МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КУРСОВА РОБОТА

з Теорія інформаційних систем
(назва дисципліни)

на тему: “Інформаційна система "online shop" для розробки додатку на web сайті”
»

Студента(ки) 3 курсу, групи 6.1260
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

(ініціали та прізвище)

Керівник асистент кафедри програмної інженерії, Шило Г.М
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала:
Кількість балів: Оцінка ECTS:

Члени комісії:
(підпис) (ініціали та прізвище)

(підпис)	(ініціали та прізвище)
(підпис)	(ініціали та прізвище)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет — математичний
Кафедра — програмної інженерії
Дисципліна — Теорія інформаційних систем
Спеціальність — 126 Інформаційні системи та технології
(шифр і назва)
Освітня програма — програмна інженерія

ЗАВДАННЯ НА КУРСОВУ РОБОТУ СТУДЕНТОВІ (СТУДЕНТЦІ)

Назаров Микита Юрійович

1. Тема роботи — “Інформаційна система "online marketplace" для розробки додатку на web сайті”

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи
1. Постановка задачі.
2. Перелік літератури.
3. Практична реалізація коду

4. Зміст роботи(перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Система управління базою даних

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітка
1.	Розробка плану роботи	01.02.2023-07.02.2023	
2.	Збір вихідних даних	07.02.2023-29.02.2023	
3.	Обробка методичних джерел	29.02.2023-13.03.2023	
4.	Розробка першого розділу	13.04.2023 -27.04.2023	
5.	Розробка другого розділу	27.03.2023 -7.04.2023	
6.	Розробка третього розділу	7.04.2023 -14.04.2023	
7.	Оформлення курсової роботи	14.04.2023-18.04.2023	
8.	Захист курсової роботи	18.04.2023-22.04.2023	

Студент _____ _____
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Зміст

РЕФЕРАТ

Курсова робота «Інформаційна система "online shop" для розробки додатку на web сайті»: 41с.,31 мал.,3 джерела

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗИ ДАНИХ, STRAPI, ДІАГРАМИ БІЗНЕС-ПРОЦЕСІВ, ДІАГРАМА КОМПОНЕНТІВ, МОДЕЛЮВАННЯ ДАНИХ, ІНТЕРФЕЙС

Об’єкт дослідження – процес розробки та функціонування онлайн-магазину, його бізнес-модель, вимоги користувачів до додатку, процеси взаємодії між користувачами та системою.

Предмет дослідження – створення та розробка онлайн магазину (online shop) на web-сайті.

Мета роботи – розробка функціонального web-додатку для онлайн-магазину, який дозволить користувачам здійснювати покупки і забезпечить власникові магазину зручний і ефективний інтерфейс для керування замовленнями та іншими аспектами бізнесу

Методи дослідження – методи об’єктно-орієнтованого програмування.

Моделювання даних та бізнес-процесів у методології SADT.

Результати роботи можуть бути використані для подальшого застосування у сфері.

ВСТУП

Сучасний розвиток засобів інформаційних технологій призводить до того, що інформаційні системи стають все більш складними і розширеними, надаючи користувачам все більше можливостей. Прикладом інформаційної системи є онлайн магазин, який забезпечує можливість здійснення покупок за допомогою Інтернету та забезпечує зручність та доступність взаємодії між покупцем та продавцем.
Отже, актуальною задачею є розробка інформаційної системи онлайн магазину з використанням веб-технологій та баз даних.

Метою курсової роботи є розробка інформаційної системи онлайн магазину з можливістю створення та збереженням облікових записів користувачів, перегляду та додавання товарів до кошика, оформлення замовлення, а також адміністрування системи з боку адміністратора.

Задачі, які необхідно розв’язати для досягнення поставленої мети:
- проаналізувати існуючи системи керування базами даних;
- проаналізувати предметну область;
- сформулювати вимоги до інформаційної системи;
- спроектувати схему даних;
- спроектувати макет клієнтського програмного застосунку;
- реалізувати базу даних засобами обраної СКБД;
- реалізувати клієнтський програмний застосунок;
- протестувати інформаційну систему

Об’єктом дослідження є процес розробки інформаційної системи онлайн магазину.
Предметом дослідження є система керування базами даних Strapi.
Методи дослідження, що будуть використані, це методи програмної інженерії, системний аналіз та проектування, а також методологія Agile.

