Міністерство освіти і науки України
Національний університет водного господарства та природокористування
Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки
Кафедра комп'ютерних технологій та економічної кібернетики

04-05-71М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт і самостійної роботи з навчальної дисципліни
«**Корпоративні інформаційні системи**»
для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Інформаційні технології в бізнесі» спеціальності 126 «Інформаційні системи та технології»
денної і заочної форми навчання

Рекомендовано науково-методичною радою з якості ННІ АКОТ
Протокол № 9 від 31.08.2023 р

Рівне – 2023

Методичні вказівки до виконання лабораторних робіт і самостійної роботи з навчальної дисципліни «Корпоративні інформаційні системи» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Інформаційні технології в бізнесі» спеціальності 126 «Інформаційні системи та технології» денної і заочної форми навчання. [Електронне видання] / Волошин В. С. – Рівне: НУВГП, 2023. – 61 с.

Укладач:
Волошин В. С., к.е.н., доцент, доцент кафедри комп'ютерних технологій та економічної кібернетики.

Схвалено на засіданні кафедри комп'ютерних технологій та економічної кібернетики
Протокол № 1 від 30 серпня 2023 року

Відповідальний за випуск: Грицюк П. М., д.е.н., професор, завідувач кафедри комп'ютерних технологій та економічної кібернетики

Керівник групи забезпечення спеціальності:
Барановський С. В., к.т.н, доцент, доцент кафедри комп'ютерних технологій та економічної кібернетики

Зміст

Вступ	4
Лабораторна робота №1. Поняття фреймворків як засобу для створення КІС	5
Лабораторна робота №2. Проектування корпоративних інформаційних систем	13
Лабораторна робота №3. Бази даних корпоративної інформаційної системи	23
Лабораторна робота №4. Розробка дизайну КІС	27
Лабораторна робота №5. Розробка програмного забезпечення КІС	36
Лабораторна робота №6. Реалізація функціоналу виведення інформації з КІС	50
Лабораторна робота №7. Розмежування прав доступу до КІС	52
Лабораторна робота №8. Розгортання КІС для роботи в комп’ютерних мережах	57
Рекомендована література	61

Вступ

Діяльність суб'єктів економіки в сучасному суспільстві супроводжується використанням комп'ютерних технологій та автоматизованих інформаційних систем. Їх розвиток і вдосконалення вимагають від фахівців знання програмування та інших спеціальних дисциплін, пов'язаних з інформаційними системами і технологіями. Якщо об'єднати ці знання і спрямувати їх на створення системи, яка б повністю автоматизувала всі функції управління та бізнес-процеси на підприємстві, то результатом буде корпоративна інформаційна система.

Системи такого класу активно і масово почали використовуватися наприкінці минулого століття. За цей період вони перетворилися в потужний інструмент управління діяльністю великих підприємств – корпорацій, концернів, холдингів та інших. Тому набуття знань і навиків проектування, розробки, тестування, налаштування, впровадження, експлуатації, супроводження таких систем є актуальним для майбутнього фахівця з інформаційних технологій у бізнесі.

Метою викладання дисципліни «Корпоративні інформаційні системи» є засвоєння теоретичних і практичних знань з основ створення та функціонування корпоративних інформаційних систем.

Основні завдання дисципліни «Корпоративні інформаційні системи» полягають у засвоєнні як теоретичних основ, так і в набутті практичних умінь та навичок застосування засобів сучасних інформаційних технологій для створення корпоративних систем.

Лабораторна робота №1
Поняття фреймворків як засобу
для створення КІС

Хід роботи

Встановлення Node.js

1. Завантажити та встановити LTS версію Node.js: https://nodejs.org/uk/
2. Перевірити роботу фреймворку, для цього:
- відкрити командний рядок cmd;
- виконати команду: node -v

Отримуємо інформацію про версію.

3. Зайдіть в Node.js за допомогою командного рядка, для цього в cmd ввести команду «node». Вийдіть з Node.js командою «.exit»
4. Встановіть редактор Visual Studio Code: https://code.visualstudio.com/
5. Запустіть Visual Studio Code та перейдіть у розділ розширень та при потребі встановіть розширення для зміни мови інтерфейсу. Перезавантажте Visual Studio Code.
6. Аналогічним чином встановіть розширення для підтримки коду Node.js:
- «Node.js Modules Intellisense»
- «Node.js Extension Pack»
- «VS Code for Node.js - Development Pack».

Основи написання програмного коду Node.js

7. Створіть каталог «NodePIB» (де PIB – це Ваше прізвище) та відкрийте його у Visual Studio Code.
8. Створіть новий файл з іменем «index.js» у каталозі «NodePIB.
9. В даному файлі напишіть довільний код на мові JavaScript, наприклад:

console.log('NUWM');

10. Щоб запустити скрипт потрібно створити термінал (команда головного меню «Термінал»-«Створити термінал»). Далі у терміналі прописуємо ключове слово «node» і назву файлу, який потрібно запустити:

node index.js

В результаті виконання скрипта в консолі повинен з'явитися текст «NUWM»

11. Перевірити чи працює npm (менеджер пакунків для мови програмування JavaScript) командою в терміналі (повинна з'явитися версія):

npm –v

12. Здійсніть ініціалізацію проекту (створюється файл «package.json», який потрібний для збереження інформації про проект, а також інформації про пакети, які використовувалися (зручно при передачі проекту)) командою в терміналі:

npm init

Далі послідовно вводимо:

- назву пакету (package name), наприклад: *projectnodejs*;
- версію проекту (version): залишаємо *1.0.0*;
- опис проекту (description), наприклад: *myNewProject;*
- головний файл проекту (entry point): залишаємо index.js;

- тестова команда (test command): пропускаємо;
- репозиторій (git repository): пропускаємо;
- ключові слова (keywords), наприклад: *node, node js*;
- автор (author), наприклад: *PIB pib@gmail.com*;
- ліцензія (license): залишаємо *ISC;*
- натискаємо Enter для збереження.

13. Перевірте чи створився файл «package.json» з всіма раніше введеними даними.

Встановлення потрібних пакетів (модулів) для проекту Node.js

14. Встановіть модуль форматування тексту chalk версії 4.1.0. Для цього виконайте команду в терміналі (відбувається підключення до сервера в мережі Інтернет і завантаження потрібних файлів, і - install):

npm i chalk@4.1.0

15. Перевірте чи з'явився опис модуля chalk у файлі «package.json», наприклад:

```
...
"dependencies":                                    {
    "chalk": "^4.1.0",
...
```

та створилася папка «node_modules» у каталозі каталог «NodePIB».

Примітка: при потребі для видалення модуля використовується команда:

npm remove -D chalk

16. Встановіть модуль nodemon (інструмент, який допомагає розробляти додатки на основі node.js шляхом автоматичного перезапуску програми вузла, коли виявляються зміни файлів у каталозі). Команда в терміналі (тут два дефіса):

npm i nodemon --D

Робота із модулями Node.js

17. У файлі «index.js» пропишіть простий код:

const s1 = require('chalk'); //імпорт модуля chalk; s1 - змінна

console.log(s1.blue('NUWM')); //виводить синім кольо ром в консоль

console.log(s1.rgb(34, 208, 95)('NUWM')); //колір у форматі rgb

18. Збережіть зміни та запустіть файл «index.js» в терміналі. Перегляньте результат.

19. Допишіть код файлу «index.js», щоб у консоль виводилося Ваші прізвище (колір вибираємо самостійно).

20. Для створення свого власного модуля створіть файл «data.js» з кодом:

const text1='НУВГП'
module.exports = text1 //експорт даних в інші файли

а у файлі «index.js» дописуємо:

const s2 = require('./data')
console.log(s1.blue(s2))

21. Збережіть зміни та перевірте експорт/імпорт власного модуля запустивши файл «index.js».

22. Встановіть модуль для отримання інформації про файли, каталоги та шляхи path.

23. Створіть файл «path.js» та запишіть у ньому код для роботи зі шляхами файлів (рис. 1).

```
const path1 = require('path')  11.9K (gzipped: 4.3K)

console.log('Назва файлу:', path1.basename(__filename)) //повертає назву поточного файлу
console.log('Каталог:', path1.basename(__dirname)) //повертає назву каталогу
console.log('Розширення:', path1.extname(__filename)) //розширення файлу
console.log('Інфо:', path1.parse(__filename)) //повна інформація про файл

console.log(path1.join(__dirname,'node_modules','.bin','nodemon.cmd')) //отримаємо шлях до файлу
```

Рис. 1. Приклад коду Node.JS для роботи зі шляхами

24. Збережіть зміни та перевірте відображення даних у консолі.

25. Встановіть модуль для отримання інформації про операційну систему os.

26. Створіть файл «os.js» та запишіть код для отримання інформації про операційну систему (рис. 2).

```
const os1 = require('os')  208 (gzipped: 150)

console.log('Операційна система ', os1.platform())
console.log('Розрядність ЦП ', os1.arch())
console.log('ЦП ', os1.cpus())
console.log('Вільна память ОЗУ, Байт', os1.freemem())
console.log('Всього ОЗУ', os1.totalmem())
console.log('папка Документи', os1.homedir())
console.log('Час безвідмовної роботи ОС (від перезагрузки), сек', os1.uptime())
console.log('Назва хоста в мережі', os1.hostname())
```

Рис. 2. Приклад коду Node.JS для роботи з операційною системою

27. Збережіть зміни та перевірте відображення даних про операційну систему у консолі.

28. Встановіть модуль для роботи з файлами fs.

29. Створіть файл «fs1.js» та запишіть код для створення каталогу (рис. 3).

```
const fs1 = require('fs') // fs - file system  187 (gzipped: 146)
const path1 = require('path')  11.9K (gzipped: 4.3K)

// створення каталогу "PIB". Якщо каталог вже створено то виникає помилка
Complexity is 3 Everything is cool!
fs1.mkdir(path1.join(__dirname, 'PIB'), (err) => {
    if (err) {
        throw err //throw показує визначений користувачем виняток (в даному випадку помилку)
    }
    console.log('Каталог створено')
})
```

Рис. 3. Приклад коду Node.JS для роботи зі шляхами

30. Збережіть зміни та перевірте створення каталогу.

