­­РЕФЕРАТ

Пояснювальна записка до курсового проекта на тему «База даних магазину одягу Allium» складається з переліку умовних скорочень, вступу, чотирьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 37 сторінках друкованого тексту.

Пояснювальна записка має 9 сторінок додатків. Список використаних джерел містить 9 найменувань і займає 1 сторінку. В роботі наведено 19 рисунків. Загальний обсяг роботи – 46 сторінки.

У першому розділі був проведен теоретичний аналіз інформаційних потоків та особливостей предметної області дослідження, обґрунтовано вибір бази даних.

У другому розділі проведено проектування і розробка бази даних.

У третьому розділі було реалізовано підсистему обробки даних.

У четвертому розділі визначено загальну концепцію адміністрування бази даних.

Висновок містить в собі результати виконаної роботи при розробці бази даних «База даних магазину одягу Allium».

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: JAVASCRIPT, HTML, CSS, NODE.JS, MYSQL, EXPRESS, WEB-САЙТ, MVC, АДМІН, АВТОРИЗАЦІЯ, ІНТЕРНЕТ, ДАНІ.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 2](#_Toc74217527)

[ВСТУП 3](#_Toc74217528)

[РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ 4](#_Toc74217529)

[1.1 Аналіз інформаційних потреб та визначення предметної області дослідження 4](#_Toc74217530)

[1.2 Обґрунтування вибору засобів реалізації 6](#_Toc74217531)

[Висновки до розділу 8](#_Toc74217532)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ЗА НАПРЯМКОМ КУРСОВОЇ РОБОТИ 9](#_Toc74217533)

[2.1 Аналіз інформаційних процесів 9](#_Toc74217534)

[2.2 Проектування структури бази даних за напрямом курсової роботи 10](#_Toc74217535)

[2.3 Розробка математичної моделі та алгоритмів обробки даних в системи 16](#_Toc74217536)

[Висновки до розділу 18](#_Toc74217537)

[РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ ЗА НАПРЯМКОМ 19](#_Toc74217538)

[КУРСОВОЇ РОБОТИ 19](#_Toc74217539)

[3.1 Проектування інтерфейсу обробки даних 19](#_Toc74217540)

[3.2 Реалізація операцій обробки даних в БД за напрямом курсової роботи 22](#_Toc74217541)

[3.3 Організація звітності системи 25](#_Toc74217542)

[Висновки до розділу 30](#_Toc74217543)

[РОЗДІЛ 4 АДМІНІСТРУВАННЯ БАЗ ДАНИХ 31](#_Toc74217544)

[4.1 Розробка заходів захисту інформації в БД 31](#_Toc74217545)

[4.2 Налаштування параметрів роботи SQL-сервера 33](#_Toc74217546)

[Висновки до розділу 34](#_Toc74217547)

[ВИСНОВКИ 35](#_Toc74217548)

[ВИКОРИСТАНІ ДЖЕРЕЛА 36](#_Toc74217549)

[ДОДАТКИ 37](#_Toc74217550)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. БД – база даних
2. СУБД – система управління базами даних
3. ПК – персональний комп’ютер
4. HTML – мова розмітки гіпертексту
5. CSS – Таблиця каскадних стилів
6. JS – JavaScript
7. MVC — Model View Controll
8. CRUD — 4 основні функції управління даними «створення, читання, оновлення і вилучення»

# ВСТУП

Безумовно, тему створення веб-додатків можна вважати актуальною, адже відсоток використання веб-технологій помітно зростає в усіх сферах людського життя.

Використовуючи веб-ресурси, можна суттєво підвищити ефективність виконання буденних задач. Більш того, це найкраща реклама, інформаційний сторінка для будь-якої організації, це важливий крок компанії до розширення кордонів власного бізнесу та здобуття нової аудиторії.

Метою курсової роботи є дослідження особливостей проектування та реалізації БД, а також створення інтерфейсу та бази даних для роботи з онлайн крамницею одягу.

Об‘єктом дослідження є методи та засоби проектування БД та розробки CMS за визначеними предметними областями, а також прогнозування поведінки користувача.

Предметом дослідження є використання можливостей концепцій БД та СУБД для забезпечення інформаційних потеб предметної області.

# РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

* 1. Аналіз інформаційних потреб та визначення предметної області дослідження

База даних - це організований набір даних, які зазвичай зберігаються та мають доступ з комп’ютерної системи. Система управління базами даних - це програмне забезпечення, яке взаємодіє з кінцевими користувачами, програмами та самою базою даних для збору та аналізу даних. Програмне забезпечення СУБД додатково охоплює основні засоби, що надаються для адміністрування бази даних.

Саме тому створення веб сайту з використанням можливостей БД та системою адміністрування - це комплексна задача, що потребує відповідального ставлення та чіткого визначення задач, їх пріоритетності та порядок виконання, для забезпечення чіткого функціонування сайти, можливість підтримувати його працездатність та управління його контентом.

Для реалізації цього програмного продукту, перед нами стоять задачі:

1. Спроектувати та заповнити базу даних.

Для того, щоб досягти максимальної організованості структури даних, їх надійного збереження та полегшеного доступу для зміни і обробки взаємозалежної інформації великих обсягів, ми створимо БД та заповнимо її тестовими значеннями.

Для цього оберемо СУБД MySQL — вільна реляційна система управління базами даних. Гнучкість цієї системи забезпечується підтримкою великої кількості типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються новий функціонал. Також дана БД має API та конектори для багатьох мов програмування, наразі ми будемо використовувати мову програмування NODE.JS.

Для втілення можливості виконання над даними чотирьох основних операцій (створення, читання, оновлення і вилучення), реалізуємо власну CMS систему.

1. Створення інтерфейсу

Завдяки MVS – шаблону (поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача та модуль керування) реалізуємо можливість переглядати коротку інформацію про всі товари, їх детальну сторінку та систему адміністрування.

Для відділення даних представлення від виконуваного коду, будемо використовувати шаблонізатор PUG.

1. Забезпечення безпеки користувача

Для реалізації максимальної безпеки даних користувача, право редагувати, переглядати та змінювати данні необхідно надати лише адміністратору, передбачити шифрування паролю перед збереженням його у БД.

* 1. Обґрунтування вибору засобів реалізації

Основною частиною розробки додатків є робота з базою даних. На ринку представлений широкий вибір БД, кожна з яких має власні переваги та недоліки. Перед початком розробки веб-сайту, нам необхідно було обрати оптимальну СУБД, що відповідала б усім поставленим цілям. Для порівняльної характеристики було обрано три бази даних: MySQL, SQLite, MongoDB.

Таблиця 1.1 Порівняльна таблиця

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристика** | **MySQL** | **SQLite** | **Mongo** |
| Різноманітність типів даних | + | - | + |
| Портативність | - | + | + |
| Безпека | + | - | + |
| Необхідність високого рівень кваліфікації серед персоналу | - | - | + |
| Простота налаштування | + | + | + |
| Продуктивність | + | + | + |
| Можливість локалізації (кирилиця) | + | + | + |
| Безкоштовна ліцензія | + | + | + |
| Підтримка структури відносних даних | + | - | - |
| Наявність засобів для створення індексів і кластерів | + | - | + |
| Механізм блокування транзакцій під час запису | + | - | + |
| Імпорт/експорт таблиць БД | + | + | + |
| Підтримка утиліт резервування БД | + | - | - |
| Графічні інструменти | + | - |  |
| Інтеграція з іншими СУБД | + | + | - |
| Процедури, що зберігаються та тригери | + | - | +/- |
| Засоби аналізу | + | - | + |
| Адміністративне керування | + | - | + |
| Простота обслуговування | + | + | + |
| Одночасний доступ декількох користувачів | + | - | - |
| Відлагоджений механізм реплікації даних | + | - | + |
| Підтримка технології клієнт/сервер | + | - | + |

SQLite - автономне рішення, інтегрується безпосередньо в додаток, безсерверне рішення, легке налаштування. Файлова система робить її дуже портативною. Чудово підходить для розробки та тестування малих БД.

Недоліки: не забезпечує мережевий доступ. Не призначений для масштабних додатків. Відсутність управління користувачами.

MySQL, з іншого боку, є серверною БД, тому ми не можемо безпосередньо інтегрувати його з додатком, для цього потрібно зробити попередні налаштування, а потім підключити до додатка. Далі додаток буде використовувати протокол TCP / IP для надсилання та отримання інформації.

Плюси: потужний і здатний обробляти інформацію для додатків з високим трафіком. Багатий функціонал.

MongoDB - NoSQL, використовує подібні до JSON документи з необов'язковими схемами. Плюси: Без схем. Простота масштабування. При додавання більшої кількості машин отримуємо більше оперативної пам'яті.

Мінуси: розмір даних у MongoDB, як правило, вищий, менша гнучкість при запитах (відсутність приєднань).

SQLite, MongoDB і MySQL є широко використовуваними та перевіреними СУБД, проте після короткого порівняльного аналізу будо вирішено, що саме MySQL найбільше відповідає заданим критеріям.