Структура роботи складається з 5 розділів. У першому розділі проводиться аналіз предметної області, описуються основні вимоги до системи. Другий розділ містить опис проектування бази даних та веб-інтерфейсу. Третій розділ містить опис реалізації програмного забезпечення. У четвертому розділі описується тестування та валідація розробленої системи. В п’ятому розділі наводиться висновок

Перша частина

1.Аналіз предметної області

1.1 Архітектура СУБД Strapi

Архітектура СУБД Strapi складається з декількох компонентів, які взаємодіють між собою для забезпечення стабільної та ефективної роботи системи.

Основним компонентом є Node.js, який використовується для побудови серверної частини додатку. Node.js дозволяє виконувати JavaScript на сервері та взаємодіяти з базою даних та іншими компонентами системи.

Для зберігання даних Strapi використовує СУБД PostgreSQL, яка забезпечує надійне зберігання даних та швидкий доступ до них. Також Strapi підтримує різні СУБД, такі як MySQL, SQLite та MongoDB.

Для доступу до даних засобами Strapi використовується ORM (Object-Relational Mapping) Mongoose, яка дозволяє взаємодіяти з базою даних у вигляді об'єктів, що спрощує роботу з даними та зменшує кількість коду.

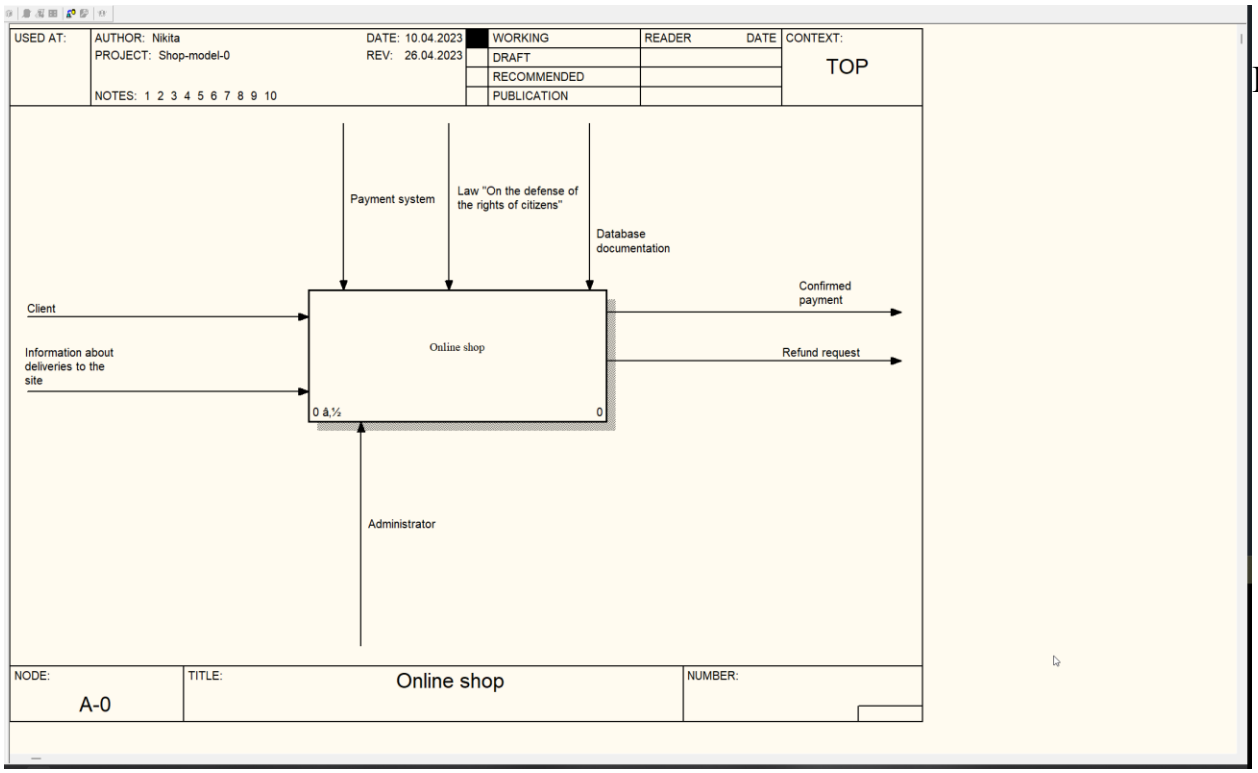
Компоненти Strapi взаємодіють між собою за допомогою HTTP-протоколу, що дозволяє розподіляти завдання та забезпечує стабільну та швидку роботу системи. Strapi також підтримує використання REST API, що дозволяє зручно та ефективно взаємодіяти з додатком через HTTP-запити.