31. Створіть файл «fs2.js» та запишіть код для створення файлу та додавання до нього тексту (рис. 4).

```
const fs1 = require('fs') // fs - file system  187 (gzipped: 146)
const path1 = require('path')  11.9K (gzipped: 4.3K)

//створення файлу, якщо існує то перезаписується
const filePath = path1.join(__dirname,'PIB','text.txt') // у змінну filePath
// записуємо повний шлях до певного файлу, перевірка console.log (filePath)
Complexity is 3 Everything is cool!
fs1.writeFile(filePath, 'Текст у файлі, що створюється', (err) => {
    if (err) {
        throw err //throw викидає визначений користувачем виняток (в даному випадку помилку)
    }
    console.log('файл створено')
} )

//додавання тексту до створеного файлу
Complexity is 3 Everything is cool!
fs1.appendFile(filePath, '\nТекст 2', (err) => {
    if (err) {
        throw err //throw викидає визначений користувачем виняток (в даному випадку помилку)
    }
    console.log('файл дописано')
} )
```

Рис. 4. Приклад коду Node.JS для створення файлу

32. Створіть файл «fs3.js» та запишіть код для читання файлу (рис. 5).

```
const fs1 = require('fs')   187 (gzipped: 146)
const path1 = require('path')  11.9K (gzipped: 4.3K)

const filePath = path1.join(__dirname,'PIB','text.txt') // у змінну filePath
//читання з файла
Complexity is 3 Everything is cool!
fs1.readFile(filePath, (err, content) => { //content отримуємо контент файлу у
    // форматі буферу, наприклад:  <Buffer d0 a2 d0 b5 d0 ba d1 81 d1 82 20 d1 83 2
    //сихронна функція  readFileSync
    if (err) {
        throw err //throw викидає визначений користувачем виняток (в даному випадку помилку)
    }
    console.log('Контент у форматі буферу:', content)
    const s1 = Buffer.from(content) //s1 - конетнт у звичному форматі
    console.log('Контент: ', s1.toString())
} )
```

Рис. 5. Приклад коду Node.JS для читання з файлу

33. Створіть файл «fs4.js» та запишіть альтернативний код для читання файлу (рис. 6).

```
const fs1 = require('fs')   187 (gzipped: 146)
const path1 = require('path')  11.9K (gzipped: 4.3K)


const filePath = path1.join(__dirname,'PIB','text.txt')

//читання з файла (простіше за рахунок прописування кодування utf-8)
Complexity is 3 Everything is cool!
fs1.readFile(filePath,'utf-8', (err, content) => {
    if (err) {
        throw err //throw викидає визначений користувачем виняток (в даному випадку помилку)
    }
    console.log('Контент ', content)
} )
```

Рис. 6. Альтернативний приклад коду Node.JS для читання з файлу

34. Встановіть модуль для обробки подій events.

35. Створіть файл «events.js» та запишіть код з подіями та використання таймеру (рис. 7).

36. Збережіть зміни та перевірте вивід повідомлень у консоль в певному порядку подій.

37. Встановіть модуль для роботи з електронною поштою nodemailer.

38. Включіть двохетапну перевірку акаунту Google (відмінного від корпоративної пошти), який буде

використовуватися для відправки електронних листів. Для цього: перейдіть у налаштування акаунту Google: https://myaccount.google.com/; далі виберіть «Безпека»-«Двохетапна перевірка» та дотримуйтеся подальших інструкцій.

```
const eventEmitter = require('events') // events - події  6K (gzipped: 2K)
const eventEmitter1 = new eventEmitter()

eventEmitter1.on('name1', (data1) => { // name1 - назва події, data1 - параметр (функція)
    console.log('Повідомлення ', data1)
} )

eventEmitter1.emit('name1', {a: 1})  //виклик функції
//метод emit дозволяє передавати довільний набір аргументів функціям
eventEmitter1.emit('name1', {b: 2})

setTimeout(() => { //таймер
    eventEmitter1.emit('name1', {c: 3}) //функція таймера
}, 2000) //мілісекунди
```

Рис. 7. Приклад коду Node.JS для роботи з таймером

39. Після успішного ввімкнення двохетапної перевірки, згенеруйте ключ доступу у налаштуваннях акаунту Google «Безпека»-«Паролі додатків». При цьому потрібно задати користувацьку назву додатка ««node.js».

40. Створіть файл «mail.js» та запишіть код для відправки електронних листів, при цьому потрібно вказати пароль, що був згенерований вище (рис. 8).

41. Збережіть зміни. Перевірте відправлення електронних листів.

42. Показати та захистити роботу викладачеві.

43. При бажанні знову вимкніть двохетапну перевірку акаунту Google.

```
const nodemailer = require('nodemailer');
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'vvcrv84@gmail.com',
        pass: 'вкажіть Ваш пароль'
    }
});


const mailOptions = {
    from: 'vvcrv84@gmail.com', // Відправник
    to: 'km11@ukr.net', // Отримувач, якщо декільком, то через кому
    subject: 'Працюємо у NodeJS', // Тема
    html: '<h1>Лист надіслано</h1>'// Текст повідомлення
};

Complexity is 3 Everything is cool!
transporter.sendMail(mailOptions, function(error, info){
    if (error) {
        console.log(error);
    } else {
        console.log('Email sent: ' + info.response);
    }
});
```

Рис. 8. Приклад коду Node.JS для відправки електронних листів

Лабораторна робота №2
Проектування корпоративних інформаційних систем

Теоретичні відомості

Побудована належним чином АІС (автоматизована інформаційна система) забезпечує доступ до оновлених і точних відомостей. Оскільки правильна структура є необхідною умовою для досягнення поставленої мети під час роботи з АІС, доцільним є вивчення принципів її правильної побудови.

У АІС всі дані організовуються в таблиці: списки рядків і стовпців, які нагадують бухгалтерську книгу або електронну таблицю. У простих системах може міститися лише одна таблиця. Для більшості АІС необхідна наявність кількох таблиць. Наприклад, перша таблиця може містити відомості про конференції, друга таблиця – відомості про секції, а третя – відомості про учасників конференції.

Кожен рядок також називається записом, а кожен стовпець – полем. Запис – це осмислений і узгоджений спосіб поєднання відомостей про щось. Поле – це окремий елемент відомостей – тип елемента, відображений у кожному записі. Наприклад, у таблиці «Конференції» кожен рядок або запис містить відомості про одну конференцію. Кожен стовпець або поле містить певні типи відомостей про конференцію, наприклад дату початку або завершення.

Правильна структура АІС

Під час створення АІС стануть у нагоді певні принципи. Відповідно до першого принципу, потрібно уникати повторюваних відомостей (надлишкових даних), оскільки вони займають зайве місце та збільшують вірогідність виникнення помилок і невідповідностей. За другим принципом, важливу роль відведено правильності та завершеності даних. Якщо база даних містить неправильні відомості, всі звіти, в яких об'єднано відомості з АІС, також міститимуть неправильну інформацію. Це може призвести до прийняття неправильних рішень на основі цих звітів.

Ознаки правильної структури АІС:

- Розділення даних на тематичні таблиці для зменшення обсягу надлишкових даних.

- Забезпечення відомостями, необхідними для об'єднання даних у таблицях.
- Допомога в підтриманні та забезпеченні точності й цілісності інформації.
- Приведення даних у відповідність потребам оброблення та звітування.

Процес розробки АІС

Процес розробки АІС складається з таких етапів:

1. Визначення мети створення АІС. Допомагає підготуватися до виконання наступних кроків. Потрібно дати відповідь на питання: Для чого створюється інформаційна система? Наприклад для АІС «Конференції»: АІС створюється для автоматизації обробки даних про проведені конференції.

2. Пошук і впорядкування потрібних відомостей. Збирає всі типи даних, які потрібно зберегти в базі даних, наприклад, назва конференції, дата початку та закінчення конференції, дані про учасників.

3. Розділення даних на таблиці. Розділяє елементи даних на групи, наприклад, «Конференції» або «Секції». Кожну групу у подальшому буде перетворено на таблиці.

4. Перетворення елементів даних на стовпці. Вирішіть, які дані потрібно зберегти в кожній таблиці. Кожен елемент буде перетворено на поле та відображено як стовпець у таблиці. Наприклад, таблиця «Конференції» може містити такі поля, як «Назва конференції», «Дата початку», «Дата закінчення».

5. Визначення первинних ключів. Виберіть первинні ключі для кожної таблиці. Первинним ключем є стовпець, який використовується для унікального визначення кожного рядка в таблиці. Наприклад, «Код конференції» («id_konferencii»).

6. Створення зв'язків між таблицями. Прогляньте всі таблиці та визначте, як дані однієї таблиці зв'язано з даними в інших таблицях. Додайте поля до таблиць або створіть нові таблиці, щоб у разі потреби уточнити зв'язки.

7. Удосконалення структури. Проаналізуйте структуру АІС на наявність помилок. Створіть таблиці та додайте кілька записів зі зразками даних. Перегляньте, чи за допомогою цих таблиць можна отримати потрібні результати. Якщо потрібно, внесіть до структури зміни.

8. Застосування правил нормалізації. Застосуйте правила нормалізації даних, щоб переглянути правильність структури таблиці. Якщо потрібно, внесіть до таблиць зміни.

Визначення мети створення АІС

Рекомендовано записати мету створення АІС на папері, її призначення, хто і як її планує використати. Для невеликої АІС, наприклад, для діловодства, можна визначити таку просту мету «АІС клієнтів містить список відомостей про клієнтів і використовується для створення розсилок і звітів». Якщо система складніша або використовується багатьма користувачами, наприклад, в організації, опис мети може складати один або кілька абзаців і має містити час і способи використання АІС кожним користувачем. Основним завданням є створення добре організованого опису завдання, до якого можна звернутися під час процесу розробки. Наявність такого опису допоможе зосередитися на визначених цілях під час прийняття рішень.

Пошук і впорядкування потрібних даних

Пошук і впорядкування потрібних даних починається з аналізу наявних відомостей на підприємстві або

організації. Наприклад, можна записати замовлення на придбання в основній книзі або зберегти відомості про клієнтів на паперових формах у картотеці. Зберіть ці документи та складіть список відображених даних. Наприклад, припустімо, що список клієнтів наразі зберігається на облікових картках. Можливо, аналіз цих карток з'ясує, що кожна картка містить ім'я клієнта, адресу, місто, область, поштовий індекс і номер телефону. Кожен із цих елементів може відповідати стовпцю в таблиці.

Під час підготовки цього списку немає потреби намагатися одразу надати йому досконалого вигляду. Натомість, внесіть у список усі елементи, які спадають на думку. Якщо АІС використовуватиметься іншими користувачами, попросіть їх запропонувати свої варіанти. Список можна точно настроїти пізніше.

