

*const* mongoose = require('mongoose');

/\* const Member = require('../models/member'); \*/

mongoose.connect('mongodb://localhost:27017/Gym');

mongoose.connection.on('connected', () *=>*{

    console.log("Connected success");

});

mongoose.connection.on('error', (*err*) *=>*{

    console.log("Connection failed" + *err*);

});

**Метод GET**

Код програми для отримання списку усіх членів клубу:

router.get('/members', async (*req*, *res*) *=>* {

    try {

*const* members = await mongoose.connection.db.collection('Members').find({}).toArray();

*res*.json(members);

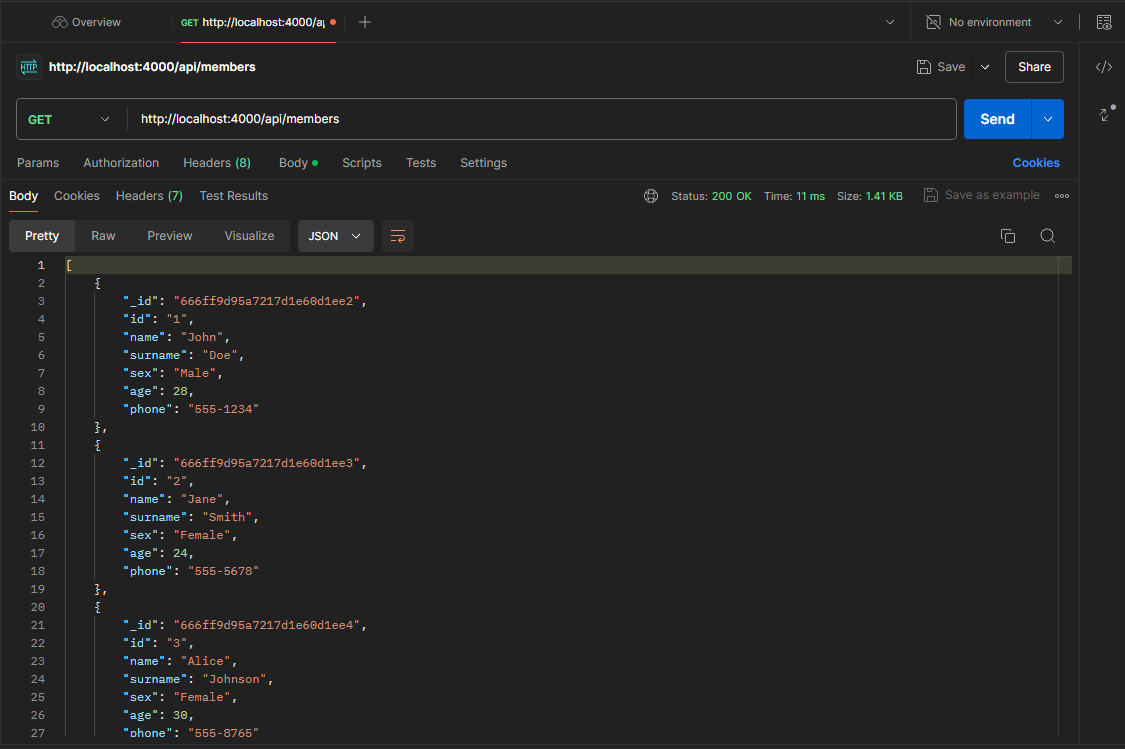
    } catch (err) {

*res*.status(500).json({ message: err.message });

    }

});

Результат робити програми перевіряємо за допомогою Postman:



Все чудово працює.

Далі представлений код програми для виведення інформації про члена клубу за допомогою його ID:

router.get('/members/:id', async (*req*, *res*) *=>* {

    try {

*const* member = await mongoose.connection.db.collection('Members').findOne({ id: (*req*.params.id) });

        if (!member) {

            return *res*.status(404).json({ message: 'Member not found' });

        }

*res*.json(member);

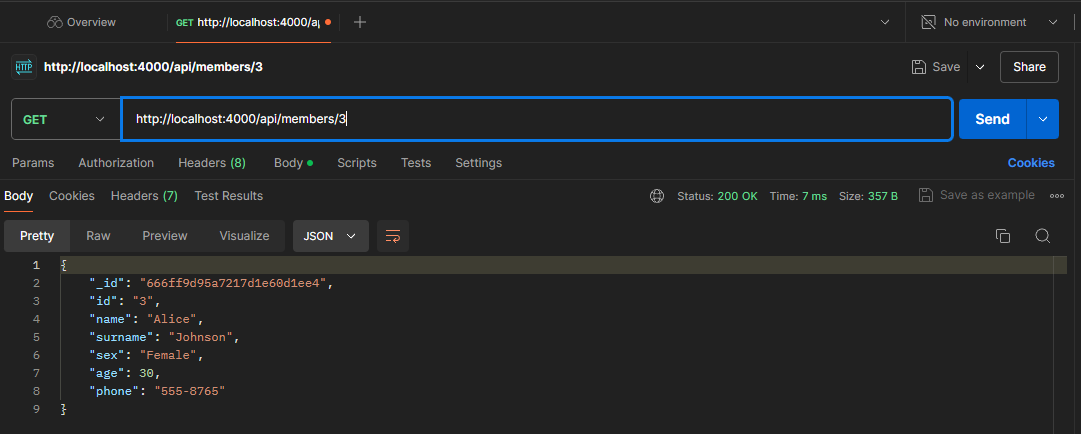
    } catch (err) {

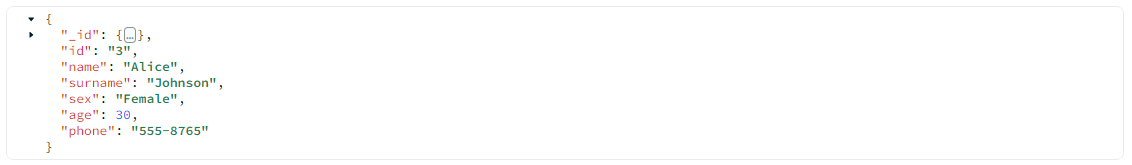
*res*.status(500).json({ message: err.message });

    }

});

Результат:





Працює коректно.