Висновки до розділу

При виконанні даного розділу курсової роботи було проаналізовано поставлене перед нами завдання щодо розробки бази даних оптимальної для онлайн крамниці одягу та демонстрація її можливостей завдяки створеному інтерфейсу, який буде містити як клієнтську частину так і патель адміністратора, з якої буде відбуватися подальша робота зі створеною БД. На основі первинного розгляду поставленої перед нами задачі ми визначили напрямок подальшої розробки веб-застосунку.

Також ми виконали порівняльний аналіз СУБД MySQL, SQLite та MongoDB визначили їх переваги, недоліки та доцільність використання при різних умовах та поставлених цілей, обрали найбільш оптимальну БД для подальшої роботи.

Результатом проведеного аналізу є розробка технічного завдання – переліку основних функцій, етапі розробки БД, вимоги до неї та інтерфейсних особливостей вебсайту, визначені проблеми та знайдені методи для їх вирішення. Вивчено нові теоретичні відомості, необхідні для виконання поставлених перед нами задач.

# РОЗДІЛ 2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ЗА НАПРЯМКОМ КУРСОВОЇ РОБОТИ

2.1 Аналіз інформаційних процесів

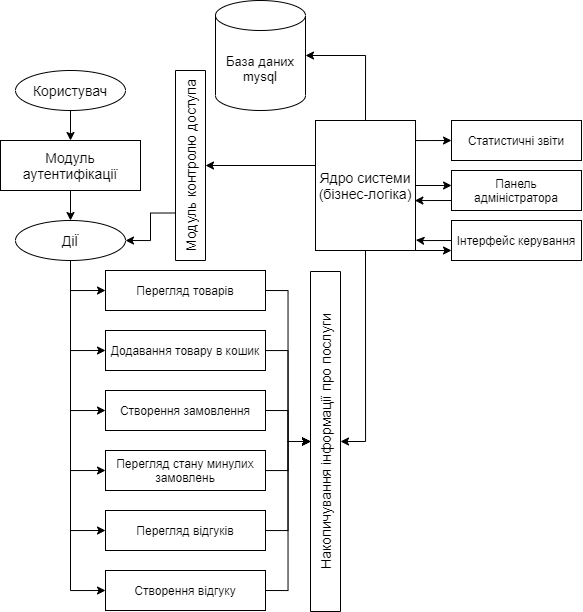


Рис.2.1 Структура автоматизованої інформаційної системи

Дана схема (Рис.2.1) ілюструє, що авторизований користувач може переглядати всі товари, їх детальні сторінки, додавати їх у кошик, створювати замовлення та переглядати статус власних минулих замовлень, а також створювати та редагувати відгуки до наявних товарів.

2.2 Проектування структури бази даних за напрямом курсової роботи

Після завершення аналізу інформаційних проектів та формування чітких задач, буди поставлені перед нами, переходимо до проектування структури бази даних інтернет-магазину одягу за допомогою обраних нами раніше засобів реалізації.

Внаслідок проектування було створено 19 таблиць:

* storage
* product\_storage
* product
* price
* category
* category\_ product
* property\_name
* property\_value
* property\_name\_value
* category\_property\_name\_value
* product\_category\_property\_name\_value
* group
* user\_group
* user
* review
* backet
* backet\_product
* order
* status

Розглянемо детально призначення таблиць та їх поля.

Для збереження даних про товар призначена таблиця “product”. В дану таблицю заносяться дані про назву товару, його опис, шлях до детального зображення та окремо, при необхідності шлях до фото попереднього перегляду зі значенням за замовчуванням, що вказує назву шаблонної обкладинки, поля – часові мітки встановлені на збереження моменту при створенні запису та часу його зміни, а також ідентифікатор ціни товару.

Фрагмент коду:

CREATE TABLE `product` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

`description` text NOT NULL,

`id\_price` int(11) NOT NULL,

`detail\_img` varchar(50) DEFAULT 'default.jpg',

`preview\_img` varchar(50) DEFAULT 'default.jpg',

`createdAt` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`updatedAt` datetime NOT NULL ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `id\_price` (`id\_price`),

FOREIGN KEY (`id\_price`) REFERENCES `price` (`id`)

);

Таблиця 2.1 Структура таблиці товарів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| Id | int(11) | + | - | Код товару |
| name | varchar(50) | - | - | Назва |
| description | text | - | - | Опис |
| id\_price | int(11) | - | + | Код ціни товару (зв’язаний з таблицею price) |
| detail\_img | varchar(50) | - | - | Шлях до детального зображення |
| preview\_img | varchar(50) | - | - | Шлях до зображення для попереднього перегляду |
| createdAt | datetime | - | - | Часова мітка створення запису |
| updatedAt | datetime | - | - | Часова мітка зміни запису |

Для збереження даних користувача призначена таблиця “user”. В дану таблицю заносяться дані про ім’я, пошту, телефон, пароль, який буде вподальшому зашифрований, країну, місто та адресу користувача.

Фрагмент коду:

CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`first\_name` varchar(20) COLLATE NOT NULL,

`last\_name` varchar(20) COLLATE NOT NULL,

`mail` varchar(20) COLLATE NOT NULL,

`phone` varchar(20) COLLATE NOT NULL,

`password\_hash` varchar(60) COLLATE NOT NULL,

`country` varchar(20) COLLATE NOT NULL,

`city` varchar(25) COLLATE NOT NULL,

`address` varchar(50) COLLATE NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `mail` (`mail`)

)

Таблиця 2.2 Структура таблиці користувачів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| Id | int(11) | + | - | Код користувача |
| first\_name | varchar(20) | - | - | Ім’я |
| last\_name | varchar(20) | - | - | Прізвище |
| mail | varchar(20) | - | - | Пошта |
| phone | varchar(20) | - | - | Телефон |
| password\_hash | varchar(60) | - | - | Пароль |
| country | varchar(20) | - | - | Країна |
| city | varchar(25) | - | - | Місто |
| address | varchar(50) | - | - | Адреса |

Для збереження даних складу призначена таблиця “storage”. В дану таблицю заносяться дані про назву, телефон та адресу складу.

Фрагмент коду:

CREATE TABLE `storage` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(20) COLLATE NOT NULL,

`phone` varchar(20) COLLATE NOT NULL,

`address` varchar(50) COLLATE NOT NULL,

PRIMARY KEY (`id`))

Таблиця 2.3 Структура таблиці складу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| Id | int(11) | + | - | Код користувача |
| name | varchar(20) | - | - | Назва |
| mail | varchar(20) | - | - | Пошта |
| address | varchar(50) | - | - | Адреса |

Для збереження даних про відносити таблиці товару та складу було створено таблицю “ product\_storage”.

Фрагмент коду:

CREATE TABLE `product\_storage` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`id\_storage` int(11) NOT NULL,

`id\_product` int(11) NOT NULL,

`amount` int(11) NOT NULL,

PRIMARY KEY (`id`),

KEY `id\_storage` (`id\_storage`),

KEY `id\_product` (`id\_product`),

CONSTRAINT `product\_storage\_ibfk\_1` FOREIGN KEY (`id\_storage`) REFERENCES `storage` (`id`),

CONSTRAINT `product\_storage\_ibfk\_2` FOREIGN KEY (`id\_product`) REFERENCES `product` (`id`))

Таблиця 2.4 Структура таблиці складу-товару

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| Id | int(11) | + | - | Код |
| id\_storage | int(11) | - | + | Код складу |
| id\_product | int(11) | - | + | Код товару |
| amount | int(11) | - | - | Кількість |

Для збереження даних про відгук до товару було створено таблицю “review” яка містить у собі такі поля, як куд користувача, код товару, числове значення відгуку, текст відгуку та дата створення.