Після цього визначте типи звітів або розсилок, які, можливо, потрібно буде створити на основі АІС. Наприклад, можна створити звіт про продаж товарів для відображення продажу за регіонами або зведений звіт про запаси для відображення кількості товару. Може також знадобитися створення стандартних листів для надсилання клієнтам, які повідомлятимуть про продажі або спеціальні пропозиції. Продумайте структуру звіту й уявіть як він виглядатиме. Які відомості потрібно включити у звіт? Складіть список усіх елементів. Виконайте ті самі дії для стандартних листів та інших звітів, які потрібно створити.

Аналіз звітів і розсилок, які потрібно створити, допоможе визначити елементи, які потрібно включити в базу даних. Наприклад, припустімо, що потрібно надати клієнтам можливість підписатися (або скасувати підписку) на отримання періодичних оновлень електронною поштою, а також потрібно надрукувати список користувачів, які підписалися. Щоб записати ці відомості, додайте до таблиці

клієнтів стовпець «Надсилання електронної пошти». У цьому полі для кожного клієнта можна встановити значення «Так» або «Ні».

Важливо пам'ятати, що дані потрібно розділити на найменші значимі частини. Ім'я потрібно розділити на дві частини — ім'я та прізвище, щоб прізвище було доступне для окремого використання. Наприклад, для сортування звіту за прізвищем це допоможе окремо зберегти прізвище клієнта. Загалом, якщо потрібно виконати сортування, пошук, обчислення або звіт, основані на елементі інформації, цей елемент слід зберегти в окремому полі.

Розділення даних на таблиці

Щоб поділити дані на таблиці, виберіть основні групи або теми.

Основними групами при цьому є конференції, секції, учасники. Отже, доцільним буде створення цих трьох таблиць. Хоча список не є повним, проте почати слід саме зі створення таких основних таблиць. Можна продовжити уточнювати цей список, доки не буде створено повнофункціональну структуру АІС.

Оскільки конференція може містити багато секцій, назву конференції для секції потрібно повторити кілька разів. У разі виконання таких дій місце на диску витрачається марно. Найкраще рішення — внести відомості про конференцію лише один раз в окрему таблицю «Конференції», відтак зв'язати цю таблицю з таблицею «Секції».

Під час розробки АІС завжди намагайтеся зберігати елемент інформації лише один раз.

Після вибору теми для таблиці стовпці в цій таблиці мають зберігати лише дані, які відповідають цій темі. Наприклад, таблиця «Учасники» має зберігати лише

відомості про учасників. Оскільки назва секції є інформацією про секції, а не про учасника, вона має міститися в таблиці «Секції».

Перетворення елементів даних на стовпці

Рекомендації для визначення стовпців (полів).

- Не включайте в таблицю обчислювані дані

У більшості випадків не слід зберігати результати обчислень у таблицях. Замість цього для відображення результатів можна виконати обчислення в програмному продукті. Наприклад, для розрахунку тривалості конференції не потрібно створювати окреме поле «Тривалість конференції», оскільки його можна розрахувати як різницю полів «Дата закінчення» та «Дата початку».

- Зберігайте інформацію найменшими логічними частинами

Може здатися доцільним створення одного поля для збереження прізвища, ім’я та по-батькові учасника. Проте, об'єднання кількох типів інформації в одному полі ускладнить подальше витягнення окремих даних. Потрібно розділити інформацію на логічні частини; прізвище – окремо, ім’я – окремо, по-батькові - окремо.

Визначення первинних ключів

Кожна таблиця має містити стовпець або набір стовпців для унікального визначення кожного рядка в таблиці. Зазвичай, це унікальний ідентифікаційний номер, наприклад, ідентифікаційний номер конференції, який у базі даних зазначається «id_konferentsii» . У термінології AIC ця інформація має назву первинний ключ таблиці. В AIC поля первинного ключа використовуються для

швидкого зв'язування даних із кількох таблиць і їх надання користувачу.

Первинний ключ не може містити повторювані значення. Наприклад, не використовуйте як первинний ключ імена осіб, оскільки вони не є унікальними. В одній таблиці можуть міститися дві особи з однаковими іменами.

Первинний ключ має завжди містити значення. Якщо стовпець має непризначене або невідоме (відсутнє) значення, його не можна використати як компонент первинного ключа.

Створення схематичних зв'язків між таблицями

Після того, як дані розділено на таблиці, їх необхідно знову об'єднати у зрозумілий спосіб.

У реляційній базі даних відомості розділяються на окремі тематичні таблиці. Для об'єднання цих відомостей використовуються схематичні зв'язки між таблицями, які зазначаються лише у схемі для наочного розуміння структури інформаційної системи.

Удосконалення структури

Після створення необхідних таблиць, полів і зв'язків таблиці слід заповнити зразками даних і спробувати виконати з ними певні операції: створення запитів, додавання нових записів тощо. Виконання цих дій допоможе виявити потенційні проблеми — наприклад, може знадобитися додати стовпець, який не було додано під час процесу розробки, або поділити одну таблицю на дві для видалення повторюваних даних.

Перевірте, чи можна отримати відповіді на потрібні запитання за допомогою АІС. Створіть чорнові форми й звіти та переконайтеся, що вони відображають потрібні дані. Перевірте базу даних на наявність повторюваних

даних, і якщо їх знайдено, змініть структуру для їх видалення.

Під час випробування початкової АІС можна виявити можливості для її вдосконалення. Перевірте таке:

- Відсутність потрібних стовпців. Якщо деякі стовпці відсутні, перевірте чи дані в цих стовпцях мають міститися в наявних таблицях. Якщо дані належать до іншої теми, може знадобитися створення іншої таблиці. Створіть стовпець для кожного елемента даних, який потрібно відстежувати. Якщо дані не можна обчислити з інших стовпців, необхідно створити для них новий стовпець.
- Наявність стовпців, які є непотрібними, оскільки дані в них можна обчислити за допомогою наявних полів. Якщо елемент даних можна обчислити за допомогою інших наявних стовпців, наприклад, ціну зі знижкою можна обчислити з роздрібної ціни, рекомендовано уникати створення нового стовпця.
- Наявність багаторазових повторень даних у таблиці. У такому разі потрібно поділити таблицю на дві таблиці.
- Наявність таблиць із великою кількістю полів, обмеженою кількістю записів і пустими полями в окремих записах. У такому разі може знадобитися змінення структури таблиці, щоб скоротити кількість полів і збільшити кількість записів.
- Розділення кожного елемента даних на менші значимі компоненти. Якщо потрібно використати елемент інформації для створення звітів, сортування, пошуку або обчислення, помістіть його до окремого стовпця.
- Наявність у кожному стовпці відомостей, які відповідають темі таблиці. Якщо стовпець містить дані, які не відповідають темі таблиці, їх потрібно перемістити в іншу таблицю.

- За наявності повторюваних груп розгляньте можливість поділу однієї таблиці на дві.

Хід роботи

Розробіть схему даних автоматизованої інформаційної системи накресливши її в будь-якому онлайн редакторі (наприклад, https://ondras.zarovi.cz/sql/demo/ або https://creately.com/lp/online-database-design-tool/), відповідно до варіанту, що наведений нижче (номеру по списку у журналі) за наступними основними вимогами:

1. Кількість таблиць – не менше шести.
2. Кількість полів у таблиці – не менше трьох.
3. Використати поля різних типів даних, а саме:

- числові (цілочисельні);
- числові (дробові);
- текстові;
- поля дати;
- логічні поля типу так або ні.

4. У кожній таблиці повинно бути ключове поле.

Варіанти виконання завдань

1. Створення АІС абонементу бібліотеки.
2. Створення АІС автостоянки.
3. Створення АІС архіву особових справ співробітників.
4. Створення АІС обліку новобудов міста.
5. Створення АІС гуртожитку.
6. Створення АІС підприємства харчової промисловості
7. Створення АІС підприємства машинобудівної галузі.
8. Створення АІС торгівельного підприємства.

9. Створення АІС співробітників вузу.
10. Створення АІС музею.
11. Створення АІС туристичної фірми.
12. Створення АІС перукарні.
13. Створення АІС школи.
14. Створення АІС фотосалону.
15. Створення АІС служби зайнятості.
16. Створення АІС спортивного залу.
17. Створення АІС студентів групи.
18. Створення АІС днів народжень співробітників.
19. Створення АІС сільської ради.
20. Створення АІС фермерського господарства.
21. Створення АІС лісового господарства.

Лабораторна робота №3
Бази даних корпоративної інформаційної системи

Хід роботи

1. Завантажте XAMPP (безкоштовна багатоплатформова збірка веб-сервера з відкритим початковим кодом, що містить HTTP-сервер Apache, базу даних MariaDB, MySQL й інтерпретатори скриптів для мов програмування PHP та Perl) за наступним посиланням:

https://www.apachefriends.org/ru/index.html

2. Встановіть XAMPP.

3. Завантажте сервер запустивши «xampp-control.exe» у каталозі XAMPP.

4. Запустіть потрібні служби.

5. Перевірити роботу локального сервера, ввівши у браузері «http://localhost».

6. Для зручності створення БД встановіть MySQL-Front, а також Navicat.

7. Завантажити додаток MySQL-Front за допомогою ярлика на робочому столі.

8. Створіть новий сеанс.

9. Створіть нову базу даних з назвою «konferentsiyi_*x*» (де *x* – Ваше прізвище малими латинськими літерами), для цього виконайте пункт головного меню «База даних» - «Створити» - «База даних …» . У вікні, що з'явилося введіть назву БД та встановіть кодування.

10. Створіть таблицю «kafedry». Для цього виділіть створену базу даних «konferentsiyi» із контекстного меню виконайте команду «Створити» - «Таблиця…». У вікні, що з'явилося введіть параметри, що зображені на рис. 9.

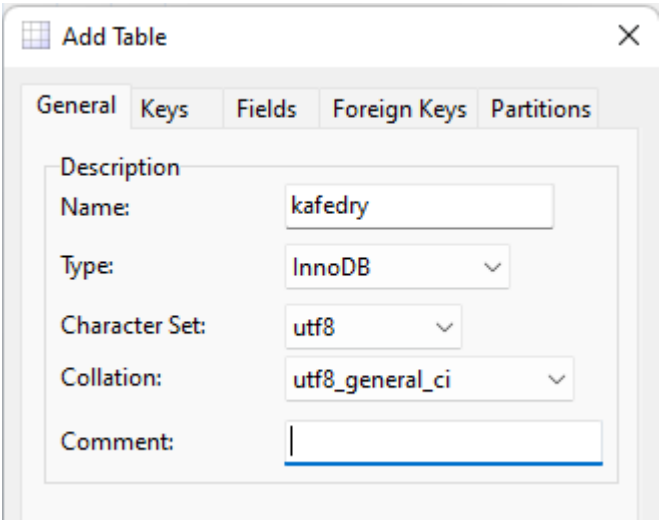


Рис. 9. Створення таблиці БД

11. Відкрийте таблицю «kafedry» та змініть назву ключового поля цієї таблиці з «id» на «id_ kafedry».