**Метод POST**

Код програми для додавання нового члена клубу:

router.post('/members', async (*req*, *res*) *=>* {

*const* new\_member = *req*.body;

    try {

*const* result = await mongoose.connection.db.collection('Members').insertOne(new\_member);

        if (result.acknowledged === true && result.insertedId) {

*const* insertedMember = await mongoose.connection.db.collection('Members').findOne({ \_id: result.insertedId });

*res*.status(201).json(insertedMember);

        } else {

*res*.status(400).json({ message: 'Failed to insert a new member' });

        }

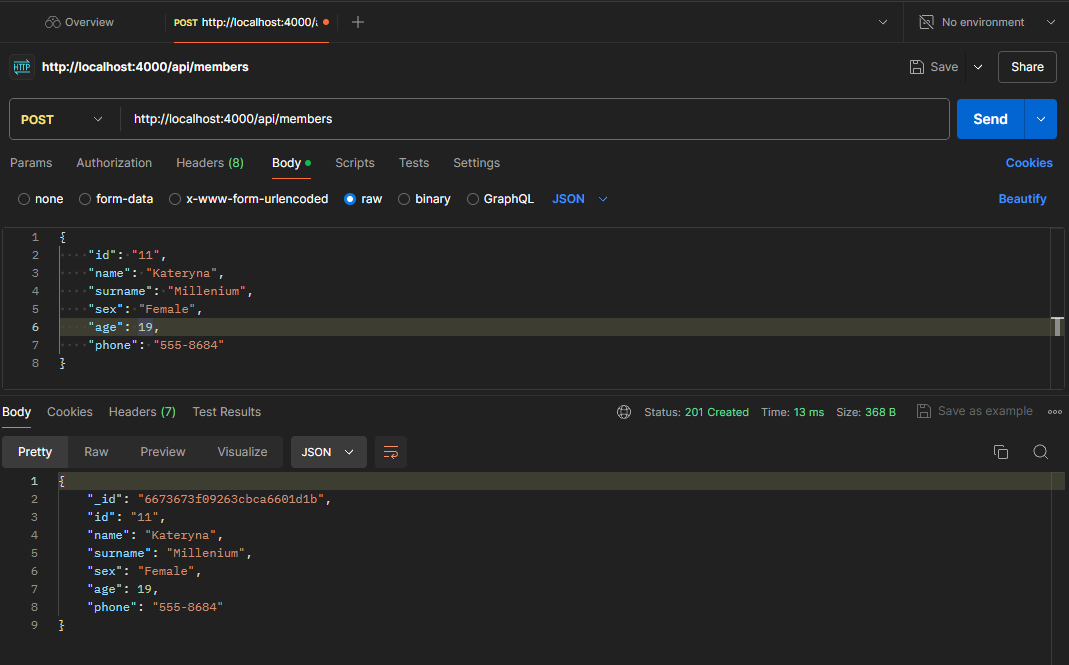
    } catch (err) {

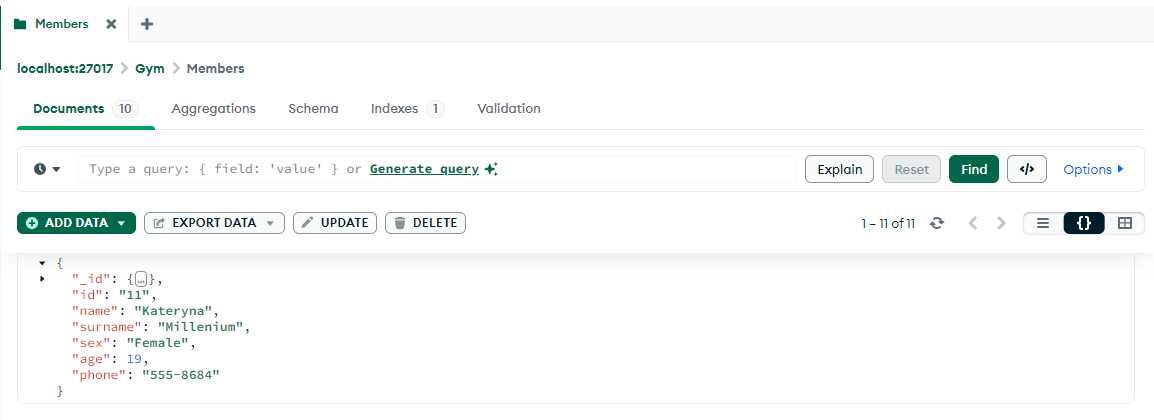
*res*.status(400).json({ message: err.message });

    }

});

Результат роботи програми у постман:





Все правильно!

**Метод PUT**

Код програми для оновлення інформаціх про члена клубу:

router.put('/members/:id', async (*req*, *res*) *=>* {

    try {

*const* result = await mongoose.connection.db.collection('Members').updateOne(

            { id: (*req*.params.id) },

            { $set: *req*.body }

        );

        if (result.modifiedCount === 0) {

            return *res*.status(404).json({ message: 'Member not found' });

        }

*res*.json({ message: 'Member updated' });