Фрагмент коду:

CREATE TABLE `review` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`id\_user` int(11) NOT NULL,

`id\_product` int(11) NOT NULL,

`raiting` decimal(10,0) NOT NULL,

`text` text COLLATE NOT NULL,

`date` date NOT NULL,

PRIMARY KEY (`id`),

KEY `id\_user` (`id\_user`),

KEY `id\_product` (`id\_product`),

CONSTRAINT `review\_ibfk\_1` FOREIGN KEY (`id\_user`) REFERENCES `user` (`id`),

CONSTRAINT `review\_ibfk\_2` FOREIGN KEY (`id\_product`) REFERENCES `product` (`id`)

)

Таблиця 2.5 Структура таблиці рейтингу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва** | **Тип даних** | **ПК** | **ЗК** | **Опис поля** |
| Id | int(11) | + | - | Код |
| id\_user | int(11) | - | + | Код користувача |
| id\_product | int(11) | - | + | Код товару |
| raiting | decimal(10,0) | - | - | Рейтинг |
| text | text | - | - | Текст |
| date | date | - | - | Дата створення |

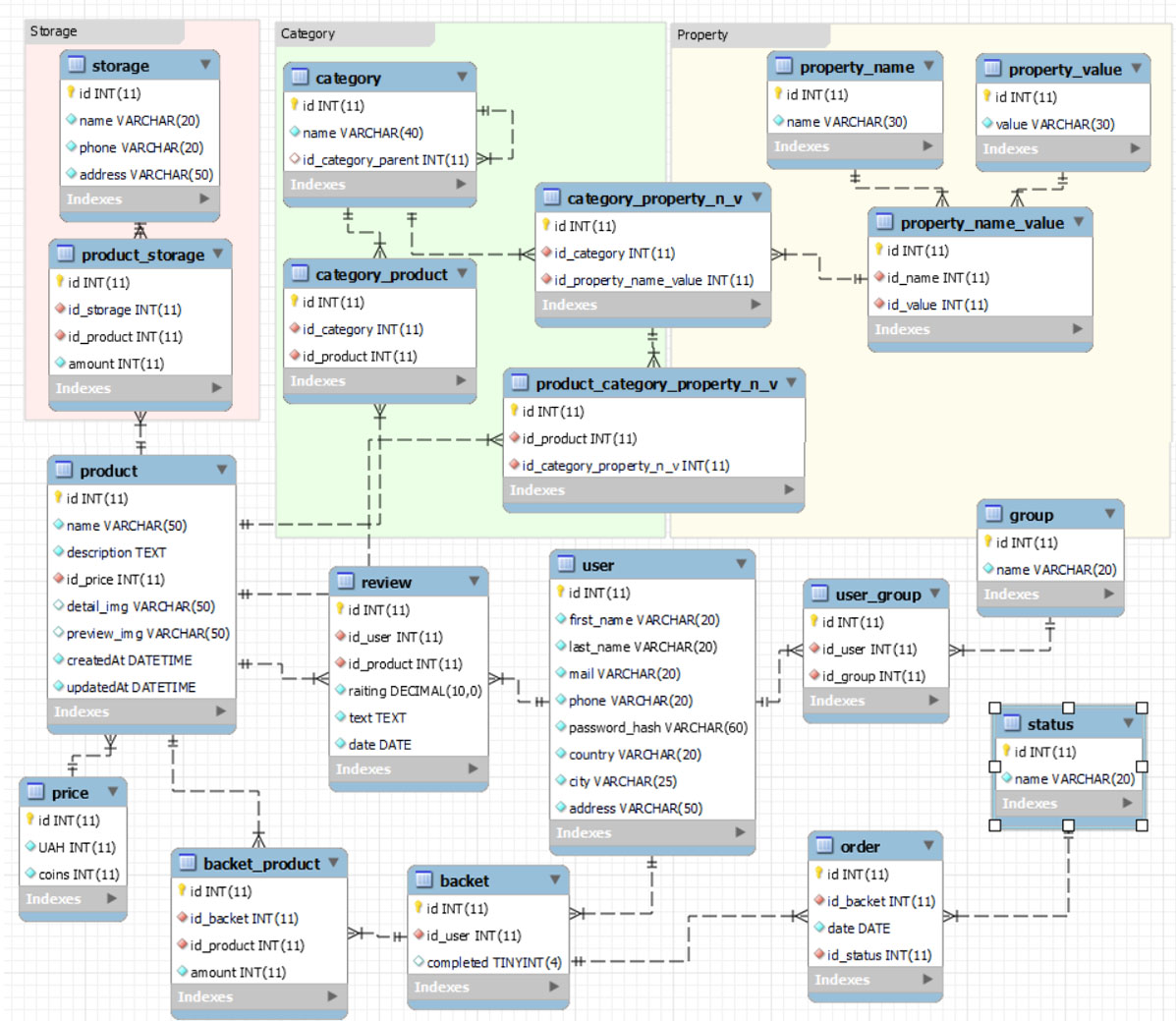


Рис.2.2 Діаграма сутність - зв’язок

2.3 Розробка математичної моделі та алгоритмів обробки даних в системи

Створимо декілька запитів (зберігаємих процедур) по обробці інформації для кращого та полегшеного збору даних по статистці продажу щоб підвищити ефективність, повноту та швидкість роботи працівників магазину щодо демонстрації товару та його продажу та надати можливість керівнику бачити повну статистики через адміністративну панель.

Перша процедура (GetProductsAmount()) дозволить нам дізнатися скільки товару залишилося на складі. Для цього ми приєднуємо таблиці товару та складів через таблиць товар-склад, за допомогою ідентифікаційних ключів.

Франмент коду:

SELECT p\_s.id\_product, p.name, SUM(p\_s.amount) AS amount

FROM product\_storage AS p\_s

RIGHT JOIN product AS p ON p\_s.id\_product= p.id

GROUP BY p\_s.id\_product, p.name;

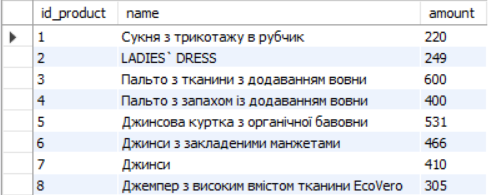


Рис.2.3 Результат виконання GetProductsAmount()

Наступна процедура (GetOrderProductAmount ()) дозволить нам отримати загальну кількість проданих одиниць по товару та ціну за один. Для цього до таблиці заказу ми приєднуємо таблиці корзини, яка в свою чергу дозволить нам дістатися до продуктів через таблицю товар-корзина. На останок до товару приєднуємо ціну.

Франмент коду:

SELECT p.id, p.name, CONCAT\_WS('.', pri.UAH, pri.coins) AS price\_one, SUM(b\_p.amount) AS amount

FROM allium.order AS o

RIGHT JOIN backet AS b ON o.id\_backet = b.id

RIGHT JOIN backet\_product AS b\_p ON o.id\_backet = b\_p.id\_backet

LEFT JOIN product AS p ON b\_p.id\_product = p.id

LEFT JOIN price AS pri ON p.id\_price = pri.id

GROUP BY p.id, p.name;

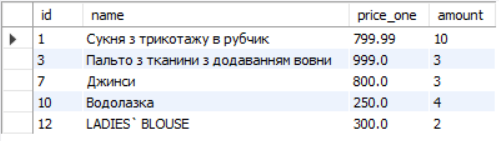


Рис.2.4: Результат виконання GetOrderProductAmount()

Висновки до розділу

При виконанні даного розділу курсової роботи було спроектовано базу даних оптимальну для роботи з онлайн крамницею одягу, проаналізовано інформаційні процеси роботи з додатком та розроблена математична модель та алгоритм обробки даних в системі.

Було визначено загальний алгоритм роботи авторизованого користувача та зазначено основні функції, які ми реалізували створивши 19 таблиць та створивши у них оптимальні поля, типи даних та зв’язки між ними, щоб отримати легку у розумінні, проте з широким функціональним списком базу даних.

Також для підвищення ефективності роботи працівників та більш коректної звітності було створено декілька зберігаємих процедур по обробці інформації, а саме задля перегляду загального обсягу товару на складі, та кількість одиниць збутого товару.

# РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБРОБКИ ДАНИХ ЗА НАПРЯМКОМ

# КУРСОВОЇ РОБОТИ

3.1 Проектування інтерфейсу обробки даних

Після проектування загального алгоритму програми ми створили структуру інтерфейсу магазину одягу (Рис. 2.1).

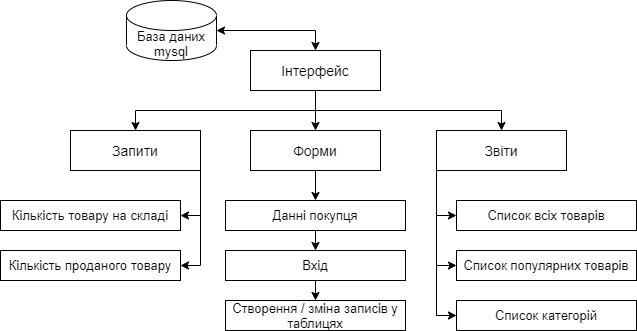


Рис.3.1 Структура інтерфейсу магазину одягу

Починаємо створювати динамічний веб-сайт, у якого сторінки формуються на стороні сервера динамічно, за запитом, зі сторінок-шаблонів та окремо зберігаємого вмісту (інформація, скрипти та ін.). Як правило, для відображення будь-якої кількості різних сторінок використовується одна сторінка-шаблон, у яку підгружається відповідний вміст, ще дозволяє виводити різні данні, редагуючи лише один шаблон.

Розглянемо алгоритм детальніше (Рис. 2.2). Сервер отримує запрошення від користувача і запускає обробку файлу-скрипта інтерпретато­ром. На цій стадії відбувається вибір необхідної інформації з бази даних та доповнення кожної сторінки, після чого готова сторінка передається користувачу.

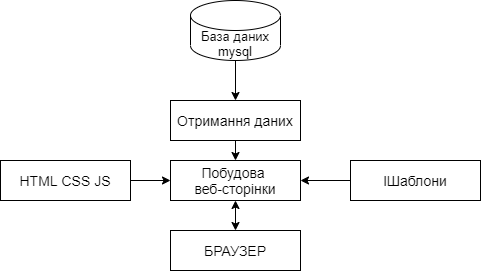


Рис.3.2 Структура побудови веб-сторінки

На клієнтській стороні використовується JavaScript, який може бути використаний як для мінімальних операцій (наприклад, перевірка правильності задання паролів при реєстрації на сайті), так і глобальних послідовностей.