Також важливою складовою архітектури Strapi є система аутентифікації та авторизації, яка дозволяє забезпечити безпеку даних та зменшити ризик несанкціонованого доступу до даних.

У цілому, архітектура СКБД Strapi є добре структурованою та забезпечує ефективну роботу системи, а також має високий рівень безпеки даних.

1.2 Діаграма процесів

Для кращого розуміння процесу роботи інформаційної системи була побудована діаграма бізнес-процесів.



На

Illustration 1: Схема-1

діаграмі зображено процес роботи інтернет-магазину. Діаграма розділювальна 3 основных блока:1 блок-”Користувацький інтерфейс,2 блок-”Інтерфейс адміністратора”,3 блок-”База-даних”.

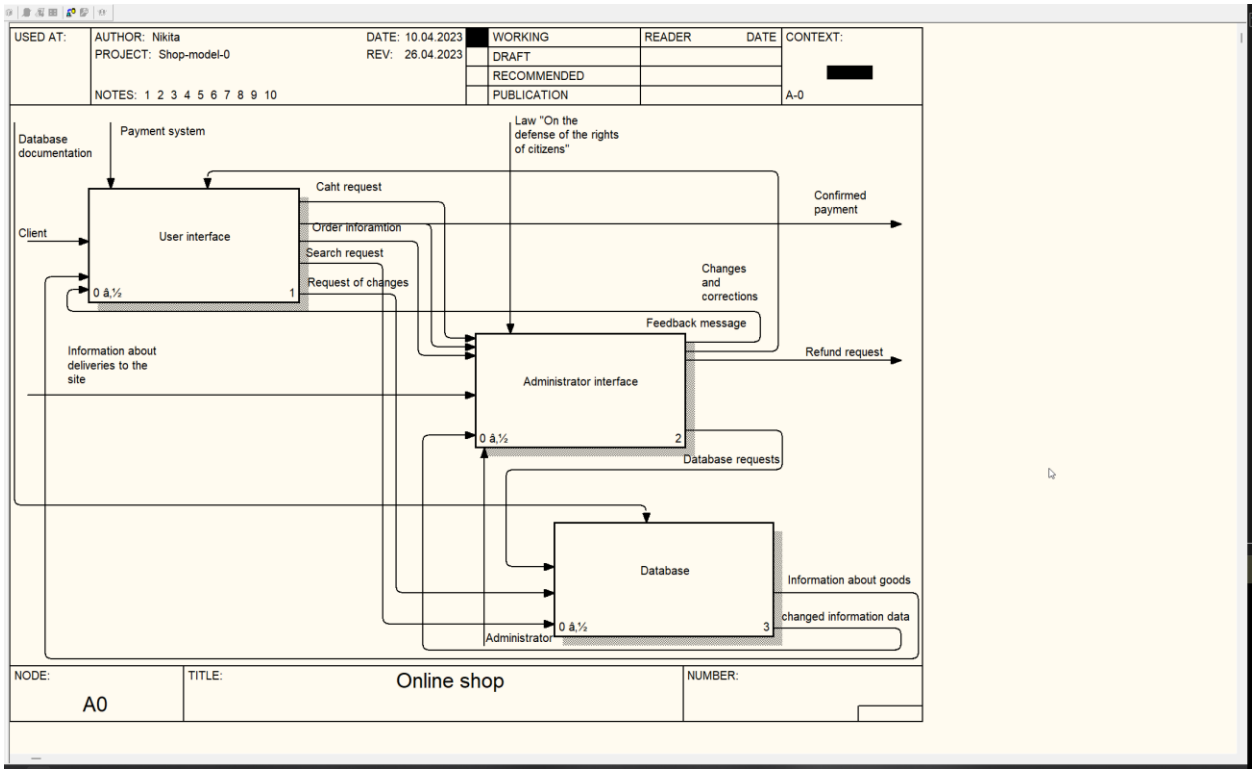
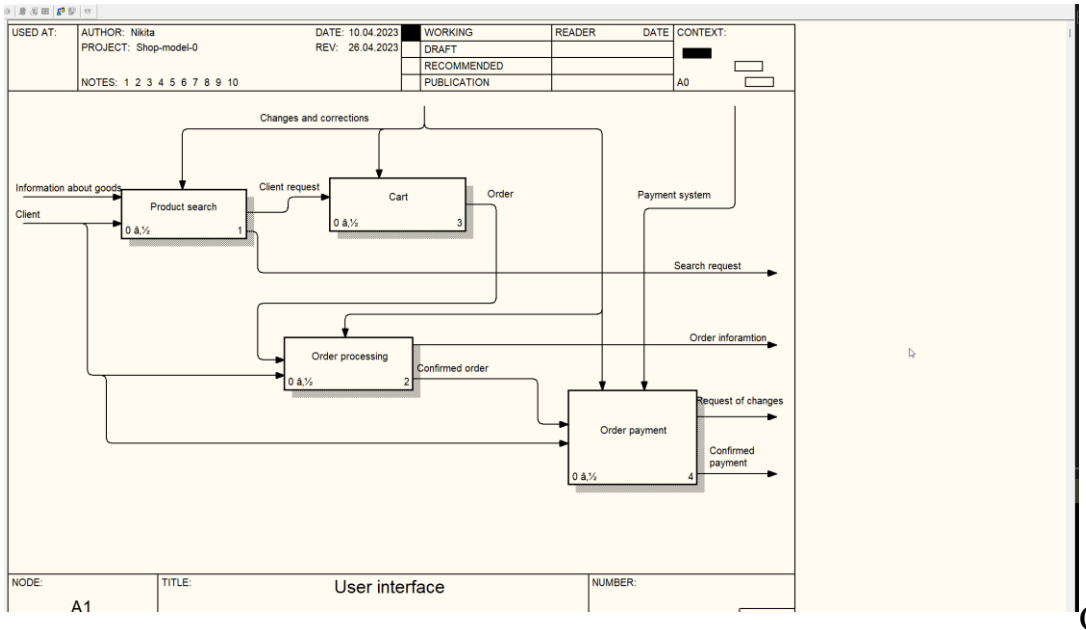