12. У таблиці «kafedry» створіть поле «nazva_kafedry».

13. У таблиці «kafedry» аналогічним чином створіть поле «roztashuvannya» (довжина поля повинна бути – 255 символів).

14. Створіть таблицю «spivrobitnyky».

15. У таблиці «spivrobitnyky» створіть поля, що наведені на рис. 10.

Поля (5)

id_spivrobitnyky	int(11) unsigned	Нет	<auto_increment>
id_ kafedry	int(11) unsigned	Да	<NULL>
prizvyshche	varchar(100)	Да	<NULL>
imya	varchar(100)	Да	<NULL>
pobatkovi	varchar(100)	Да	<NULL>

Рис. 10. Поля таблиці «Співробітники»

16. Відповідно до наведеної схеми даних (рис. 11) створіть таблицю «Конференції» з необхідними полями.

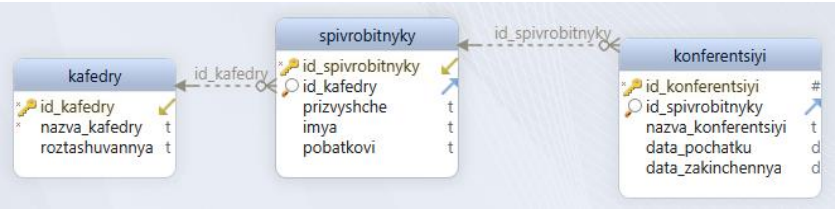


Рис. 11. Схема даних автоматизованої інформаційної системи «Конференції»

17. Запустіть Navicat MySQL за допомогою ярлика на робочому столі.

18. Створіть нове підключення за допомогою головного меню «File»-«New Connection». У вікні, що з’явилося введіть параметри, що зображено на рис. 12.

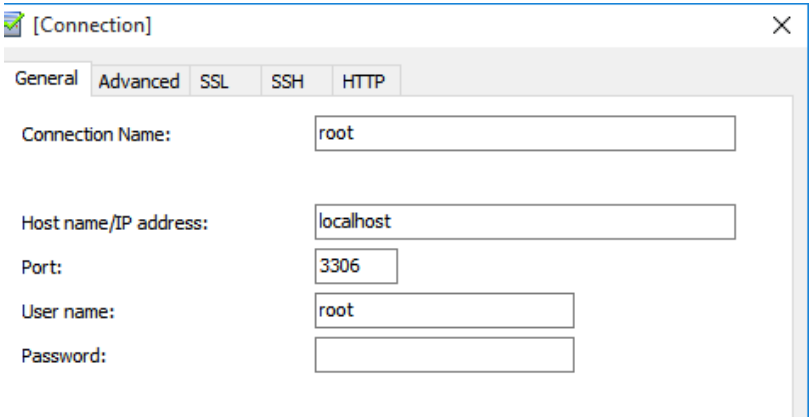


Рис. 12. Вікно підключення до сервера

19. Відкрити підключення «root».
20. Відкрийте свою базу даних.
21. Відкрийте таблицю «kafedry» та заповніть дані про шість кафедр (поле «id_ kafedry» заповнювати не потрібно).
22. Відповідно до логічної моделі даних АІС «Конференції», створіть зв’язок між таблицями «Кафедри» та «Співробітники». Для цього:

- відкрийте властивості таблиці «spivrobitnyky» (пункт контекстного меню «Design Table») та перейдіть на вкладку «Foreign Keys» («Зовнішні Зв’язки»);
- у вікні, що з’явилося введіть наступні параметри: назва зв’язку, поля, що з’єднуються та відповідні таблиці;
- збережіть виконані зміни.

23. Відкрийте таблицю «spivrobitnyky» та заповніть дані про десятьох співробітників.

24. Відповідно до схеми даних АІС «Конференції створіть аналогічні зв’язки між таблицями «Співробітники» та «Конференції».

25. Заповніть таблицю «Конференції» 6 записами.

26. Створіть резервну копію бази даних «konferentsiyi». Для цього у властивостях бази даних «konferentsiyi» потрібно вибрати «Dump SQL File...».

27. Покажіть роботу викладачеві.

Лабораторна робота №4
Розробка дизайну КІС

Хід роботи

1. Створіть каталог «konferentsiyi_PIB» (де PIB – це Ваше прізвище) та відкрийте його у Visual Studio Code.

2. Створіть підпорядкований каталог «views».

3. Створіть файл «package.json» командою в терміналі:

npm init

4. Для роботи сервера вносимо зміни у файл «package.json» у розділі «scripts» (рис. 13).

```
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node index",
9     "dev": "nodemon index"
10  },
```

Рис. 13. Код файлу «package.json» для налаштування сервера

5. Встановлюємо необхідні модулі: «http» (0.0.1-security), «fs» (0.0.1-security), «path» (0.12.7), «express» (4.17.1), «nodemon» (2.0.7), «body-parser» (1.19.0), «mysql2» (2.2.5), «express-handlebars» (5.3.2) і так далі наприклад:

npm i http

Можна всі зразу через пробіл:

npm i fs path express nodemon body-parser mysql2 express-handlebars

6. Інсталюємо модуль шаблонів hbs:

npm i hbs@4.1.2

7. У каталозі «views» створюємо пусті файли для макету сторінки: «header.hbs» (файл шапки сайту), «menu.hbs» (файл меню), «footer.hbs» (файл нижнього колонтитула), «start.hbs» (файл контенту початкової сторінки).

8. У каталозі «konferentsiyi_PIB» створюємо пустий файл «index.js».

9. Файл «index.hbs» містить структуру сайту. Вигляд коду файлу «index.hbs» зображено на рисунку 14.

10. Для візуального оформлення Web-сторінок підключаємо Bootstrap (безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript) у файлі «index.hbs» (через CDN):

```
<!DOCTYPE html>
<html>
<head>
    <title>Конференції</title>
    <meta charset="utf-8" />

    <style>
        .jumbotron {
            background: url("/header.jpg");
        }
    </style>
</head>
<body>
{{> header}} <!-- Виокремлення блоку шапки -->

{{> menu}} <!-- Виокремлення блоку меню -->

{{{body}}} <!-- Вставка контенту кожної окремої сторінки body - зарезервоване слово-->

{{> footer}} <!-- Виокремлення блоку footer -->



</body>
</html>
```

Рис. 14. Структура сайту Node.js

в тезі <head>:

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

та перед закриттям тегу </body>:

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZw

rnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>

11. В каталозі «konferentsiyi_PIB» створіть каталог «img», який буде зберігати зображення для майбутнього сайту.

12. Віднайдіть в мережі Інтернет іконку для майбутнього сайту. Збережіть її у каталог «img» з іменем «favicon.ico» та допишіть наступний код після тегу <title> у файлі «index.hbs»:

<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon">

13. Змініть мову Web-сторінки:

<html lang="uk">

14. Відкрийте файл шапки сайту «header.hbs».

15. Додайте блок шапки сайту та забезпечте приховування блоку заголовку на малих пристроях.

16. Виберіть фонове зображення шапки сайту та збережіть його у каталог «img» з іменем «header.jpg».

17. Поставте курсор після тегу <header> та додайте код компоненту Bootstrap Jumbotron (рис. 15):

```
<header class="d-none d-sm-block">
    <div class="jumbotron text-white text-center">
        <h1 class="display-4">{{zagolovok}}</h1>
        <p class="lead">Web-сайт обліку конференцій закладів вищої освіти</p>
        <hr class="my-4">
    </div>
</header>
```

Рис. 15. Код компоненту Bootstrap Jumbotron

18. У файлі «index.js» (головному файлі сервера) прописуємо константи (рис. 16).

```
const mysql = require("mysql2");  Calculating...
const express = require("express");  Calculating...
const bodyParser = require("body-parser");  Calculating...
const hbs = require("hbs");  Calculating...
const expressHbs = require("express-handlebars");
const path = require('path') //робота з шляхами  11.9K (gzipped: 4.3K)
const app = express();
const urlencodedParser = bodyParser.urlencoded({extended: false}); //для обробки POST форм
```

Рис. 16. Константи Node.js

19. У файлі «index.js» для доступу до файлів у каталозі «img» прописуємо статичний шлях:

```
app.use(express.static('img'))
```

20. У файлі «index.js» для виокремлення однотипного тексту (наприклад тексту заголовку у шапці) прописуємо константу:

```
const zagolovok = "Облік конференцій"
```

21. Відкрийте файл шапки сайту «menu.hbs» та пропишіть код головного меню (використовується компонент Bootstrap Navbar – навігаційна панель для головного меню сайту):

<nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">
*<img src="/favicon.ico"> *

```
        <button class="navbar-toggler" type="button" data-
toggle="collapse"
            data-target="#navbarSupportedContent1" aria-
controls="navbarSupportedContent1"
            aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-
collapse" id="navbarSupportedContent1">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/konferentsiyi">
                        Конференції <span class="sr-
only">(current)</span>
                    </a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-
toggle" href="/" id="navbarDropdown1"
                    role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
                        Довідники
                    </a>
                    <div class="dropdown-menu" aria-
labelledby="navbarDropdown1">
                        <a class="dropdown-
item" href="/kafedry">
                            Кафедри
                        </a>
                        <a class="dropdown-
item" href="/mistsya_prozhyvannya">
                            Співробітники
```

```
                </a>
              </div>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="/logout">
                Вихід
              </a>
            </li>
          </ul>
        </div>
      </nav>
```

22. У файлі «footer.hbs» аналогічним чином пропишіть код нижнього колонтитула (рис. 17).

```
<footer class="text-center my-5">
    <div class="container">
        <div class="row">
            <div class="col-12">
                <p>Copyright © Іванов І.І. Всі права захищені.</p>
            </div>
        </div>
    </div>
</footer>
```