Для реалізації роботи з даними ми будемо використовувати MVC схему, таким чином ми зможемо досягти того, що модифікація кожного компонента може здійснюватися незалежно. Розглянемо детальніше алгоритм роботи (Рис.2.3).

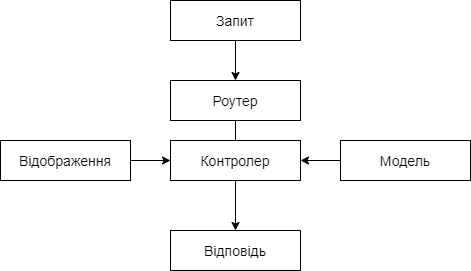


Рис.3.3 Реалізація роботи з даними

Контролер забезпечує «зв'язок» між користувачем та системою, слідкує та направляє дані від користувача до системи та навпаки. Використовує модель і представлення для реалізації необхідних дій.

Представлення відповідає за отримання необхідних даних із моделей та надсилає їх користувачу, не обробляє введені дані.

Модель надає дані та методи роботи з ними: запрошення в базу даних, перевірка коректності. Модель побудована таким чином, щоб відповісти на запрошення, змінивши свій стан.

3.2 Реалізація операцій обробки даних в БД за напрямом курсової роботи

Робота зі створеним програмним додатком починається з підключення до створеної БД за допомогою бібліотеки Sequelize - ORM-бібліотека для додатків Node.js, яка зіставляє таблиці в БД і відносин між ними з класами. При використанні Sequelize ми можемо не писати SQL-запити, а працювати з даними як зі звичайними об'єктами.

Фрагмент коду:

const sequelize = new Sequelize(process.env.DB\_NAME, process.env.DB\_USERNAME, process.env.DB\_PASSWORD, {

    host: process.env.DB\_HOST,

    dialect: process.env.DB\_DIALECT,

});

var initModels = require("./models/init-models");

var models = initModels(sequelize);

Після підключення, ініціалізуємо моделі в окрему папку, за потреби додаємо додатковий функціонал, як наприклад у моделі користувача ми створили функції для шифрування паролю (Додаток В). Переходимо до створення адміністративної панелі, за допомогою засобів бібліотеки AdminBro. Виконуємо певні конфігураційні налаштування та створюємо розтер з перевіркою зашифрованого паролю користувача.

Фрагмент коду:

const router = AdminBroExpress.buildAuthenticatedRouter(adminBro, {

    authenticate: async (email, password) => {

      const user = await User.findOne({ email })

        if (user) {

          if (user.validPassword(password)){

            return user

          }

        }

      return false

    },

    cookiePassword: 'session Key',

  });

Розглянемо приклади згенерованих SQL-запитів. Виборка серед товарів, в назві яких присутнє слово джинси.

Фрагмент коду: SELECT count(\*) AS `count` FROM `product` AS `product` WHERE (`product`.`name` LIKE '%Джинси%');

Створення нового запису

Фрагмент коду: INSERT INTO `product` (`id`,`name`, `description`, `id\_price`, `detail\_img`, `preview\_img`, `createdAt`, `updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?,?);

Видалення товару.

Фрагмент коду: DELETE FROM `product` WHERE `id` = 13

На головній сторінці створюємо карусель з чотирьох останніх добавлених товарів.

Фрагмент коду:

exports.getOverview = catchAsync(async (req, res, next) => {

  let products = await Product.findAll({

    include: [{

      model: Price,

      as: 'id\_price\_price',

    }],

    limit: 4,

    order: [['createdAt', 'DESC']]

  });

  res.status(200).render('index', {

    title: 'Головна',

    products

  });

});

Також задля виведення статистики створюємо функцію, де ми викликаємо процедуру отримання кількості збутого товару, результат якого ми далі передаємо у таблицю.

Фрагмент коду:

exports.getOrderAmount = catchAsync(async (req, res, next) => {

  sequelize

    .query('CALL GetOrderProductAmount();')

    .then(result => {

      res.status(200).render('getTotalAmount', {

        title: 'Головна',

        result

      });

    });

});

Розглянемо функцію перегляду детальної сторінки товару.

Фрагмент коду:

exports.getProduct = catchAsync(async (req, res, next) => {

  const product = await Product.findOne({

    where: {

      id: req.params.id

    },

    include: [{

      model: Price,

      as: 'id\_price\_price'

    }]

  });

  if (!product) {

    return next(new AppError('There is no product with that name.', 404));

  }

  res.status(200).render('product', {

    title: `${product.name}`,

    product

  });

});

3.3 Організація звітності системи

Після того, як користувач перейде на головну сторінку нашого веб-додатку, він бачить шапку сайту, де знаходяться посилання на головну сторінку, каталог, коротку інформацію про сайт та кнопка входу на адміністративну панель.

Далі користувач може розглянути банер з акціями та новинами сайту, коротку інформацію про створену онлайн крамницю (Рис.3.4).