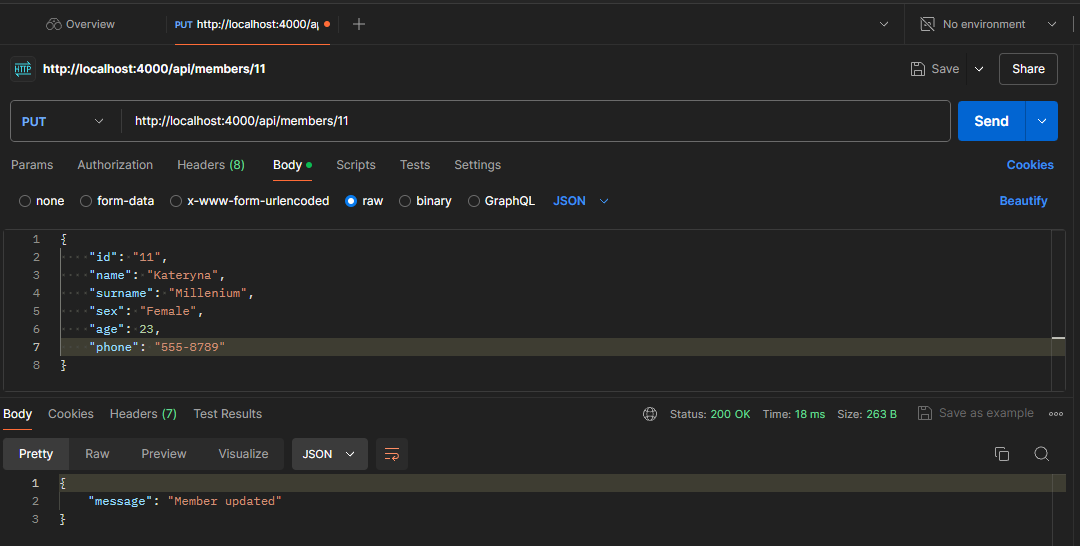
    } catch (err) {

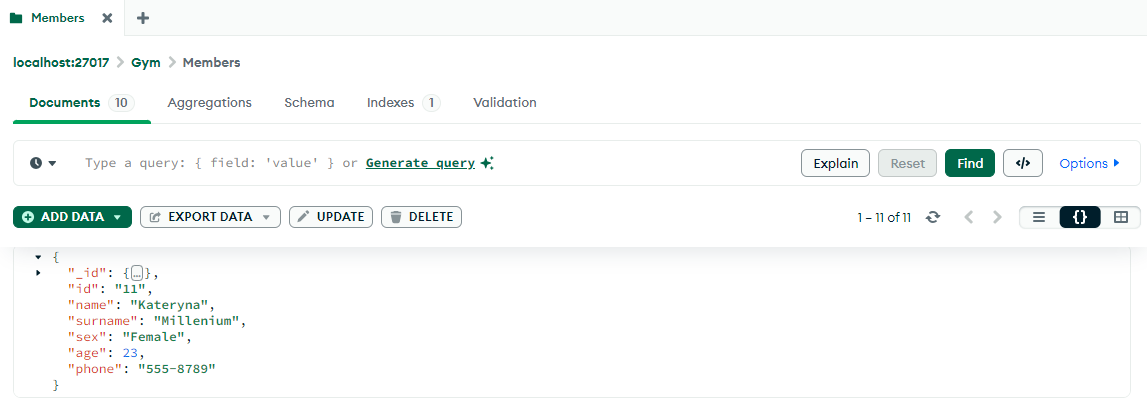
*res*.status(400).json({ message: err.message });

    }

});

Результат роботи програми (оновлюєм інформацію про номер телефону та вік клієнтки; до можна побачити зверху у минулому пункті):



  
Все коректно!

**Метод Delete**

Код програми для видалення члена клубу з БД за його ID:

router.delete('/members/:id', async (*req*, *res*) *=>* {

    try {

*const* result = await mongoose.connection.db.collection('Members').deleteOne({ id: (*req*.params.id) });

        if (result.deletedCount === 0) {

            return *res*.status(404).json({ message: 'Member not found' });

        }

*res*.json({ message: 'Member deleted' });