Рис. 17. Код нижнього колонтитула Web-сайту

23. У файлі «index.js» прописуємо підключення до БД MySQL локального сервера (рис. 18).

```
const pool = mysql.createPool({
    connectionLimit: 5,
    host: "localhost",
    user: "root",
    database: "konferentsiyi",
    password: ""
});
```

Рис. 18. Код підключення локального сервера

24. У файлі «index.js» вказуємо налаштування блоків шаблонізатора hbs (рис. 19)

```
app.engine("hbs", expressHbs( //налаштування блочної структури
    {
        layoutsDir: "views", //каталог для головного файлу index.hbs
        defaultLayout: "index",
        extname: "hbs",
        partialsDir: "views" //каталог для використаних блоків, можна підкаталог
    }
))
app.set("view engine", "hbs");
```

Рис. 19. Код налаштування блоків шаблонізатора hbs

25. У файлі «index.js» прописуємо код генерації головної сторінки по маршруту «/» (рис. 20):

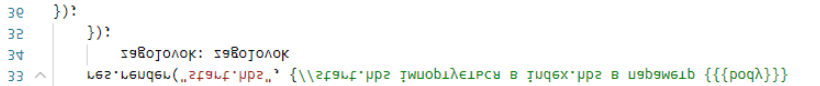


Рис. 20. Код генерації головної сторінки

26. Відкрийте файл стартової сторінки «start.hbs» та розмістіть у ньому форму для майбутньої авторизації на сайті:

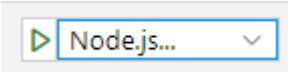
```
<div class="container-fluid" style="min-height: 50vh">

    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-6 col-lg-
3 text-center mx-auto">
            <h4 class="text-primary">Авторизація</h4>
            <form name="login">
                <div class="form-group">
                    <label for="name_user">Логін (E_mail):</
label>
```

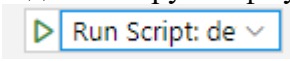
```
                    <input type="email" class="form-
control" name="name_user" placeholder="Введіть логін">
                </div>
                <div class="form-group">
                    <label for="parol">Пароль:</label>
                    <input type="password" class="form-
control" name="parol" placeholder="Введіть пароль">
                </div>
                <input type="button" class="btn btn-
primary mx-2 my-2" value="Зберегти" id="btn"/>
            </form>
        </div>
    </div>

</div>
```

27. У файлі «index.js» прописуємо код запуску сервера (рис. 21).

```
38  const PORT = process.env.PORT || 3000 //якщо порт є то його, інакше стаавимо 3000
39  app.listen(PORT, function(){
40      console.log("Сервер");
41  });
```

Рис. 21. Код запуску сервера

28. Збережіть зміни у всіх файлах та запустіть сервер. Для цього виконайте команду головного меню Visual Studio Code: «Виконати»-«Запустити налогодження»-«Node.js». Далі перейдіть у розділ «Запуск та налагодження» та в розділі вибору конфігурацій виберіть - 



.

29. Введіть у браузері адресу сервера http://localhost:3000/ та переконайте у правильності відображення стартової сторінки Web-сайту.

30. Перевірте адаптивність сайту у браузері, зокрема приховування блоку заголовку на малих екранах. Наприклад, у Google Chrome в контекстному меню сторінки «index.php» вибрати «Переглянути код», далі натиснути  та у панелі, що з'явилася подивитися як виглядає сторінка на різних гаджетах 

, або задати власні параметри екрану 

, або змінювати розмір шляхом перетягування вертикального розділювача.

31. Показати та захистити роботу викладачеві.

Лабораторна робота №5
Розробка програмного забезпечення КІС

Хід роботи

1. Запустіть сервер БД MySQL (XAMPP).
2. Відкрийте у Visual Studio Code проект Web-сайту про облік конференцій (з попередньої лабораторної роботи).
3. У каталозі «views» створіть пусті файли для роботи з таблицею «Кафедри» БД MySQL: «kafedry.hbs», «create.hbs», «edit.hbs.

Відображення даних головної таблиці БД MySQL

4. Відкрийте файл шаблону сторінки для відображення даних про кафедри «kafedry.hbs».
5. Розмістіть у ньому код основного блоку (рис. 22).

```
<div class="container-fluid" style="min-height: 50vh">
    <div class="row">
        <div class="col-12 text-center">
            <h1 class="text-primary">Кафедри</h1>
        </div>
    </div>
</div>
```

Рис. 22. Код основного блоку сторінки Web-сайту

6. Запишіть у даному файлі в межах коду основного блоку наступний код динамічної таблиці, яка відображатиме дані про кафедри (рис. 23).

```
<div class="text-right">
            <a href="/create" class="btn btn-primary btn-sm my-2">Додати запис </a>
</div>
<div class="table-responsive">
    <table class="table table-hover table-bordered">
        <tbody>
            <tr class="text-dark table-primary">
                <th scope="col" ></th>
                <th scope="col" ></th>
                <th scope="col" >Код кафедри</th>
                <th scope="col" >Назва</th>
                <th scope="col" >Розташування</th>
            </tr>
            {{#each kafedry}} <!-- початок циклу для динамічного додавання рядків -->
            <tr class="text-dark">
                <td>
                    <form action="delete/{{this.id_kafedry}}" method="POST" style="display:inline;">
                        <input type="submit" value="Видалити" class="btn btn-warning btn-sm"/>
                    </form>
                </td>
                <td>
                    <a href="/edit/{{this.id_kafedry}}" class="btn bg-success btn-sm">Редагувати</a>
                </td>
                <td>
                    {{this.id_kafedry}}
                </td>
                <td>
                    {{this.nazva_kafedry}}
                </td>
                <td>
                    {{this.roztashuvannya}}
                </td>
            </tr>
            {{/each}} <!-- кінець циклу для динамічного додавання рядків -->
        </tbody>
    </table>
</div>
```

Рис. 23. Код динамічної таблиці

7. У файлі «index.js» запишіть код генерації сторінки відображення даних про кафедри (рис. 24).

```
app.get("/kafedry", function(req, res){
    pool.query("SELECT\n" +
        "kafedry.id_kafedry,\n" +
        "kafedry.nazva_kafedry,\n" +
        "kafedry.roztashuvannya\n" +
        "FROM\n" +
        Complexity is 3 Everything is cool!
        "kafedry\n", function(err, data) {
        if(err) return console.log(err);
        res.render("kafedry.hbs", {
            kafedry: data,
            zagolovok: zagolovok
        });
    });
});
```

Рис. 24. Код генерації сторінки відображення даних

8. Збережіть зміни. Запустіть сервер та перевірте у браузері відображення даних про кафедри (пункт головного меню сайту «Довідники»-«Кафедри»).

Додавання, редагування та видалення даних головної таблиці БД MySQL

9. Зупиніть сервер.

10. Відкрийте файл шаблону сторінки для додавання нового запису про кафедри «create.hbs».

11. Розмістіть у даному файлі форму для додавання нового запису (рис. 25).

```
<div class="container" style="min-height: 50vh">
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-6 col-lg-6 text-center mx-auto">
            <h2 class="text-primary">Кафедри <br> (новий запис)</h2>
          <form method="post">
              <div class="form-group">
                <label for="nazva_kafedry">Назва кафедри</label>
                <input type="text" class="form-control" name="nazva_kafedry" placeholder="Введіть назву кафедри">
               </div>
              <div class="form-group">
                <label for="roztashuvannya">Розташування</label>
                <input type="text" class="form-control" name="roztashuvannya" placeholder="Введіть розташування кафедри">
               </div>
              <button type="submit" class="btn btn-primary mx-2 my-2">Зберегти</button>
              <button type="reset" class="btn btn-primary">Очистити</button>
          </form>
      </div>
  </div>
</div>
```

Рис. 25. Код форми для додавання нового запису

12. У файлі «index.js» запишіть код переходу на форму додавання запису (рис. 26).

```
app.get("/create", function(req, res){
    res.render("create.hbs", {
        zagolovok: zagolovok
    });
});
```

Рис. 26. Код переходу на форму додавання запису

13. У файлі «index.js» запишіть код на виконання запиту на додавання (маршрут той самий «/create», але метод «POST» – отриманий з форми файлу «create.hbs») (рис. 27).

```
app.post("/create", urlencodedParser, function (req, res) {
    if(!req.body) return res.sendStatus(400);
    const nazva_kafedry = req.body.nazva_kafedry;
    const roztashuvannya = req.body.roztashuvannya;
    pool.query("INSERT INTO kafedry (nazva_kafedry, roztashuvannya) VALUES (?,?)",
    [nazva_kafedry, roztashuvannya], function(err, data) {
        if(err) return console.log(err);
        res.redirect("/kafedry");
    });
});
```

Рис. 27. Код на виконання запиту на додавання

14. Збережіть зміни. Запустіть сервер та перевірте у браузері додавання нової кафедри.

15. Зупиніть сервер.

16. Відкрийте файл шаблону сторінки для редагування запису про кафедри «edit.hbs».

17. Розмістіть у даному файлі форму для редагування запису (рис. 28).

```
<div class="container" style="min-height: 50vh">
    <div class="row">
        <div class="col-6 text-center mx-auto">
            <h2 class="text-primary">Кафедра
                &#8220;{{kafedry.nazva_kafedry}}&#8221;
                <br>(редагування)
            </h2>
          <form action="/edit" method="post">
            <input type="hidden" name="id_kafedry" value="{{kafedry.id_kafedry}}" />
            <div class="form-group">
                <label for="nazva_kafedry">Назва кафедри</label>
                <input type="text" class="form-control" name="nazva_kafedry" placeholder="Введіть назву кафедри"
                value = "{{kafedry.nazva_kafedry}}">
              </div>
            <div class="form-group">
                <label for="roztashuvannya">Розташування</label>
                <input type="text" class="form-control" name="roztashuvannya" placeholder="Введіть розташування кафедри"
                value = "{{kafedry.roztashuvannya}}">
            </div>
            <button type="submit" class="btn btn-primary mx-2 my-2">Зберегти</button>
            <button type="reset" class="btn btn-primary">Очистити</button>
        </form>
    </div>
</div>
</div>
```

Рис. 28. Код форми для редагування запису

18. У файлі «index.js» запишіть код переходу на форму редагування запису (рис. 29).

```
app.get("/edit/:id_kafedry", function(req, res){
    const id_kafedry = req.params.id_kafedry;
    pool.query("SELECT\n" +
        "kafedry.id_kafedry,\n" +
        "kafedry.nazva_kafedry,\n" +
        "kafedry.roztashuvannya\n" +
        "FROM\n" +
        Complexity is 3 Everything is cool!
        "kafedry WHERE id_kafedry=?", [id_kafedry], function(err, data) {
        if(err) return console.log(err);
        res.render("edit.hbs", {
            kafedry: data[0],
            zagolovok: zagolovok
            });
    });
});
```

Рис. 29. Код переходу на форму редагування запису

19. У файлі «index.js» запишіть код на виконання запиту на редагування (маршрут той самий «/edit», але метод «POST» – отриманий з форми файлу «edit.hbs») (рис. 30).

```
app.post("/edit", urlencodedParser, function (req, res) {

    if(!req.body) return res.sendStatus(400);
    const nazva_kafedry = req.body.nazva_kafedry;
    const roztashuvannya = req.body.roztashuvannya;
    const id_kafedry = req.body.id_kafedry;

    pool.query("UPDATE kafedry SET nazva_kafedry=?, roztashuvannya=? WHERE id_kafedry=?",
    Complexity is 3 Everything is cool!
    [nazva_kafedry, roztashuvannya, id_kafedry], function(err, data) {
        if(err) return console.log(err);
        res.redirect("/kafedry");
    });
});
```

Рис. 30. Код виконання запиту на редагування

20. Збережіть зміни. Запустіть сервер та перевірте у браузері редагування даних про кафедри.