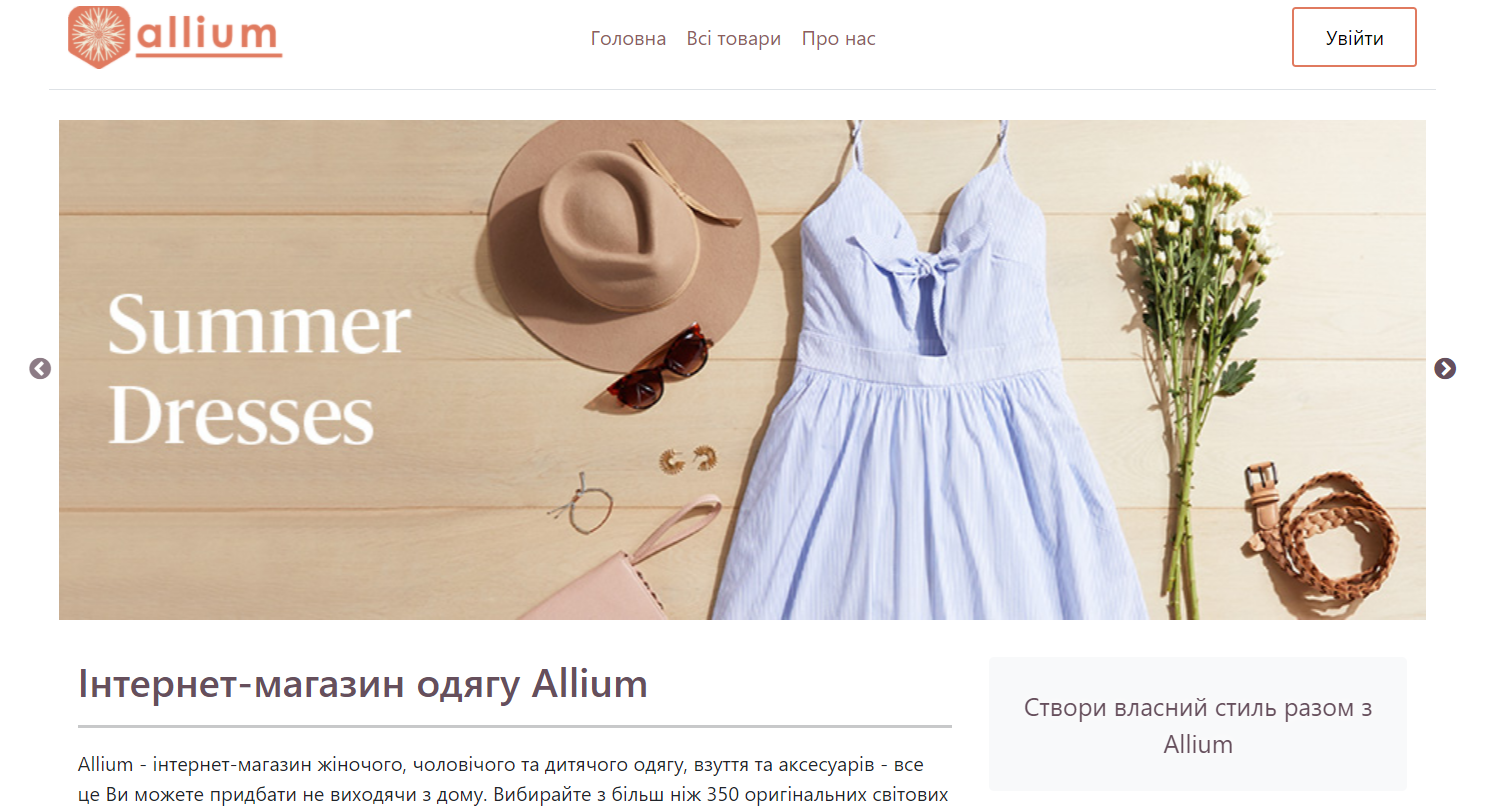


Рис.3.4 Шапка сайту та слайдер

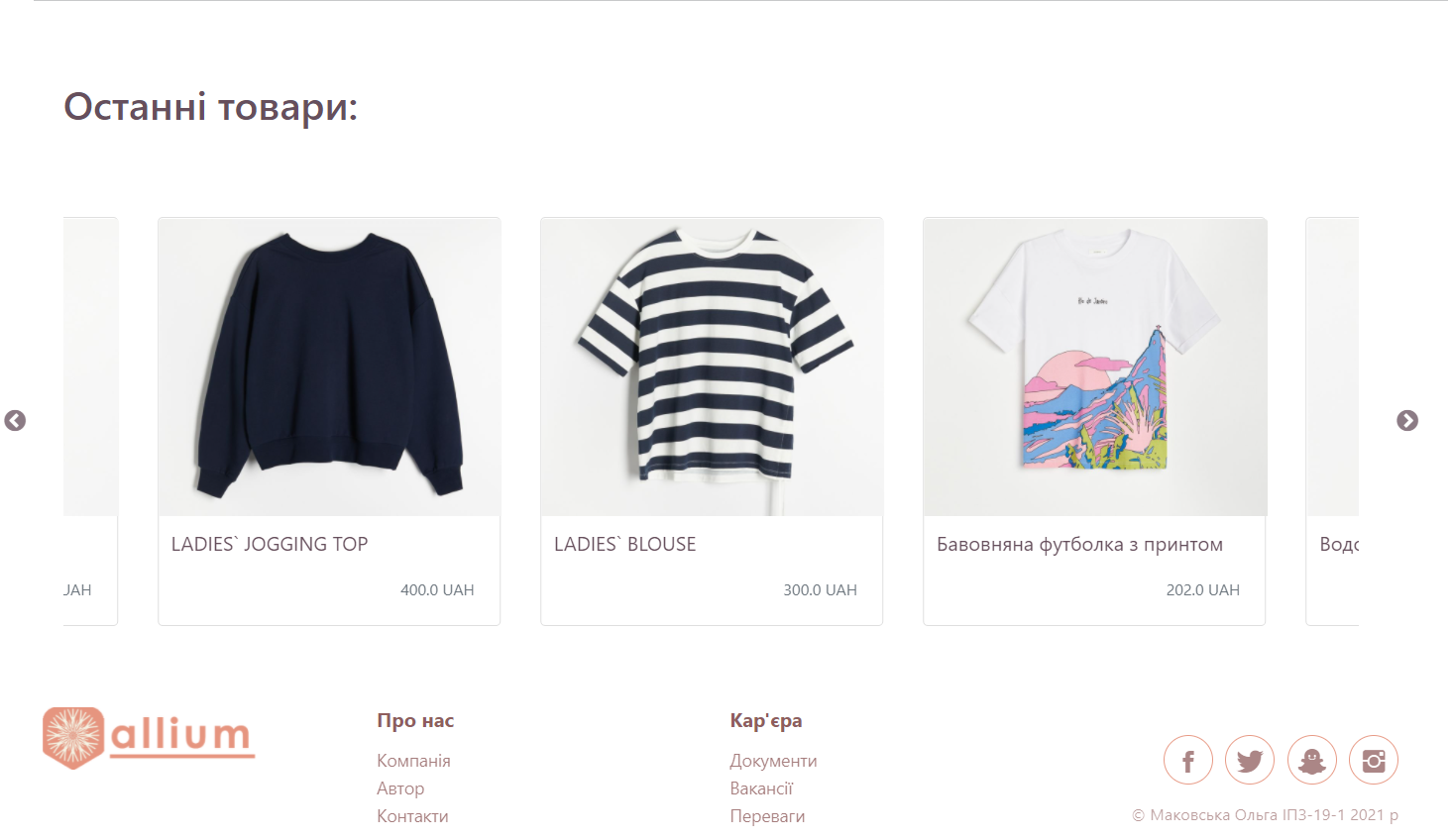


Рис.3.5 Нові товари

Після цього користувач може звернути увагу на один із чотирьох нових товарів, та розглянути футер сайту (Рис.3.5).

Сторінка каталог (Рис.3.6) складається з карток товарів, де відображається назва, ціна та фото елементу одягу,а також списку категорій, обравши одну зяких, ми можемо відфільтрувати товари.

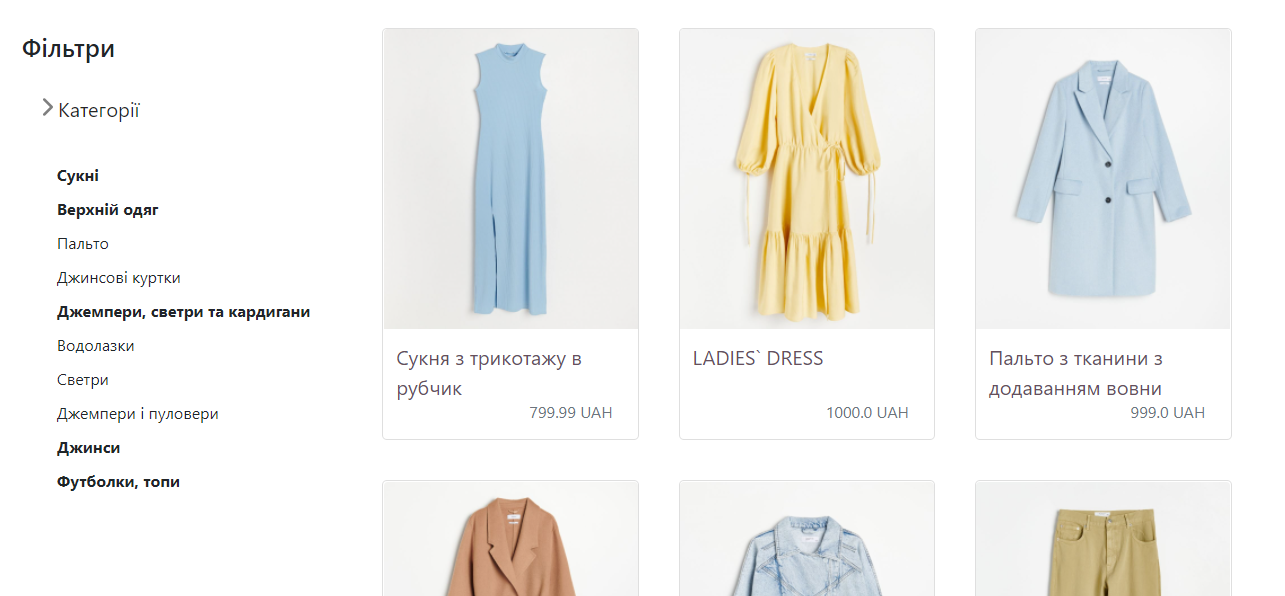


Рис.3.6 Каталог

Натиснувши на посилання – назву товару користувач переходить на детальну сторінку товару (Рис.3.7), де він може прочитати детальний опис товару та кнопку купити товар.

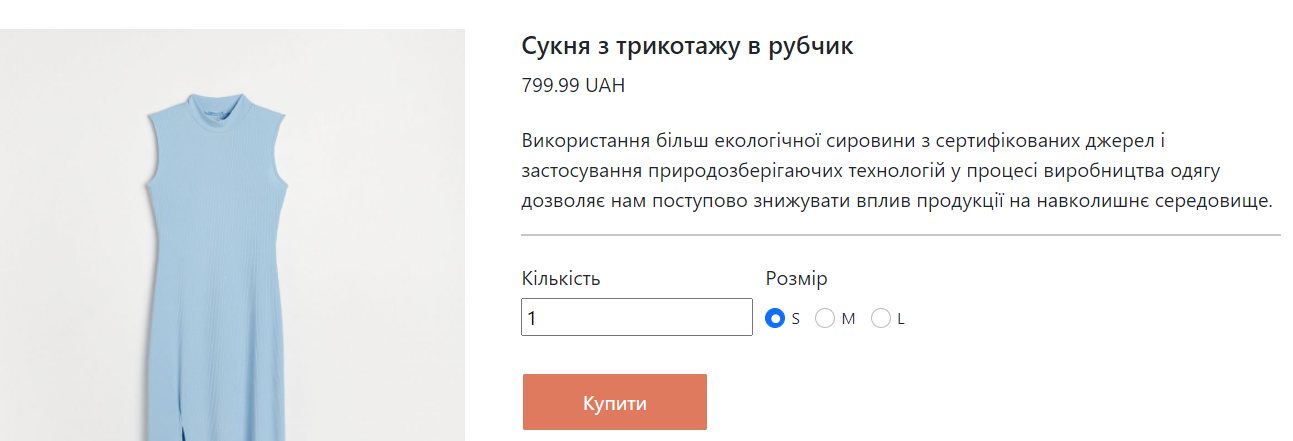


Рис.3.7 Детальна сторінка

Натиснувши на кнопку купівлі товари, перед користувачем з’являється коротка форма (Рис.3.8) з полями ім’я та поштова скринька, на яку буде відправлено повідомлення з деталями про замовлення.

