Міністерство освіти та науки

Харківський національний університет радіоелектроніки

Лабораторна робота №2

З дисципліни «Аналіз та рефакторінг коду»

За темою: «Розробка backend»

Виконав:

ст.гр. ПЗПІ-18-3

Борщов І. С.

Перевірив:

Сокорчук І. П.

Харків 2020

Мета: створити сервер для програмної системи автоматизації догляду за пристарілими людьми. Система повинна отримувати запити за допомогою REST-технологій і мати наступний функціонал: реєстрація користувача, заповнення профілю, збір оброблених данних з IoT пристрою та їх обробкаю, сигналізування у випадку відхилення від норми.

1) Принцип роботи

Сервер працює за принципом мікросервісу з використанням REST-технологій. Запити відправляються з IoT, android пристрою або з web-частини за допомогою HTTP протоколу. Дані, які прикріпляються до запиту знаходяться в JSON форматі. Сервер приймає дані, обробляє їх і виконує певну бізнес логіку. Потім повертає результат роботи назад клієнту. Схему взаємодії сервера з клієнтом зображено на додатку А.

2) Архітектура серверу

Серверна частина написана на мові програмування JavaScript з використанням фреймворку NodeJS, bcrypt для шифрування паролів, joi для валідації даних, jsonwebtoken для jwt авторизації. В якості СУБД було обрано MongoDB. Структуру файлів зображено на рисунку 2.1.

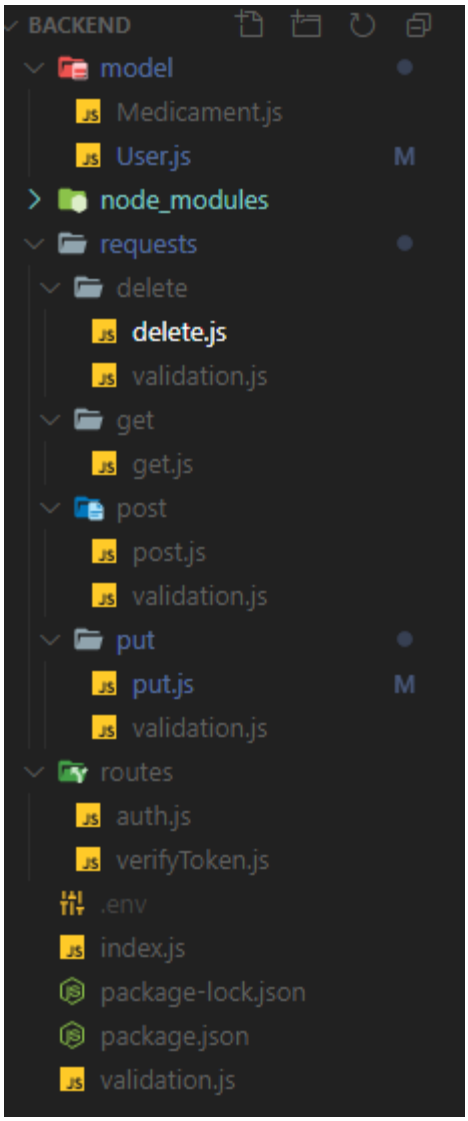


Рисунок 2.1. – Структура файлів серверу

Схему бази даних показано на рисунку 2.2.