21. У файлі «index.js» запишіть код на виконання запиту на видалення (маршрут «/delete» метод «POST» – отриманий з форми файлу «kafedry.hbs», кнопка «Редагувати») (рис. 31).

```
app.post("/delete/:id_kafedry", function(req, res){

    const id_kafedry = req.params.id_kafedry;
    Complexity is 3 Everything is cool!
    pool.query("DELETE FROM kafedry WHERE id_kafedry=?", [id_kafedry], function(err, data) {
        if(err) return console.log(err);
        res.redirect("/kafedry");
    });
});
```

Рис. 31. Код на виконання запиту на видалення

22. Збережіть зміни. Запустіть сервер та перевірте у браузері видалення даних про вибрану кафедру.

Виокремлення частин сервера в окремі файли

23. У каталозі «konferentsiyi_PIB» створюємо пустий файл «kafedry.js» для виокремлення частини сервера, що стосується сторінок кафедр.

24. У файлі «kafedry.js» прописуємо константи (можна скопіювати з «index.js»).

25. У файлі «index.js» пропишіть глобальні змінні (після коду створення підключення до БД):

```
global.zagolovok =zagolovok
global.pool =pool
```

26. У файлі «kafedry.js» отримайте значення глобальних змінних:

```
zagolovok=global.zagolovok
pool=global.pool
```

27. У файлі файл «kafedry.js» створіть функцію експорту «app»:

```
module.exports = function(app)
```

28. З файлу «index.js» в дану функцію експорту перемістіть коди всіх маршрутів, що стосуються даних про кафедри (відображення, редагування, видалення), а замість вирізаного коду у файлі «index.js» розмістіть:

```
const kafedry = require('./kafedry')
kafedry(app)
```

29. Збережіть зміни. Запустіть сервер та перевірте у браузері коректну роботу з даними про кафедру.

Створення Web-сторінок Node.js для роботи з підпорядкованими таблицями БД MySQL

30. У файлі навігації «menu.hbs» відкоригуйте код таким чином, щоб при натисненні на пункт головного меню сайту «Співробітники» виконувався маршрут «/spivrobitnyky».

31. У каталозі «views» створіть пусті файли для роботи з таблицею «Кафедри» БД MySQL: «spivrobitnyky.hbs», «spivrobitnyky_create.hbs», «spivrobitnyky_edit.hbs».

32. Відкрийте файл шаблону сторінки для відображення даних про співробітників «spivrobitnyky.hbs».

33. Розмістіть у ньому код основного блоку.

34. На сторінці «spivrobitnyky.hbs» під надписом «Співробітники» додайте форму Boostrap для вибору кафедри (рис. 32).

```
<div class="col-6 text-center mx-auto">
    <form method="post" class="form-inline">
        <div class="form-group mx-auto">
            <div class="form-group">
                <div class="input-group-text">по кафедрі</div>
                <div class="form-group">
                    <input list="id" class="form-control" name="nazva_kafedry"
                    placeholder="Виберіть кафедру" autocomplete="off">
                </div>
                <datalist id="id">
                    {{#each spivrobitnyky}}
                    <option value="{{this.nazva_kafedry}}"></option>
                    {{/each}}
                </datalist>
            </div>
            <div class="form-group">
                <button type="submit" class="btn btn-primary mx-1 my-2">Показати</button>
            </div>
        </div>
    </form>
</div>
```

Рис. 32. Код форми Boostrap для вибору кафедри

35. У каталозі «konferentsiyi_PIB» створюємо пустий файл «spivrobitnyky.js» для виокремлення частини сервера, що стосується сторінок про співробітників. Прописуємо в ньому потрібні константи та глобальні змінні, створюємо функцію експорту «app» (аналогічно до «kafedry.js»).

36. У файлі «spivrobitnyky.js» у функції експорту записуємо код по маршруту «/spivrobitnyky», метод GET).

37. У файлі «index.js» імпортуйте функцію «app» з «spivrobitnyky.js»:

```
const spivrobitnyky = require('./spivrobitnyky')
spivrobitnyky(app)
```

38. Збережіть зміни. Запустіть сервер та перевірте у браузері відображення назв кафедр у полі «Виберіть кафедру».

39. У файлі «spivrobitnyky.hbs» відкоригуйте код таким чином, щоб форма для вибору кафедри відображалася тільки тоді, коли значення змінної «spivrobitnyky_visible = false», в іншому випадку показуємо назву вибраної кафедри.

40. Для відображення даних про співробітників по вибраній кафедрі використайте компонент Bootstrap Cards (карточка – забезпечує гнучкий та розширюваний вміст контейнера з безліччю варіантів та опцій) з кодом у файлі «spivrobitnyky.hbs» (рис. 33).

```
{{#if spivrobitnyky_visible}}

<div class="text-right">
        <a href="/spivrobitnyky_create" class="btn btn-primary btn-sm my-2">Додати запис</a>
</div>


<div class="row">

    {{#each spivrobitnyky}}

    <div class="card text-center col-md-3">
        <div class="card-header"> Код співробітника {{this.id_spivrobitnyky}}</div>
    <div class="card-body">
            <h5 class="card-title">{{this.prizvyshche}}</h5>
            <p class="card-text">{{this.imya}} {{this.pobatkovi}}</p>
        </div>
        <div class="card-footer text-muted">
            <form action="spivrobitnyky_delete/{{this.id_spivrobitnyky}}" method="POST" style="display:inline;">
                    <input type="submit" value="Видалити" class="btn btn-warning btn-sm"/>
            </form>
        </a>
            <a href="/spivrobitnyky_edit/{{this.id_spivrobitnyky}}" class="btn bg-success btn-sm">Редагувати</a>
        </div>
    </div>

    {{/each}}
</div>

{{/if}}
```

Рис. 33. Код компоненту Bootstrap Cards

41. У файлі «spivrobitnyky.js» у функції експорту записуємо код по маршруту «/spivrobitnyky», метод POST,

тобто коли користувач вибрав потрібну кафедру та натиснув кнопку «Показати») (рис. 34).

```
app.post("/spivrobitnyky", urlencodedParser, function (req, res) {
    if(!req.body) return res.sendStatus(400);
    const nazva_kafedry = req.body.nazva_kafedry;
    pool.query("SELECT \
    spivrobitnyky.id_spivrobitnyky, \
    spivrobitnyky.id_kafedry, \
    spivrobitnyky.prizvyshche, \
    spivrobitnyky.imya, \
    spivrobitnyky.pobatkovi, \
    kafedry.nazva_kafedry \
    FROM \
    spivrobitnyky \
    Inner Join kafedry ON spivrobitnyky.id_kafedry = kafedry.id_kafedry \
    WHERE \
    Complexity is 3 Everything is cool!
    kafedry.nazva_kafedry = ?", [nazva_kafedry], function(err, data) {
        if(err) return console.log(err);
        res.render("spivrobitnyky.hbs", {
            spivrobitnyky: data,
            zagolovok: zagolovok,
            spivrobitnyky_visible: true,
            nazva_kafedry: nazva_kafedry
        });
    });
});
```

Рис. 34. Код фільтрації даних

42. Збережіть зміни. Запустіть сервер та перевірте у браузері відображення співробітників по обраній кафедрі.

43. Відкрийте файл шаблону сторінки для додавання нового запису про співробітника «spivrobitnyky_create.hbs».

44. Розмістіть у даному файлі форму для додавання нового запису (користувачеві пропонується список кафедр, які занесені у базу даних).

45. У файлі «spivrobitnyky.js» запишіть код переходу на форму додавання запису (маршрут «/spivrobitnyky_create»).

46. У файлі «spivrobitnyky.js» самостійно запишіть код виконання запиту на додавання нового співробітника

(маршрут той самий «/spivrobitnyky_create», але метод «POST»).

47. Збережіть зміни. Запустіть сервер та перевірте у браузері додавання нового співробітника.

48. Відкрийте файл шаблону сторінки для редагування запису про співробітника «spivrobitnyky_edit.hbs».

49. Розмістіть у даному файлі форму для редагування запису (користувачеві пропонується список кафедр, які занесені у базу даних).

50. У файлі «spivrobitnyky.js» запишіть код переходу на форму редагування запису (маршрут «/spivrobitnyky_edit»).

51. У файлі «spivrobitnyky.js» самостійно запишіть код виконання запиту на редагування даних про співробітника (маршрут той самий «/spivrobitnyky_edit», але метод «POST»).

52. Збережіть зміни. Запустіть сервер та перевірте у браузері редагування даних про співробітника.

53. У файлі «spivrobitnyky.js» самостійно запишіть код виконання запиту на видалення даних.

54. Збережіть зміни. Запустіть сервер та перевірте видалення даних.

Створення Web-сторінок Node.js для роботи з підпорядкованими таблицями БД MySQL, що містять дати

55. У файлі навігації «menu.hbs» перевірте, щоб при натисненні на пункт головного меню сайту «Конференції» виконувався маршрут «/konferentsiyi».

56. У каталозі «views» створіть пусті файли для роботи з таблицею «Конференції» БД MySQL:

«konferentsiyi.hbs», «konferentsiyi _create.hbs», «konferentsiyi_edit.hbs».