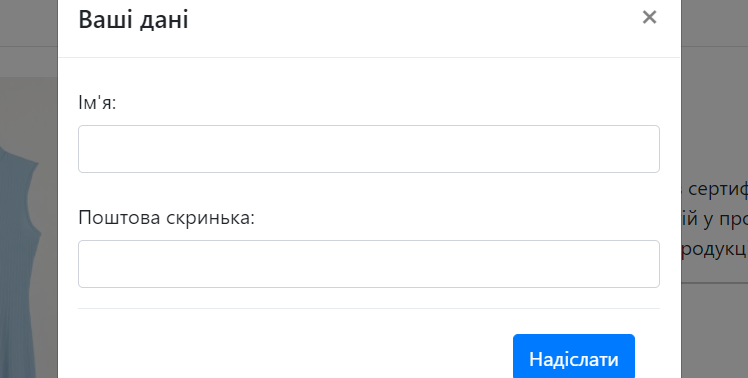


Рис.3.8 Купівля

Перед переходом до адміністративної панелі необхідно авторизуватися (Рис.3.9).

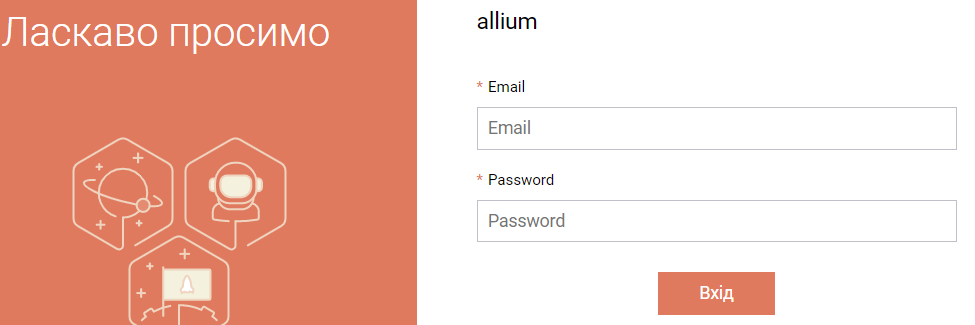


Рис.3.9 Авторизація

Далі перед користувачем з’являється головна сторінка (Рис.3.10) з посиланням на данні статистики та меню з посиланням на інформацію по всім таблицям зі створеної БД.

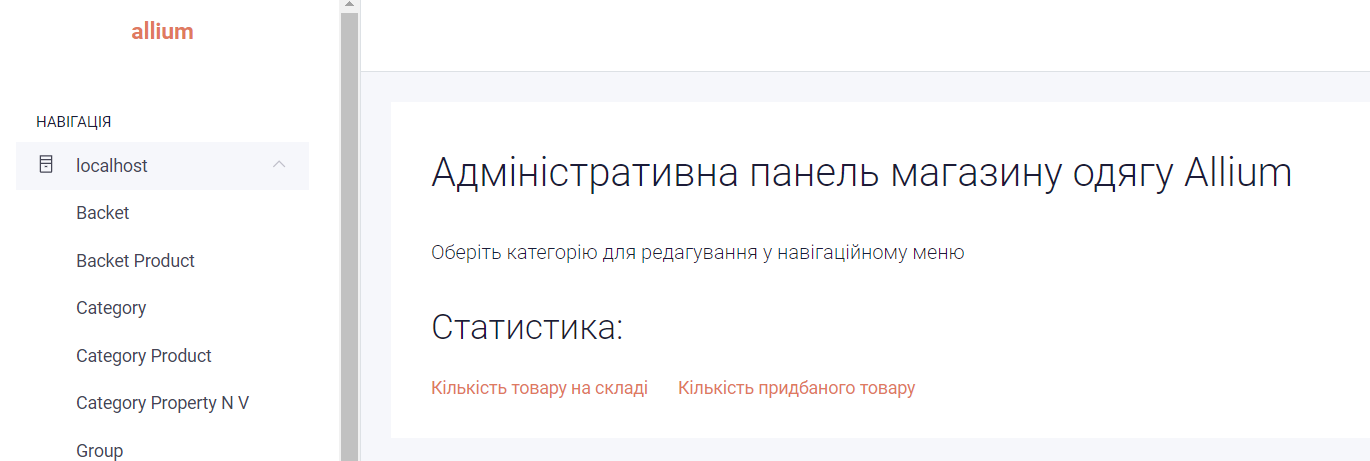


Рис.3.10 Адмін панель

Розглянемо основний функціонал адміністративної панелі на прикладі таблиці товарів. Спочатку перед користувачем з’являються всі створені записи з обраними полями (Рис.3.11). Далі ми можемо виконати фільтрацію (Рис.3.12), видалити, змінити (Рис.3.13), переглянути детальну інформацію про наявний товар або створити новий.

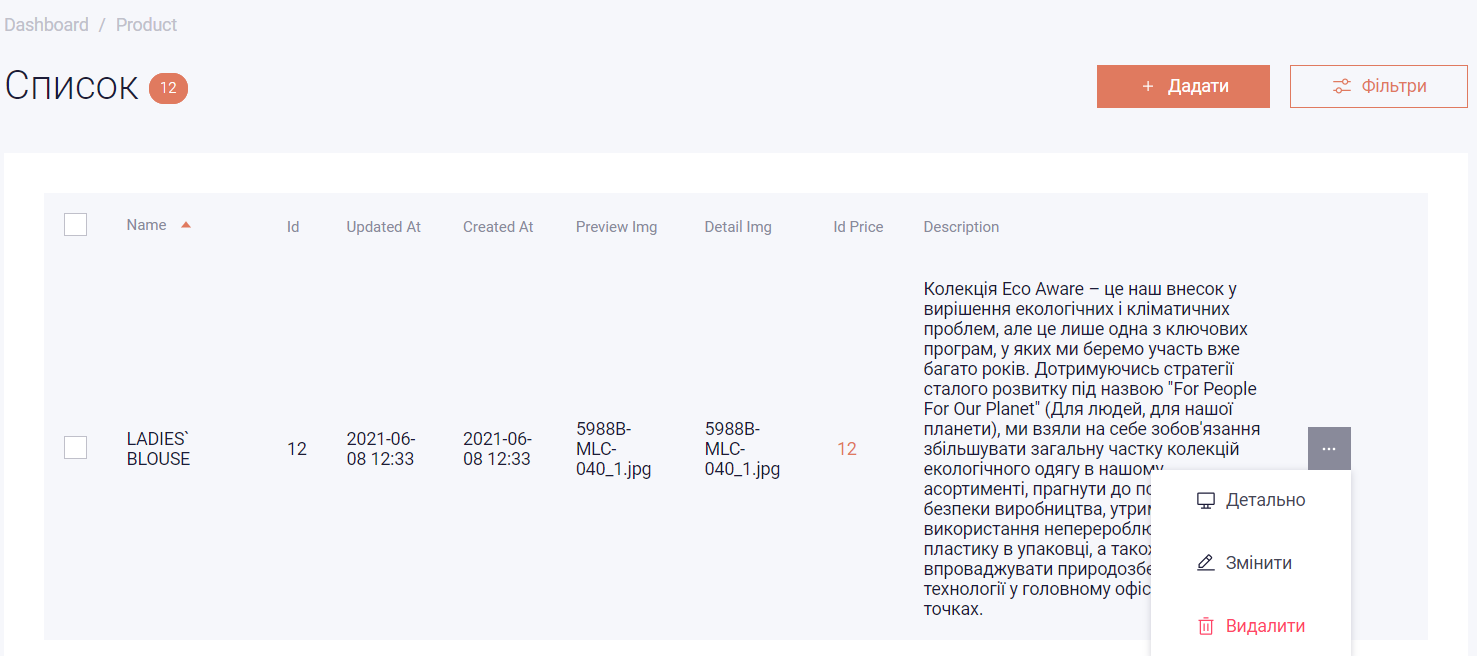


Рис.3.11 Всі товари

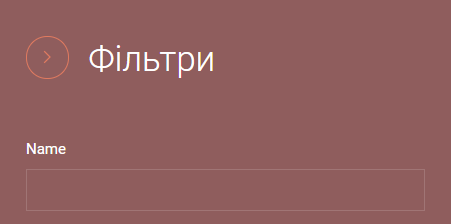


Рис.3.12 Фільтрація

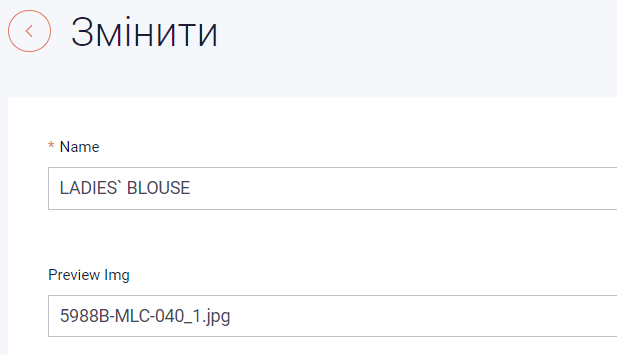


Рис.3.13 Зміна

Приклад виведення інформації про всі збуті товари (Рис.3.14).