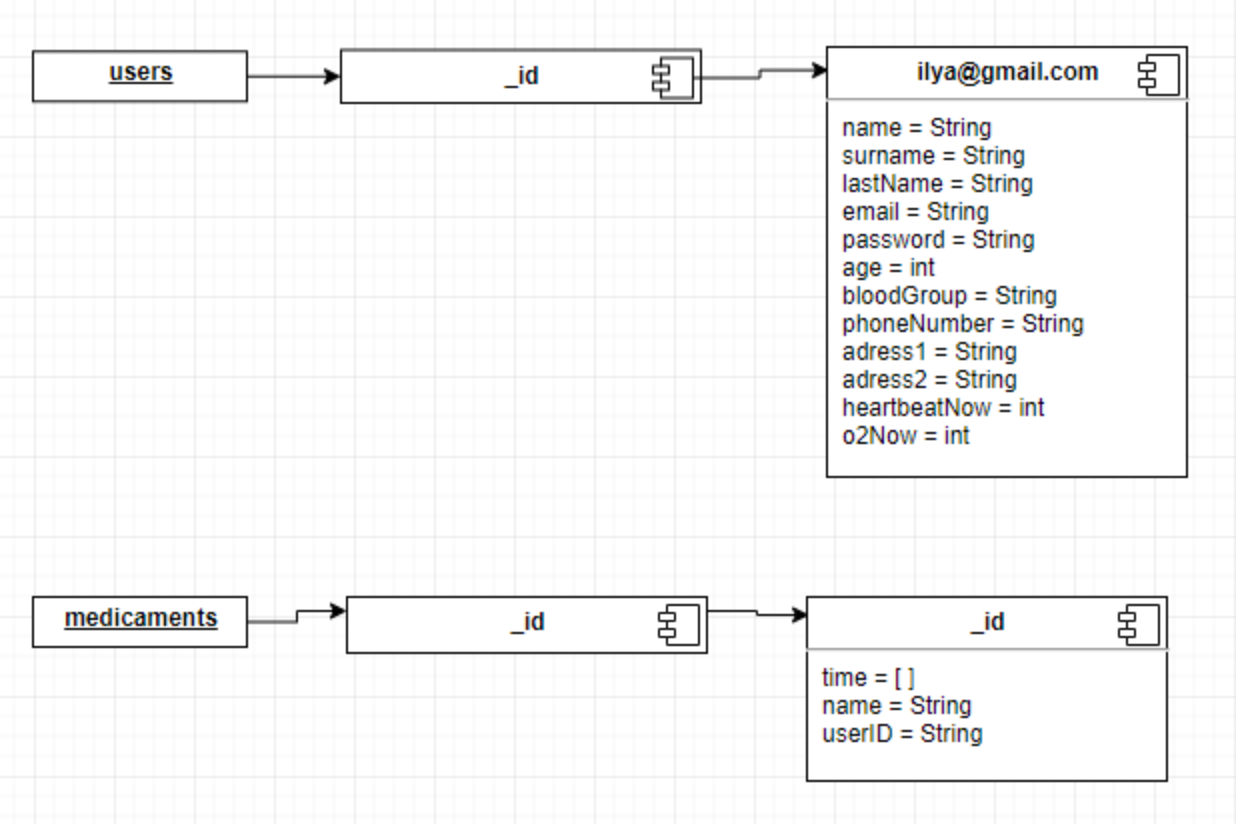


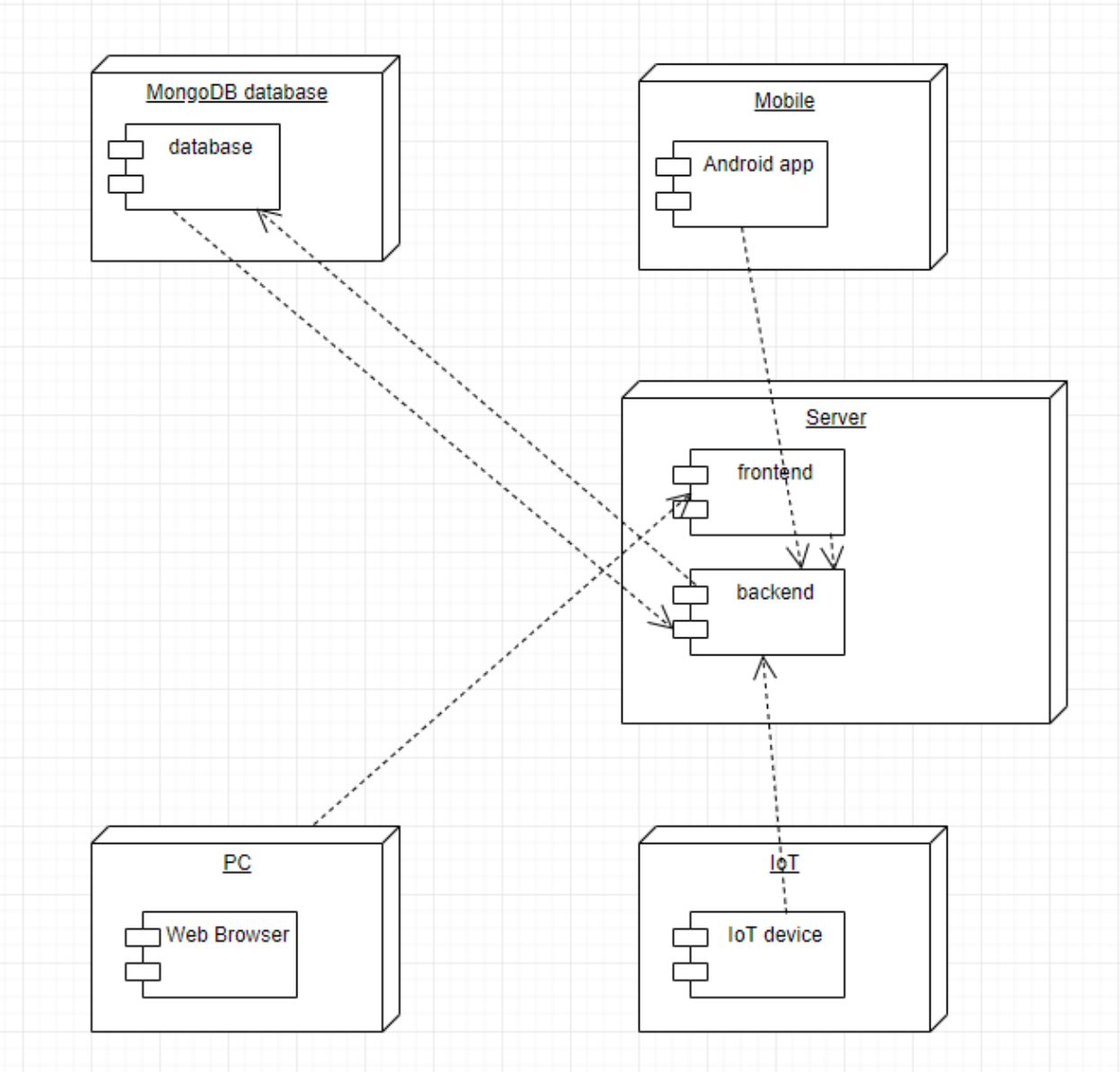
Рисунок 2.2. – Схема бази даних

Висновки:

Після закінчення лабораторної роботи було розроблено програму, на базі якої буде працювати сервер інформаційної системи. Ця частина програмної системи є майже основною для неї. Дана програма дозволяє приймати, обробляти та відповідати на запити клієнта.

Додаток А

Діаграма розгортання



Додаток Б

Фрагмент коду (реалізація створення нового користувача)