57. Відкрийте файл шаблону сторінки для відображення даних про конференції «konferentsiyi.hbs».

58. Розмістіть у ньому код основного блоку.

59. На сторінці «konferentsiyi.hbs» під надписом «Конференції» додайте форму Boostrap для вибору дати (рис. 35).

```
<div class="row">
    <div class="col-12">
    <form method="post" class="form-inline my-2">
            <div class="form-group mx-auto">
                <div class="form-group">
                    <label for="data_pochatku">з</label>
                    <input type="date" class="form-control mx-2" name="data_pochatku">
                </div>
                <div class="form-group">
                    <label for="data_zakinchennya">по</label>
                    <input type="date" class="form-control mx-2" name="data_zakinchennya">
                </div>
                <button type="submit" class="btn btn-primary">Показати</button>
            </div>
    </form>
    </div>
</div>
```

Рис. 35. Код форми для вибору дати

60. У каталозі «konferentsiyi_PIB» створюємо пустий файл «konferentsiyi.js» для виокремлення частини сервера, що стосується сторінок про конференції. Прописуємо в ньому потрібні константи та глобальні змінні, створюємо функцію експорту «app» (аналогічно до «kafedry.js»).

61. У файлі «konferentsiyi.js» у функції експорту записуємо код по маршруту «/konferentsiyi», метод GET).

62. У файлі «index.js» імпортуйте функцію «app» з «konferentsiyi.js» (аналогічно до попередніх прикладів).

63. Збережіть зміни. Запустіть сервер та перевірте у браузері відображення форми для вибору дат.

64. У файлі «konferentsiyi.hbs» відкоригуйте код таким чином, щоб форма для вибору дат відображалася

тільки тоді, коли значення змінної «konferentsiyi_visible = false», в іншому випадку показуємо вибрані дати.

65. Для відображення даних про конференції за вибраний період часу використайте компонент Bootstrap Cards з кодом у файлі «konferentsiyi.hbs» (рис. 36).

```
{{#if konferentsiyi_visible}}

<div class="text-right">
        <a href="/konferentsiyi_create" class="btn btn-primary btn-sm my-2">Додати запис</a>
</div>


<div class="row">

    {{#each konferentsiyi}}

    <div class="card text-center col-md-6">
        <div class="card-header"> Код конференції: <b>{{this.id_konferentsiyi}}</b></div>
    <div class="card-body">
            <h5 class="card-title">{{this.nazva_konferentsiyi}}</h5>
            <p class="card-text">Відповідальний: <b>{{this.prizvyshche}}</b></p>
            <p class="card-text"><h6>{{this.data_pochatku}} - {{this.data_zakinchennya}}</h6></p>
            <p class="card-text">Тривалість: <b>{{this.truvalist}} дн.</b></p>
        </div>
        <div class="card-footer text-muted">
            <form action="konferentsiyi_delete/{{this.id_konferentsiyi}}" method="POST" style="display:inline;">
                    <input type="submit" value="Видалити" class="btn btn-warning btn-sm"/>
            </form>
        </a>
            <a href="/konferentsiyi_edit/{{this.id_konferentsiyi}}" class="btn bg-success btn-sm">Редагувати</a>
        </div>
    </div>

    {{/each}}
</div>

{{/if}}
```

Рис. 36. Код сторінки «Конференції»

66. У файлі «konferentsiyi.js» у функції експорту записуємо код по маршруту «/konferentsiyi», метод POST, тобто коли користувач вибрав потрібні дати та натиснув кнопку «Показати») (рис. 37).

67. Збережіть зміни. Запустіть сервер та перевірте у браузері відображення конференцій за обрані дати.

68. Відкрийте файл шаблону сторінки для додавання нового запису про конференцію «konferentsiyi_create.hbs».

69. Розмістіть у даному файлі форму для додавання нового запису (користувачеві пропонується список

працівників, один з яких буде відповідальним за проведення конференції).

```
app.post("/konferentsiyi", urlencodedParser, function (req, res) {
    if(!req.body) return res.sendStatus(400);
    const data_pochatku = req.body.data_pochatku;
    const data_zakinchennya = req.body.data_zakinchennya;
    pool.query("SELECT \
    konferentsiyi.id_konferentsiyi, \
    konferentsiyi.nazva_konferentsiyi, \
concat(spivrobitnyky.prizvyshche,' ',Left(spivrobitnyky.imya,1),'. ', Left(spivrobitnyky.pobatkovi,1),'.') as prizvyshche, \
    DATE_FORMAT(konferentsiyi.data_pochatku,'%d.%m.%Y') AS data_pochatku, \
    DATE_FORMAT(konferentsiyi.data_zakinchennya, '%d.%m.%Y') AS data_zakinchennya, \
    (konferentsiyi.data_zakinchennya-konferentsiyi.data_pochatku) AS truvalist, \
    konferentsiyi.id_spivrobitnyky \
    FROM \
    konferentsiyi \
    Inner Join spivrobitnyky ON konferentsiyi.id_spivrobitnyky = spivrobitnyky.id_spivrobitnyky \
    WHERE \
    konferentsiyi.data_pochatku >=  ? AND \
    Complexity is 3 Everything is cool!
    konferentsiyi.data_zakinchennya <= ?", [data_pochatku, data_zakinchennya], function(err, data) {
        if(err) return console.log(err);
        res.render("konferentsiyi.hbs", {
            konferentsiyi: data,
            zagolovok: zagolovok,
            konferentsiyi_visible: true,
            data_pochatku: data_pochatku,
            data_zakinchennya: data_zakinchennya
        });
    });
});
```

Рис. 37. Код по маршруту «/konferentsiyi»

70. У файлі «konferentsiyi.js» самостійно запишіть код переходу на форму додавання запису (маршрут «/konferentsiyi_create», метод GET).

71. У файлі «konferentsiyi.js» самостійно запишіть код виконання запиту на додавання нової конференції (маршрут той самий «/konferentsiyi_create», але метод «POST»).

72. Збережіть зміни. Запустіть сервер та перевірте у браузері додавання нової конференції.

73. Відкрийте файл шаблону сторінки для редагування запису про конференцію «konferentsiyi_edit.hbs».

74. Розмістіть у даному файлі форму для редагування запису (користувачеві пропонується список працівників, один з яких буде відповідальним за проведення конференції).

75. У файлі «konferentsiyi.js» самостійно (аналогічно як по співробітникам) запишіть код переходу на форму редагування запису (маршрут «/konferentsiyi_edit», метод GET). Також потрібно врахувати формат передачі дат:

DATE_FORMAT(konferentsiyi.data_pochatku,'%Y-%m-%d') AS data_pochatku,
DATE_FORMAT(konferentsiyi.data_zakinchennya, '%Y-%m-%d') AS data_zakinchennya

76. У файлі «konferentsiyi.js» самостійно запишіть код виконання запиту на редагування даних про конференцію (маршрут той самий «/konferentsiyi_edit», але метод «POST»).

77. Збережіть зміни. Запустіть сервер та перевірте у браузері редагування даних про конференцію.

78. У файлі «konferentsiyi.js» самостійно запишіть код виконання запиту на видалення конференції.

79. Збережіть зміни. Запустіть сервер та перевірте видалення даних.

80. Показати та захистити роботу викладачеві.

Лабораторна робота №6
Реалізація функціоналу виведення інформації з КІС

Хід роботи

1. Запустіть сервер БД MySQL (XAMPP).

2. Відкрийте у Visual Studio Code проект Web-сайту про облік конференцій (з попередньої лабораторної роботи).

3. У каталозі «views» для формування звіту про 10 конференцій з найбільшою кількість учасників створіть новий файл з іменем «konf_top.hbs».

4. Розмістіть у ньому код таблиці, що відображатиме найкращі конференції (рис. 38).

```
<table width="100%" border="1">
  <tbody>
    <tr align="center" style="font-weight: 600;">
      <td>Назва конференції</td>
      <td>Дата початку</td>
      <td>Дата закінчення</td>
      <td>Кількість учасників</td>
    </tr>
      {{#each konf_top}}
    <tr>
      <td>{{this.nazva_konferentsiyi}}</td>
      <td align="center">{{this.data_pochatku}}</td>
      <td align="center">{{this.data_zakinchennya}}</td>
      <td align="center">{{this.kilkist}}</td>
    </tr>
  </tbody>
    {{/each}}
</table>
```

Рис. 38. Код таблиці кращих конференцій

5. Внесіть зміни у файл головного меню «menu.hbs», щоб при натисненні на пункт «Рейтинг конференцій» виконувався маршрут «/konf_top».

6. У файлі «index.js» запишіть код формування звіту про конференції (рис. 39).

7. Збережіть зміни, перевірте коректне відображення звіту про 10 найкращих конференцій.

8. Аналогічно до звіту «Рейтинг конференцій» забезпечте формування звітів «Рейтинг кафедр» (10 кафедр з найбільшою кількістю проведених конференцій) та

«Рейтинг місць проживання» (10 місць проживання з найбільшою кількістю учасників).

9. Збережіть зміни. Перевірте формування створених звітів.

10. Покажіть роботу викладачеві.

11. Показати та захистити роботу викладачеві.

```
app.get("/konf_top", function(req, res){
    pool.query("SELECT \
    konferentsiyi.nazva_konferentsiyi, \
    DATE_FORMAT(konferentsiyi.data_pochatku,'%d.%m.%Y') AS data_pochatku, \
    DATE_FORMAT(konferentsiyi.data_zakinchennya, '%d.%m.%Y') AS data_zakinchennya, \
    Count(uchasnyky.id_uchasnyky) AS kilkist \
    FROM \
    konferentsiyi \
    Left Join uchasnyky ON uchasnyky.id_konferentsiyi = konferentsiyi.id_konferentsiyi \
    GROUP BY \
    konferentsiyi.nazva_konferentsiyi, \
    konferentsiyi.data_pochatku, \
    konferentsiyi.data_zakinchennya \
    ORDER BY \
    kilkist DESC \
    Complexity is 3 Everything is cool!
    LIMIT 10", function(err, data) {
        if(err) return console.log(err);
        res.render("konf_top.hbs", {
            konf_top: data,
            zagolovok: zagolovok,
            boostrap_v: true
        });
    });
});
```

Рис. 39. Код формування звіту про конференції

Лабораторна робота №7
Розмежування прав доступу до КІС

Хід роботи

1. Запустіть сервер БД MySQL (XAMPP).