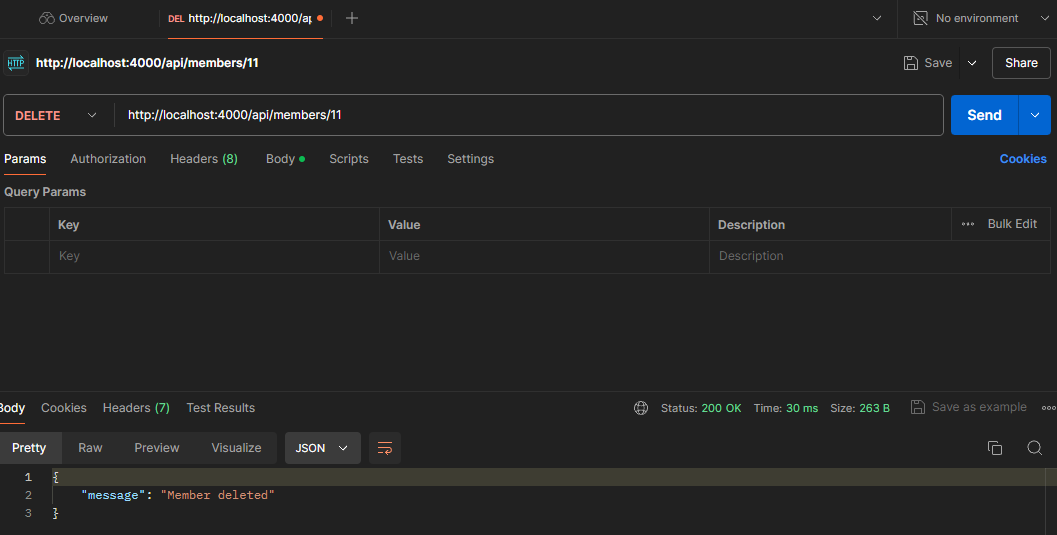
    } catch (err) {

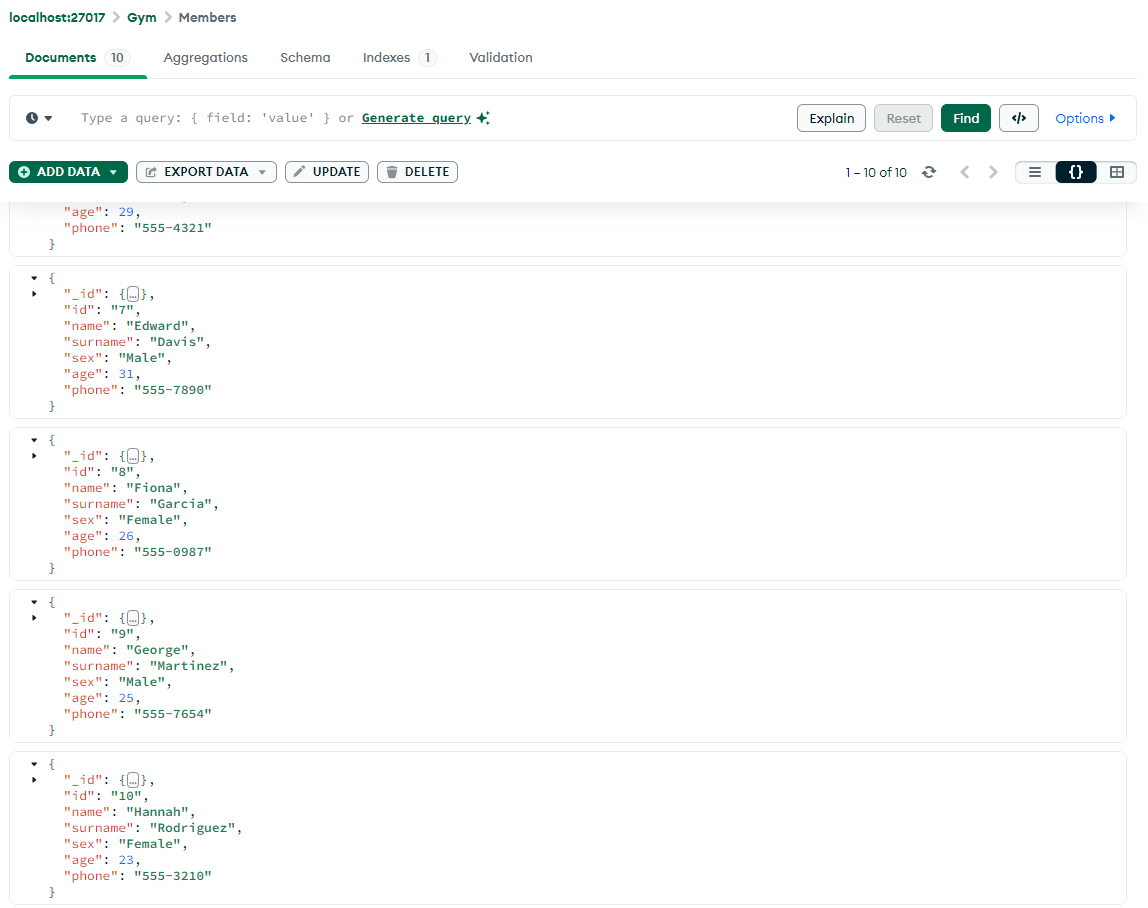
*res*.status(500).json({ message: err.message });

    }

});

Результат видалення:

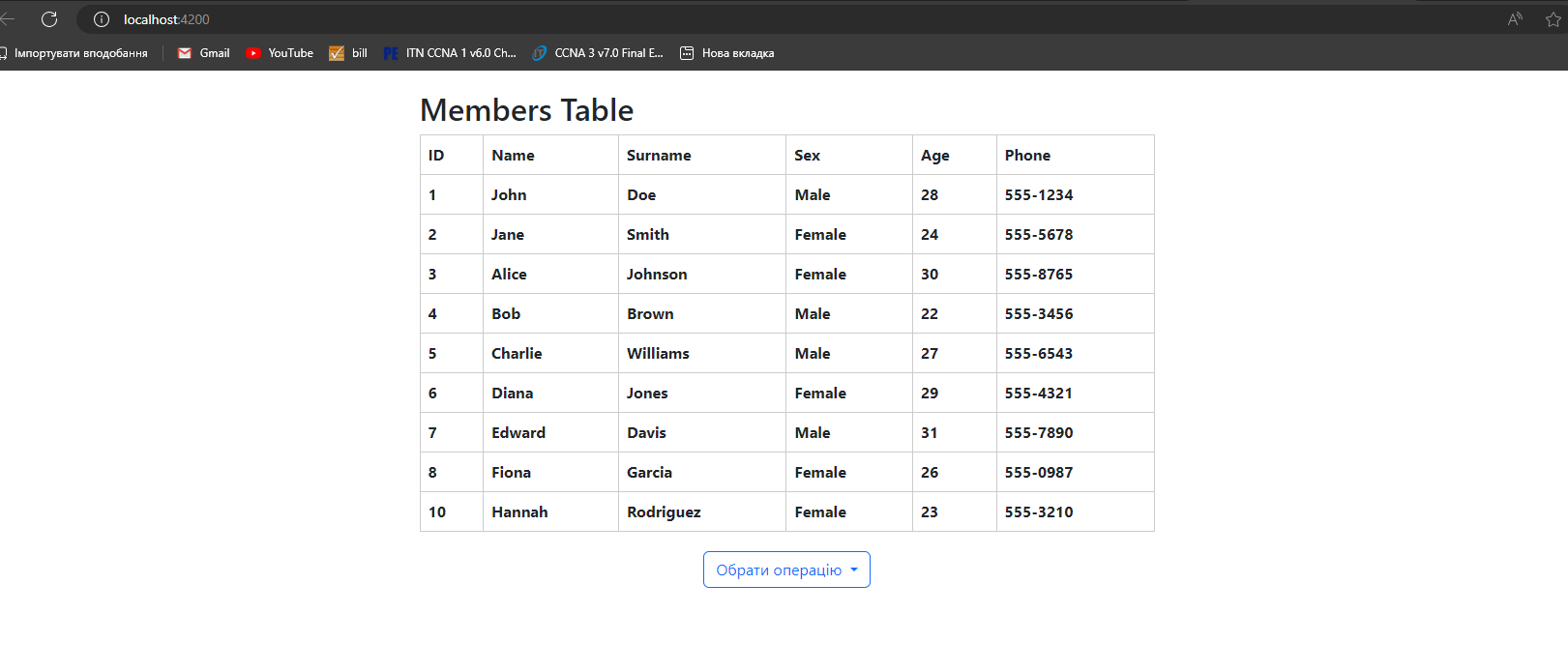




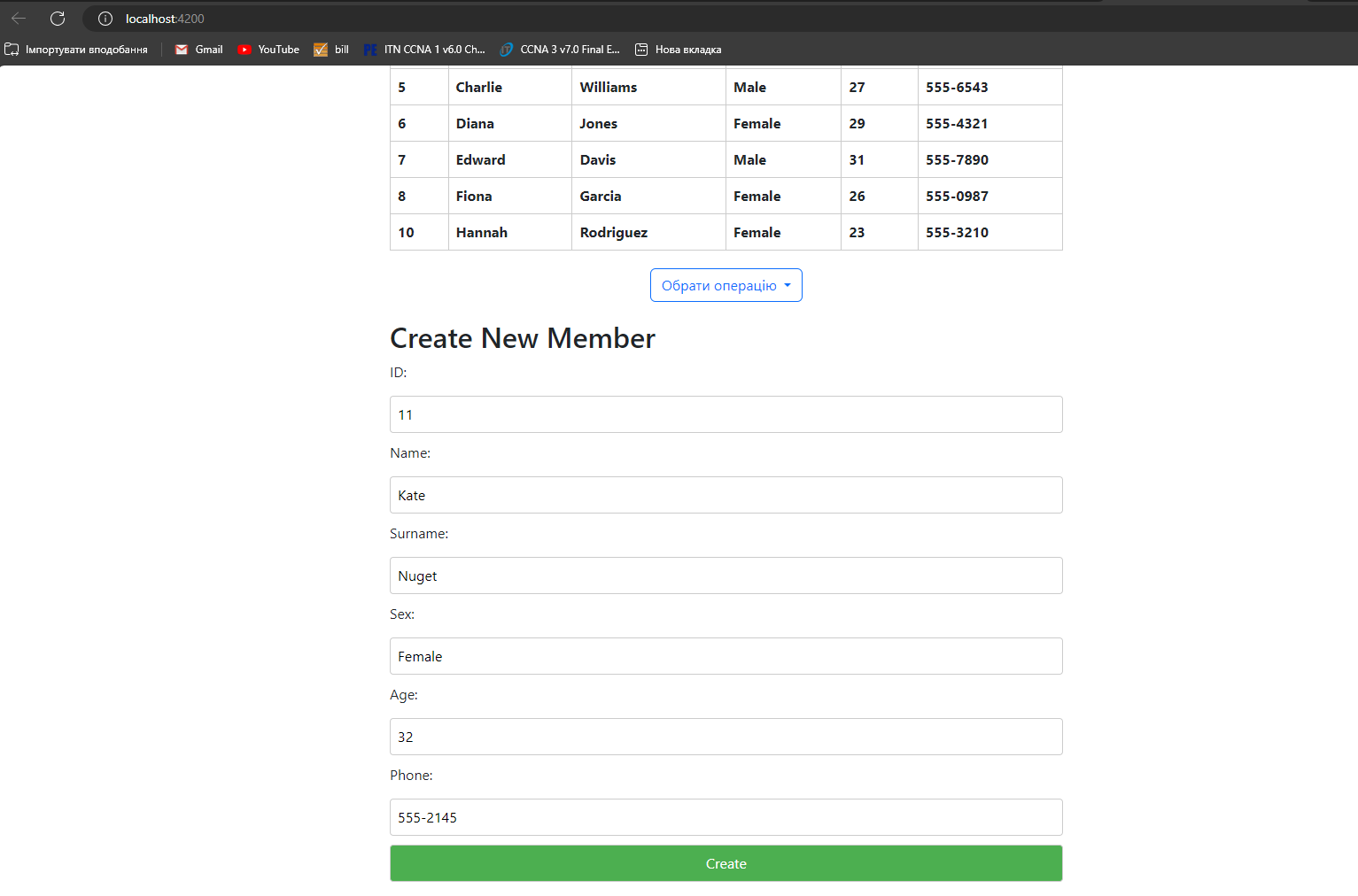
І БД повернулась до свого початкового стану. Все працює.