Рис.3.14 Статистика

Висновки до розділу

При виконанні даного розділу ми прагнули проаналізувати поведінку користувача та максимально просто розглянути створений раніше інтерфейс, переосмислити його, внести деякі корегуючі зміни та пояснити алгоритм користування.

Також ми намагалися віднайти можливі слабкі сторони нашого програмного продукту, перевірити його на коректну роботу та проаналізувати можливі неправильні дії, які користувач може здійснити.

# РОЗДІЛ 4 АДМІНІСТРУВАННЯ БАЗ ДАНИХ

4.1 Розробка заходів захисту інформації в БД

Для реалізації максимальної безпеки даних користувача було передбачено можливість шифрування паролю перед збереженням його у БД за допомогою можливостей bcrypt (адаптивна криптографічна хеш-функція формування ключа, що використовується для захищеного зберігання паролів). Функція заснована на шифрі Blowfish, час її роботи легко налаштовується і її можна уповільнити, щоб ускладнити атаку перебором.

Також для досягнення максимального рівня цілісності даних та коректних розрахунків для поля ціна товару, ніколи не потрібно використовувати тип даних float, адже це може призвести до некоректних розрахунків при придбанні товарів, а згодом і до збитків компанії. Саме тому буда обрана стратегія роздільного зберігання грошових значень та використання типу даних int.

Передбачимо можливість авторизації користувача (Рис.4.1) та надання всім зареєстрованим персонам певну роль, кожен з яких матиме свій власний рівень доступу до інформації (Таблиця 4.1). Доступні можливості: 1— перегляд даних, 2 - редагування, 3 - видалення, 4 - повний доступ. Позначка \*, вказує на можливість роботи лише з власними записами.

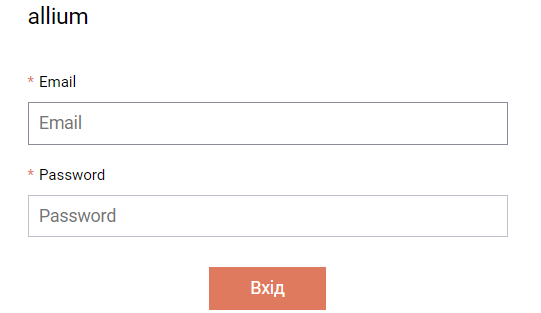


Рис.4.1 Форма авторизації користувача

Таблиця 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Матриця доступу** | | | |
| **Таблиці** | **Адміністратор** | **Розпорядник** | **Користувач** |
| Товари | 4 | 1 | 1 |
| Категорії | 1, 2 | 1, 2 | 1 |
| Відгуки | 4 | 1, 3 | 1, 2, 3\* |
| Категорія-фільтри | 4 | 1, 2 | - |
| Товар-фільтри | 4 | 1, 2, 3 | 1 |
| Склад | 4 | 1, 2 | - |
| Користувачі | 1, 2 | 1\* | 1, 2\* |
| Корзина | 1 | 1 | 1, 2, 3\* |
| Замовлення | 1, 2 | 1, 2 | 1\* |

4.2 Налаштування параметрів роботи SQL-сервера

Open Server - це портативний локальний WAMP / WNMP сервер, який має багатофункціональну керуючу програму і великий вибір компонентів. Представлений пакет програм - це перший повноцінний професійний інструмент, створений спеціально для веб-розробників з урахуванням їх рекомендацій і побажань.

Для налагодження скриптів в різному оточенні Open Server пропонує на вибір відразу два види HTTP серверів, різні версії СУБД модулів, а також можливість швидкого перемикання між ними.

Після встановлення та деяких конфігурацій Open Server, за допобогоб MySQL підключаємо створену нами БД до серверу. Також необхідно створити користувача з логіком та паролем, за допомогою якого ми будемо здійснювати подальше підключення до БД на бекенді нашого веб-додатку.

Висновки до розділу

При виконанні даного розділу курсової роботи було розроблено комплекс заходів для забезпечення безпеки інформації, що зберігається в базах даних, а також їх конфіденційність та цілісність. Для цього було розглянуто різні варіанти шифрування даних, та обрано bcrypt та за допомогою сформованого ключа шифруємо користувацький пароль при його збереженні.

Також було передбачено створення окремої таблиці для збереження ціни товару, де окремо зберігається кількість копійок та гривень у форматі int задля максимального рівня цілісності даних та коректних розрахунків для поля ціна товару.

Більше того, було розроблено матрицю доступу до таблиць створеної БД, де ми визначили повноваження кожної групи користувачів та вказали всі їх дозволені дії.

# ВИСНОВКИ

Під час виконання даної роботи було проаналізовано завдання розробки БД оптимальної для веб додатку магазину одягу та демонстрація її можливостей завдяки створеному власноруч інтерфейсу. Також ми виконали порівняльний аналіз СУБД MySQL, SQLite. Розробили технічне завдання –основні функції, етапи розробки, вимоги до інтерфейсу. Вивчили нові теоретичні відомості, необхідні для виконання поставлених перед нами задач.

Після аналізу було спроектовано БД та розроблена математична модель та алгоритм обробки даних в системі, визначено загальний алгоритм роботи авторизованого користувача та максимально просто розглянути створений раніше інтерфейс, переосмислити його, внести деякі корегуючі зміни та пояснити алгоритм користування. Намагалися віднайти можливі слабкі сторони нашого програмного продукту, перевірити його на коректну роботу та проаналізувати можливі неправильні дії, які користувач може здійснити.

В результаті курсового проекту отримано продукт, який відповідає функціональним вимогам поставленим на початку розробки проекту.

# ВИКОРИСТАНІ ДЖЕРЕЛА

* + - 1. Trott. Node.js Documentation [Електронний ресурс] / Trott – Режим доступу до ресурсу: <https://nodejs.org/docs/latest-v13.x/api/>.
      2. Express Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/ru/guide/routing.html>.
      3. Стив Розенберг. Веб-розробка із застосуванням Node і Express. Повноцінне використання стека JavaScript / Стив Розенберг., 2016.
      4. Марейн Хавербеке. Виразний JavaScript. Сучасне веб-програмування / Марейн Хавербеке. – Пітер, 2019.
      5. Дуглас Крокфорд. JavaScript: сильні сторони / Дуглас Крокфорд..
      6. Derek J. Balling. High Performance MySQL: Optimization, Backups, Replication / Derek J. Balling., 2008.
      7. Manual Sequelize [Електронний ресурс] – Режим доступу до ресурсу: <https://sequelize.org/master/>.
      8. Гай Гаррисон. MySQL Stored Procedure Programming / Гай Гаррисон, Стивен Фойерштайн., 2006.
      9. Jörg Krause. Programming Web Applications with Node, Express and Pug / Jörg Krause., 2016.

# ДОДАТКИ

Додаток А

**Технічне завдання**

**1. Загальне положення**

**1.1. Найменування програмного засобу**

Повне найменування програмної системи: "База даних магазину одягу Allium" (надалі "програма"). Коротка назва програмної системи - "Allium"

**1.2. Призначення розробки та область застосування**

Програмна система "Allium" призначена для обробки та відображення даних про наявні товари магазину для їх подальшого продажу. В програмі передбачено використання механізму пошуку та фільтрації.

**1.3.Мета**

Програмна система "База даних магазину одягу Allium" дозволить підвищити ефективність, повноту та швидкість роботи працівників магазину щодо демонстрації товару та його продажу.

**1.4. Найменування розробника та замовника.**

Розробник даного продукту - студентка групи ІПЗ-19-1 (надалі "розробник"). Замовник програмного продукту - кафедра програмного забезпечення обчислюваної техніки Житомирського державного технологічного університету в межах виконання курсової з дисципліни «Бази даних» (надалі замовник).

**2. Підстава для розробки**

**2.1. Документ на підставі якого ведеться розробка**

Робота ведеться на підставі навчального плану за напрямом 6.050201 «Програмна інженерія» наказу про закріплення тем курсових робіт за студентами.

**3. Вимоги до програми**

**3.1. Вимоги до функціональних характеристик.**

**3.1.1. Загальні вимоги**

Програмна система має забезпечувати:

* можливість дистанційної роботи з робочих;
* надійне збереження даних та можливість їх;
* можливість модернізації системи через зміну потреб;

**3.1.2. Організація вхідних і вихідних даних**

Вхідними даними є дані про товари (назва, фото, опис, характеристики, ціна, належність до категорії). Організація вхідних і вихідних даних повинна відповідати інформаційній структурі виконуваних з операцій, вхідним та вихідним паперовим документами. Введення оперативних даних повинно виконуватися з використанням форм, побудованих на візуальних компонентах.