2. Відкрийте у Visual Studio Code проект Web-сайту про облік конференцій (з попередньої лабораторної роботи).

Розмежування прав доступу

3. Для розмежування прав доступу будемо використовувати таблицю «Учасники» з наступними полями: логін (e_mail, електронна пошта), parol (поле пароля), prava («admin» або «user»).

4. Внесіть зміни у структуру БД «konferentsiyi» таким чином, що для усіх нових учасників поле «prava» по замовчуванню було «user» (тобто простий користувач).

5. Для розмежування прав доступу встановіть модулі:

- npm install cookie-parser (1.4.5)
- npm install express-session (1.17.1)

6. У файлі «index.js» внесіть зміни таким чином, щоб підключення MySQL у було у вигляді функції та перемістіть у неї оголошення глобальної змінної «pool» (рис. 40).

```
14  function createStore() {
15  var pool = mysql.createPool({
16      connectionLimit: 5,
17      host: "localhost",
18      user: "root",
19      database: "konferentsiyi_x",
20      password: ""
21  });
22  global.pool =pool
23  }
```

Рис. 40. Код функції «createStore»

7. У файлі «index.js» нижче даної функції, проведіть ініціалізацію сесії (рис. 41).

```
var cookieParser = require("cookie-parser");  3.5K (gzipped: 1.4K)
var session = require("express-session");

app.use(bodyParser.json()) //для передачі даних з форми реєстрації

var store = createStore()

app.use(cookieParser())
app.use(session({ //створення сесії
    store: store,
    resave: false,
    saveUninitialized: true,
    secret: 'supersecret' //рядок підпису в cookie  ідентифікатор сесії
}))
```

Рис. 41. Код ініціалізації сесії

8. У файлі «start.hbs» запишіть скрипт, для передачі даних форми реєстрації (рис. 42).

```
<script>
    window.onload = function () {
        var btn =document.getElementById('btn')
        var inputs = document.getElementsByTagName('input')

        btn.addEventListener('click', function () {
            var xhr = new XMLHttpRequest()

            xhr.open('POST','/login') //переходимо по маршруту /login
            var userData = {
                username: inputs[0].value, //передаємо ім'я користувача
                password: inputs[1].value
            }

            xhr.onload = function () {
                location.href = '/konferentsiyi'; //після успішної авторизації переходимо на конференції

            }

            xhr.setRequestHeader('Content-Type','application/json')
            xhr.send(JSON.stringify(userData))

        })
    }
</script>
```

Рис. 42. Скрипт передачі даних форми реєстрації

9. У файлі «index.js» запишіть скрипт, що виконується після відправки даних форми (маршрут «/login») (рис. 43).

```
app.post("/login", function(req, res){

    var foundUser
    var e_mail=req.body.username
    var parol=req.body.password
    pool.query("SELECT\n" + //запит на відбір користувача по даних форми реєстрації
        "uchasnyky.e_mail,\n" +
        "uchasnyky.parol,\n" +
        "uchasnyky.prava\n" +
        "FROM\n" +
        "uchasnyky\n" +
        "WHERE\n" +
        "uchasnyky.e_mail =  ? AND\n" +
        Complexity is 6 It's time to do something...
        "uchasnyky.parol =  ?", [e_mail, parol] ,function(err, data) {
        if(err) return console.log(err);
        if (data.length>0) { //якщо користувач існує
            foundUser = data[0].e_mail
            global.prava =data[0].prava
        }
        console.log(global.prava);

    if (foundUser !== undefined) {
        req.session.username = foundUser
        console.log(req.session.username);
        res.send(req.session.username) //відправляємо ім'я користувача
        //return  res.redirect("/kafedry") //після успішної авторизації переходимо на кафедри
    } else {
        console.log('Error 2');
    }
    }); //завершення pool.query
});
```

Рис. 43. Код маршруту «/login»

10. У файлі «index.js» запишіть скрипт, що виконується для виходу з акаунту (маршрут «/logout») (рис. 44).

```
app.get("/logout", function(req, res){
    req.session.username =''
    res.redirect("/");
});
```

Рис. 44. Код маршруту «/logout»

11. Перевірте потрібний маршрут на пункт головного меню «Вихід».

12. Забороніть доступ не авторизованим користувачам до сторінки «Кафедри». Для цього внесіть зміни на маршрут «/kafedry» у файлі «kafedry.js» (рис. 45).

```
app.get("/kafedry", function(req, res){
    if (req.session.username) {
    pool.query("SELECT\n" +
        "kafedry.id_kafedry,\n" +
        "kafedry.nazva_kafedry,\n" +
        "kafedry.roztashuvannya\n" +
        "FROM\n" +
        Complexity is 3 Everything is cool!
        "kafedry\n", function(err, data) {
        if(err) return console.log(err);
        res.render("kafedry.hbs", {
            kafedry: data,
            zagolovok: zagolovok
        });
    });
} else {
    res.redirect("/"); //якщо неавторизувався до перехід до головної сторінки
}
```

Рис. 45. Код розмежування прав доступу

13. Збережіть зміни та перевірте неможливість відображення даних про кафедру неавторизованим користувачам.

14. Забороніть доступ не адміністраторам додавати нову кафедру. Для цього внесіть зміни на маршрут «/create» у файлі «kafedry.js».

15. Збережіть зміни та перевірте можливість додавання нової кафедри лише адміністраторам..

16. За наведеним вище прикладом розмежувати права доступу до інших Web-сторінок сайту:

- дані про кафедри редагують чи видаляють тільки адміністратори;
- дані про конференції переглядають усі користувачі (навіть не зареєстровані), а додають, редагують чи видаляють тільки адміністратори;

- дані про учасників переглядають усі зареєстровані користувачі, а додають, редагують чи видаляють тільки адміністратори;
- дані про співробітників переглядають, додають, редагують чи видаляють тільки адміністратори;
- дані про місця проживання переглядають, додають, редагують чи видаляють тільки адміністратори;
- звіти переглядають тільки адміністратори.

17. Збережіть зміни та перевірте функціонування сайту.

18. Показати та захистити роботу викладачеві.

Лабораторна робота №8
Розгортання КІС для роботи в комп'ютерних мережах

Хід роботи

1. Зареєструйтеся на безкоштовному хостингу Node.js: https://www.heroku.com/

2. Після реєстрації перейдіть на :
https://devcenter.heroku.com/articles/heroku-cli
і завантажте heroku під свою ОС.

3. Встановіть heroku.

4. Відкрийте у Visual Studio Code проект Web-сайту про облік конференцій (з попередньої лабораторної роботи).

5. Перевірте чи працює додаток heroku командою в терміналі:

heroku --help

6. Далі прописуємо команду в терміналі:

heroku login

натискаємо «Enter» і входимо в браузері у свій аккаунт.

7. У Visual Studio Code прописуємо команду в терміналі:

heroku create

видається URL нового сайту, наприклад:
https://intense-dusk-13885.herokuapp.com/

8. У браузері заходимо на:
https://dashboard.heroku.com/apps

9. У Visual Studio Code створюємо в каталозі «konferentsiyi_PIB» файл «.gitignore» та прописуємо каталог, який не потрібно завантажувати на хостинг:

node_modules

10. Закрийте Visual Studio Code.

11. Завантажте та встановіть Git (один з найбільших веб-сервісів для спільної розробки програмного забезпечення):
https://git-scm.com/downloads

12. Відкриваємо свій проект Visual Studio Code та прописуємо команду в терміналі (ініціалізуємо пустий репозиторій):

git init

13. Далі прописуємо команду в терміналі (завантажуємо свої файли у репозиторій):

git add .

14. Далі прописуємо команди в терміналі (вказуємо свою пошту та прізвище автора латинськими літерами):

git config --global user.email nouma@gmail.com

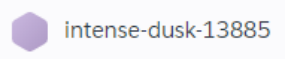
та:

git config --global user.name "Author"

15. Далі прописуємо команди в терміналі:

git commit -m "created"

16. На сайті https://dashboard.heroku.com/ переходимо по посиланню на свій сайт, наприклад:



17. Переходимо на вкладку Deploy та копіюємо код, наприклад:

heroku git:remote -a intense-dusk-13885

і виконуємо його у терміналі.

18. Далі прописуємо команду в терміналі:

git push heroku master

19. Розміщення файлів Web-сайту завершено. Для перевірки потрібно зайти на свій сайт по URL адресі, наприклад: https://intense-dusk-13885.herokuapp.com/

20. Підключення до БД MySQL можливе лише при зазначені даних про кредитну карту (https://dashboard.heroku.com/account) та встановлені Add-on ClearDB MySQL, вручну або за командою в терміналі:

heroku addons:add cleardb:ignite

21. Доступ до інформації про БД:

heroku config

та створити в корені сайту файл «composer.json» з наступним вмістом:

```
{
  "require": {
     "ext-mysql": "*"
  }
}
```

22. Для оновлення проекту (при внесені змін у файли сайту), потрібно почергово в терміналі виконати команди:

```
git add --u
git commit -m "changes that you made"
git push heroku master
```

19. Показати та захистити роботу викладачеві.

Рекомендована література

1. Petrov A. Database Internals: A Deep Dive into How Distributed Data Systems Work. USA : O'reilly Media, 2019. 376 p.
2. Фрімен Е., Робсон Е. Head First Патерни проєктування. Харків : Фабула, 2020. 688 с.
3. Gregg B. Systems Performance: Enterprise and the Cloud. Hoboken : Prentice Hall, 2013. 792 p.
4. Reis J., Housley M. Fundamentals of Data Engineering. Plan and Build Robust Data Systems. USA : O'reilly Media, 2022. 446 p.
5. Комп'ютерні мережі. Частина 1 : навчальний посібник / Б. Ю. Жураковський, І. О. Зенів. Київ : КПІ, 2020. 336 с.
6. Економічна інформатика : навч. посібник / П. М. Грицюк, В. І. Бредюк, В. Б. Василів, Т. Ю. Бабич, В. С. Волошин, О. І. Джоші, О. Л. Кардаш. Рівне : НУВГП, 2017. 311 с. URL: http://ep3.nuwm.edu.ua/6757
7. Russell J. T. Dyer. Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB 1st Edition. USA : O'reilly Media, 2015. 408 p.
8. Smirnova S., Tezuysal A. MySQL Cookbook. Solutions for Database Developers and Administrators. 4th Edition. USA : O'reilly Media, 2022. 922 p.
9. IT Consulting A Complete Guide. USA: The Art of Service - IT Consulting Publishing. 2020. 313 p.