**Завдання №3. Розробка клієнтської частини**

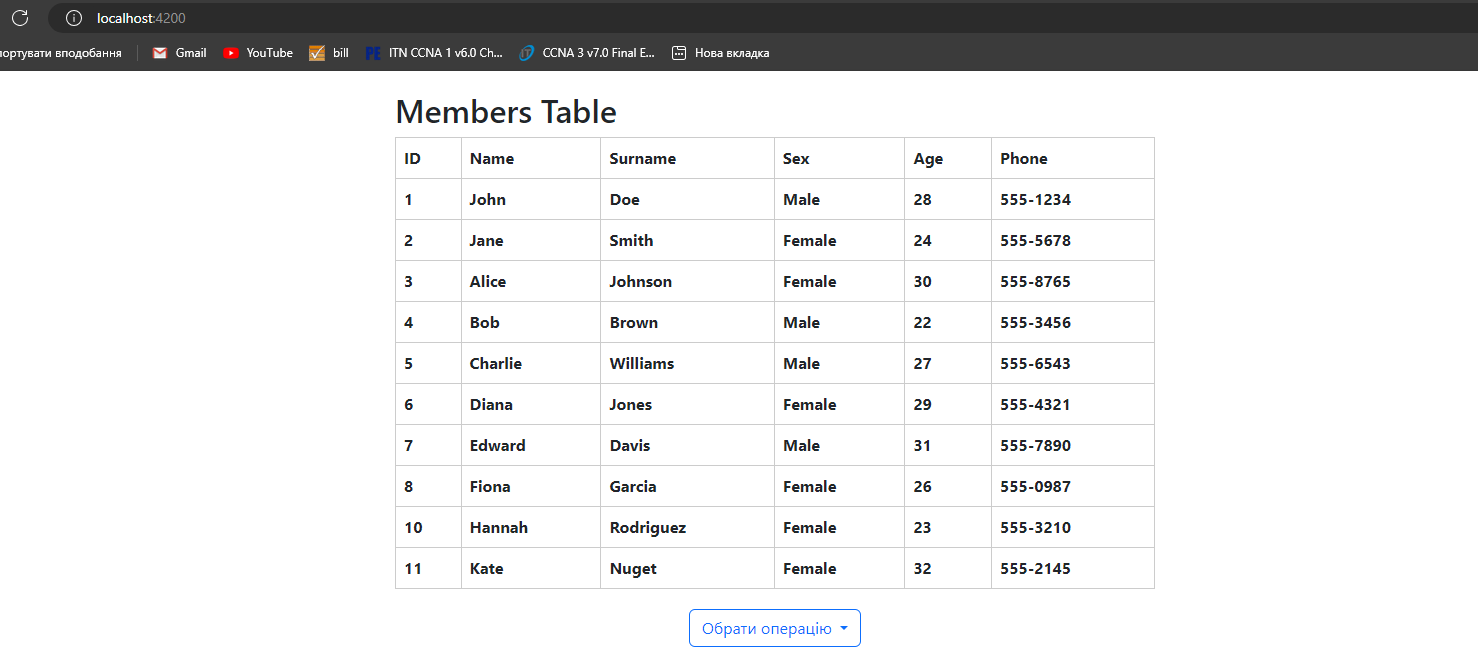
Вигляд початкової сторінки та виведеної БД:



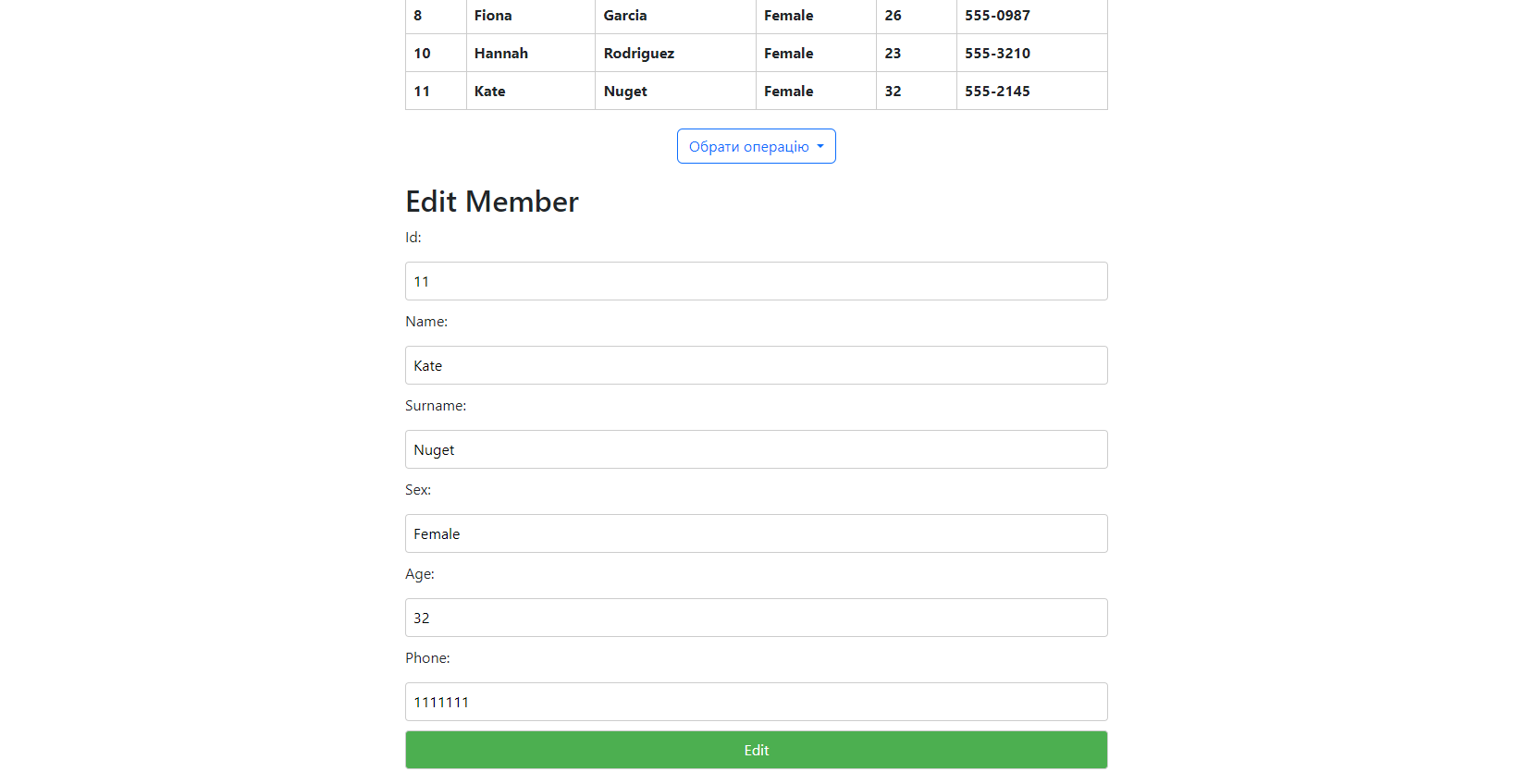
Перевірка роботи додавання нового елемента до БД:



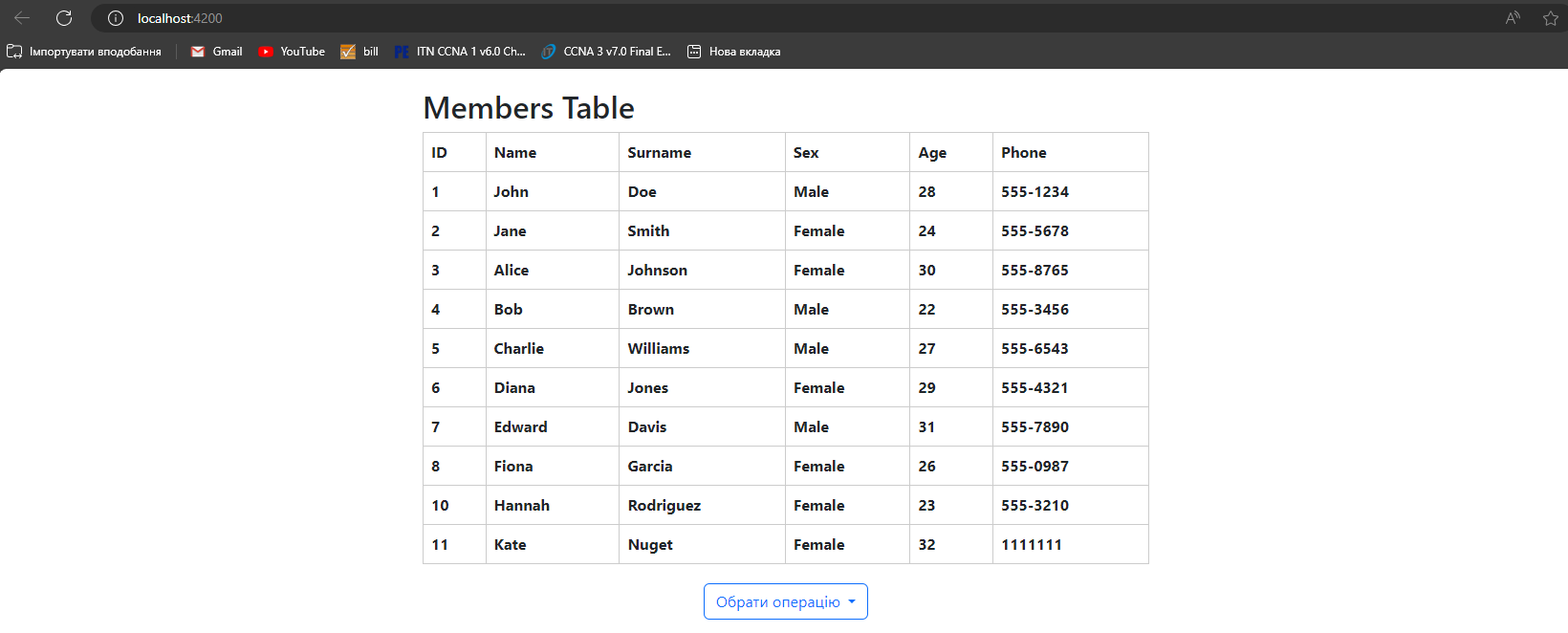
Результат:



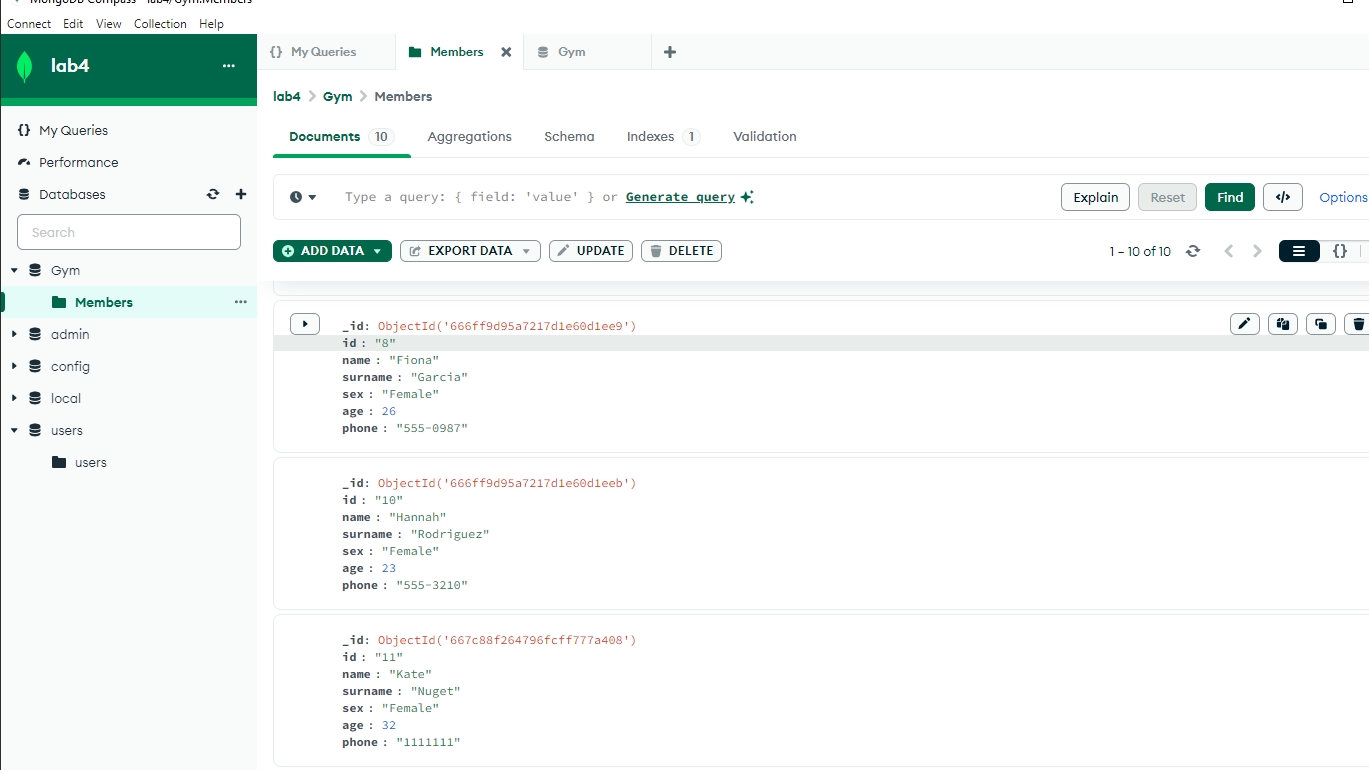
Перевірка редагування елементу:



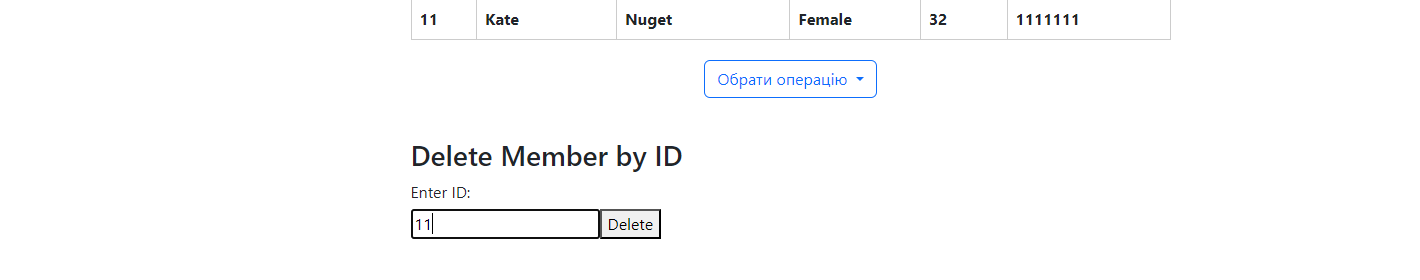
Результат:

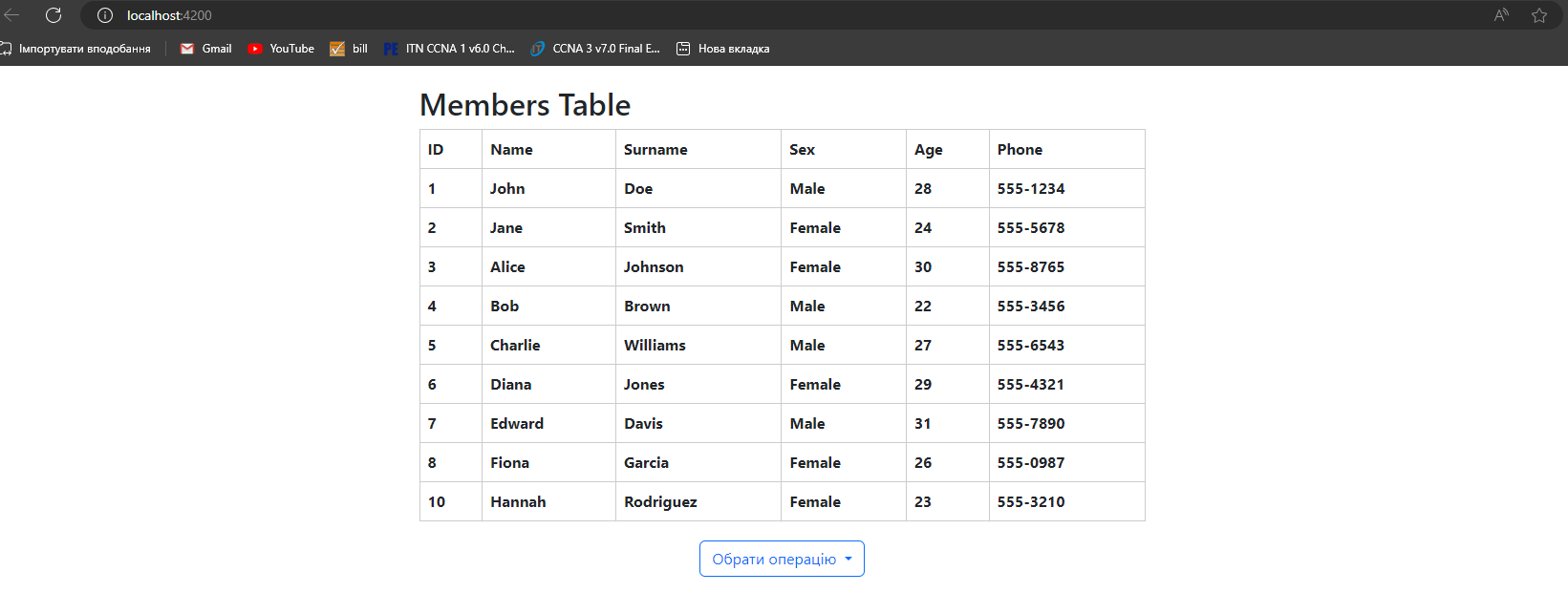


Також переглянемо у MongoDB:



Видалення елементу за його ID:

  
Результат:



**Висновки**

У ході лабораторної роботи було досліджено роботу з базами даних у середовищі MongoDB. Після ознайомлення з MongoDB було спроектовано і розроблено повнофункціональну інформаційну систему для обробки та візуалізації даних за допомогою MEAN стеку.

*Додатки:*

У папці lab4 зберігається повністю розроблена інформаційна система