**3.1.3. Часові характеристики і розмір пам'яті, необхідної для роботи**

Час реакції програми на дії користувача не повинен перевищувати 0,5 с. Час виконання команд меню не більше 1 с. Відображення масивів даних за запитами не більше 3 хвилин. Доступність БД – 90% цілодобово. Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб. Дисковій простір, необхідний для збереження програми і файлів даних не більше 300 Мб.

**3.2. Вимоги до надійності.**

**3.2.1. Вимоги до надійного функціонування**

Програма повинна нормально функціонувати при безперебійній роботі ПК. Доступність БД 90% при одночасному доступі 30 користувачів. При апаратних збоях, відновлення нормальної роботи програми повинне виконуватися після апаратного збою робочої станції – перезавантаження ОС ПК, запуск виконуваного файлу програми.

**3.2.2.Контроль вхідної і вихідної інформації**

Для контролю коректності вхідної інформації: використання механізму авто заповнення та вибору за переліком. Визначені некоректні дії повинні супроводжуватись повідомленнями про помилку і блокуванням операцій оновлення даних. В системі має бути передбачений захист від загального блокування.

**3.2.3. Час відновлення після відмови**

Час відновлення після відмови, не пов'язаною з роботою програми, повинен складатися із: часу перезапуску операційної системи; часу запуску сервера БД (підключення до сервера ) запуску виконуваного файлу, часу повторного введення або зчитування даних.

**3.3. Умови експлуатації і збереження**

Програма використовується у багато користувальницькому середовище. Регламенті операції проводити за графіком:

* створення резервних копій даних: 1 раз на тиждень;
* архівування даних – 1 раз на місяць;
* оновлення системи ідентифікації та аутентифікації користувачів – 1 раз на місяць та за потребою.

**3.4. Вимоги до інформаційної і програмної сумісності.**

**3.4.1. Вимоги до інформаційних структур на вході і виході**

Формат відображення даних має дозволяти імпорт даних в додатки MS Office для редагування та перегляду.

**3.4.2. Вимоги до методів рішення і мов програмування**

Вибір методів рішення здійснюється розробникам без узгодження з замовником. СУБД обирається у відповідності до характеристик визначених в п.3. З замовником погоджується вибір варіанту за вартісною ознакою.

**3.4.3. Вимоги до системи програмних засобів**.

Вимоги до програмного забезпечення сервера:

ОС – серверна версія ОС Windows не нижче Windows 2000 Server, IIS (або інший сервер), СУБД визначається встановлюється та налаштовується розробником (ліцензування СУБД виконується замовником). Вимоги до програмного забезпечення робочої станції: ОС - родина Windows не нижче Windows XP. Драйвери периферійних пристроїв - введення/виводу визначаються та встановлюються при встановленні ОС в залежності від конфігурації робочої станції.

**3.5. Вимоги до складу і параметрів технічних засобів**

Вимоги до складу технічних засобів:

- сервер у базовій конфігурації із підтримкою RAID;

- з’ємний запам’ятовуючий пристрій для архівування даних;

- ПК на базі процесорів Intel, AMD у стандартній комплектації;

- оперативна пам’ять - 128Мбайт;

- об’єм дискової пам’яті - 20Гбайт.

**4. Вимоги до програмної документації**

Керівництво користувача" складається з опису послідовності завантаження програми, основних режимів роботи, опису основних екранних форм, переліку виняткових ситуацій та реакції користувача на них

"Керівництво адміністратора баз даних", складається з опису складу таблиць бази даних та правил доступів до них, опису та послідовності робіт по обслуговуванню бази даних (архівування, резервне копіювання, з вказівкою періодичності виконання та засобів, що для цього використовуються.

Додаток Б

**Створення БД**

CREATE DATABASE allium CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

CREATE TABLE `status` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(20) NOT NULL );

CREATE TABLE `property\_name` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(30) NOT NULL);

CREATE TABLE `property\_value` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`value` VARCHAR(30) NOT NULL);

CREATE TABLE `property\_name\_value` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_name` INT NOT NULL,

`id\_value` INT NOT NULL,

FOREIGN KEY (id\_name) REFERENCES `property\_name` (id),

FOREIGN KEY (id\_value) REFERENCES `property\_value` (id));

CREATE TABLE `price` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`UAH` INT NOT NULL,

`coins` INT NOT NULL);

CREATE TABLE `product` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

`description` TEXT NOT NULL,

`id\_price` INT NOT NULL,

`discount\_percent` DECIMAL,

`detail\_img` VARCHAR(50),

`preview\_img` VARCHAR(50),

FOREIGN KEY (id\_price) REFERENCES `price` (id),

CREATE TABLE `category` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(40) NOT NULL,

`id\_category\_parent` INT,

FOREIGN KEY (id\_category\_parent) REFERENCES `category` (id));

CREATE TABLE `category\_product` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_category` INT NOT NULL,

`id\_product` INT NOT NULL,

FOREIGN KEY (id\_category) REFERENCES `category` (id),

FOREIGN KEY (id\_product) REFERENCES `product` (id));

CREATE TABLE `product\_category\_property\_n\_v` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_product` INT NOT NULL,

`id\_category\_property\_n\_v` INT NOT NULL,

FOREIGN KEY (id\_product) REFERENCES `product` (id),

FOREIGN KEY (id\_category\_property\_n\_v) REFERENCES `category\_property\_n\_v` (id));

CREATE TABLE `storage` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(20) NOT NULL,

`phone` VARCHAR(20) NOT NULL,

`address` VARCHAR(50) NOT NULL);

CREATE TABLE `product\_storage` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_storage` INT NOT NULL,

`id\_product` INT NOT NULL,

`amount` INT NOT NULL,

FOREIGN KEY (id\_storage) REFERENCES `storage` (id),

FOREIGN KEY (id\_product) REFERENCES `product` (id));

CREATE TABLE `user` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`first\_name` VARCHAR(20) NOT NULL,

`last\_name` VARCHAR(20) NOT NULL,

`mail` VARCHAR(20) NOT NULL,

`phone` VARCHAR(20) NOT NULL,

`password\_hash` VARCHAR(20) NOT NULL,

`country` VARCHAR(20) NOT NULL,

`city` VARCHAR(25) NOT NULL,

`address` VARCHAR(50) NOT NULL,

`id\_status` INT,

FOREIGN KEY (id\_status) REFERENCES `status` (id));

CREATE TABLE `group` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(20) NOT NULL);

CREATE TABLE `user\_group` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_user` INT NOT NULL,

`id\_group` INT NOT NULL,

FOREIGN KEY (id\_user) REFERENCES `user` (id),

FOREIGN KEY (id\_group) REFERENCES `group` (id));

CREATE TABLE `backet` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_user` INT NOT NULL,

FOREIGN KEY (id\_user) REFERENCES user (id));

CREATE TABLE `backet\_product` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_backet` INT NOT NULL,

`id\_product` INT NOT NULL,

FOREIGN KEY (id\_backet) REFERENCES `backet` (id),

FOREIGN KEY (id\_product) REFERENCES `product` (id));

CREATE TABLE `order` (

`id` INT PRIMARY KEY AUTO\_INCREMENT,

`id\_backet` INT NOT NULL,

`date` DATE NOT NULL,

`id\_status` INT NOT NULL,

FOREIGN KEY (id\_backet) REFERENCES backet (id),

FOREIGN KEY (id\_status) REFERENCES status (id));

Додаток В

USER MODEL

const Sequelize = require('sequelize');

const bcrypt = require('bcrypt')

module.exports = (sequelize, DataTypes) => {

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

    id: {

      autoIncrement: true,

      type: DataTypes.INTEGER,

      allowNull: false,

      primaryKey: true

    },

    first\_name: {

      type: DataTypes.STRING(20),

      allowNull: false

    },

    last\_name: {

      type: DataTypes.STRING(20),

      allowNull: false

    },

    mail: {

      type: DataTypes.STRING(20),

      allowNull: false,

      unique: "mail"

    },

    phone: {

      type: DataTypes.STRING(20),

      allowNull: false

    },

    password\_hash: {

      type: DataTypes.STRING(60),

      allowNull: false

    },

    country: {

      type: DataTypes.STRING(20),

      allowNull: false

    },

    city: {

      type: DataTypes.STRING(25),

      allowNull: false

    },

    address: {

      type: DataTypes.STRING(50),

      allowNull: false

    }

  }, {

    sequelize,

    tableName: 'user',

    timestamps: false,

    indexes: [

      {

        name: "PRIMARY",

        unique: true,

        using: "BTREE",

        fields: [

          { name: "id" },

        ]

      },

      {

        name: "mail",

        unique: true,

        using: "BTREE",

        fields: [

          { name: "mail" },

        ]

      },

    ]

  });

  User.associate = function(models) {};

  User.prototype.validPassword = function(password) {

    return bcrypt.compareSync(password, this.password\_hash);

};

User.beforeSave((user, options) => {

    const {

      password\_hash

    } = user;

    console.log(password\_hash);

    var saltRounds = 10;

    var salt = bcrypt.genSaltSync(saltRounds);

    var hash = bcrypt.hashSync(password\_hash, salt);

    console.log(hash);

    user.password\_hash = hash;

});

return User;

};