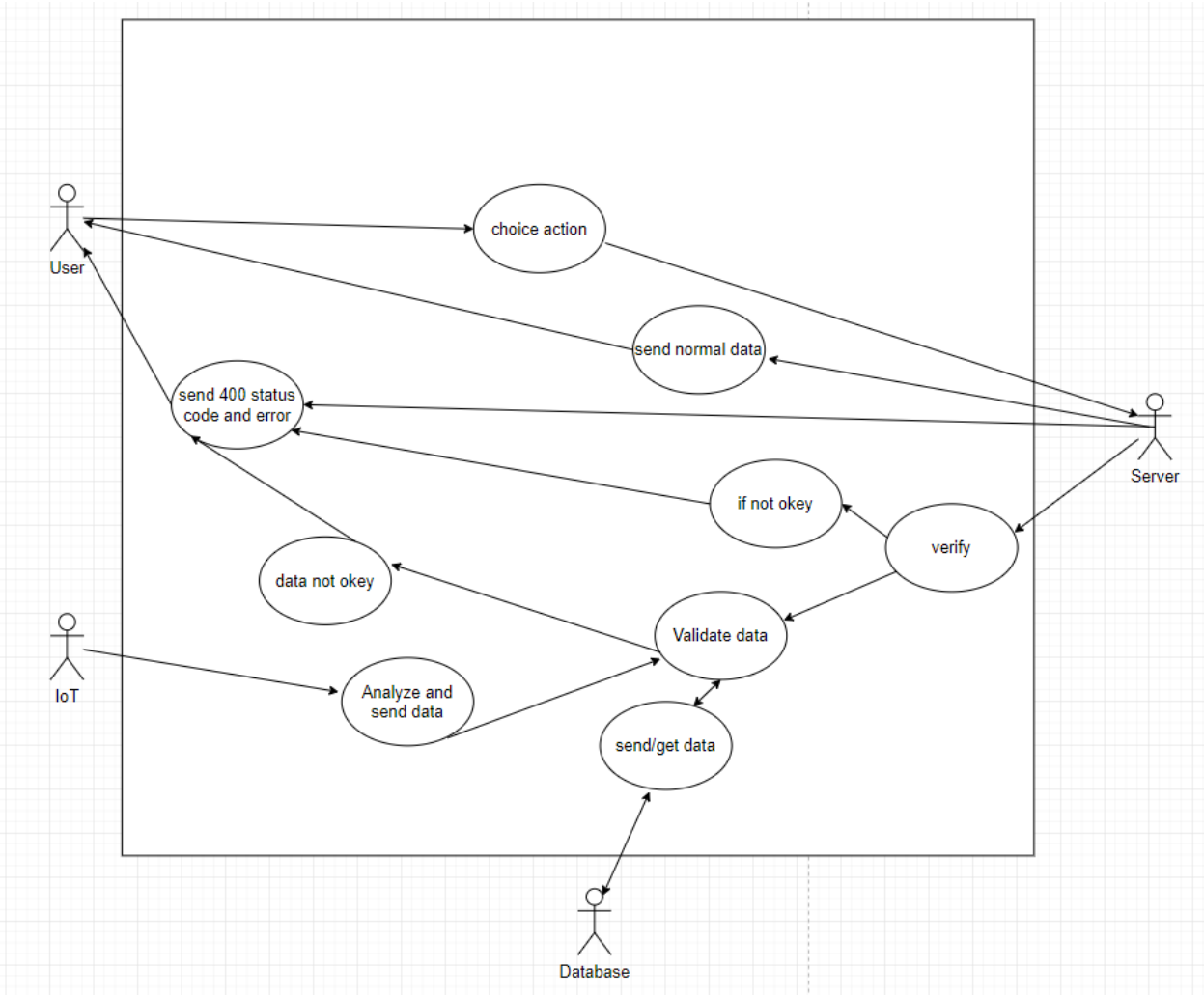
```
router.post('/register', async (req, res) => {
```

```
//! Validate data
const { error } = registerValidation(req.body);
if (error) {
   return res.status(400).send(error.details[0].message);
}

//! Checking is user already exists
const emailExist = await User.findOne({ email: req.body.email });

if (emailExist) {
   return res.status(400).send('Email already exists');
}

//! Hash password
const salt = await bcrypt.genSaltSync(10);
const hashedPassword = await bcrypt.hash(req.body.password, salt);

//! Create a new user
const user = new User({
   email: req.body.email,
   password: hashedPassword,
});
try {
   const savedUser = await user.save();
```

```
    res.send({ user: user._id });
  } catch (error) {
    res.status(400).send(error);
  }
});
```

Додаток В

Use Case Diagram



Додаток Г

Посилання на програмний код:

https://drive.google.com/file/d/1qPMcYUBrwhd4zNherLZdT9CjQ5h4eS6U/view?usp=sharing