Illustration 2: Схема-2

Роздивимось кожний з блоків більш детально.

Діаграма користувацького інтерфейсу містить процеси:

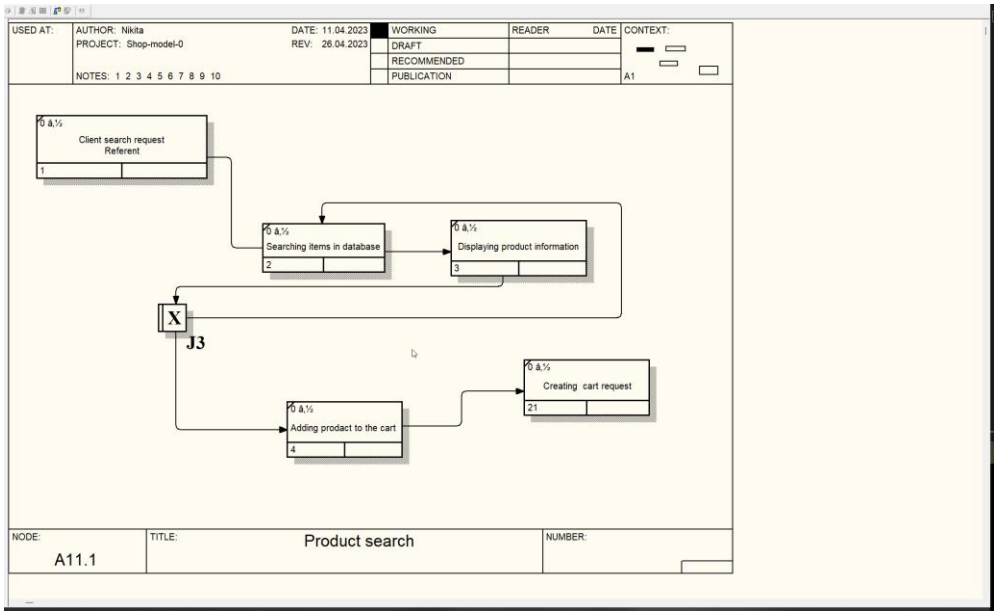
- Пошуку товару
- Кошик
- Створення замовлення
- Форму оплати



Опишемо дані процеси за допомогою методології IDEF3.

1. Пошук товару

Illustration 4: Схема-4



2. Створення замовлення

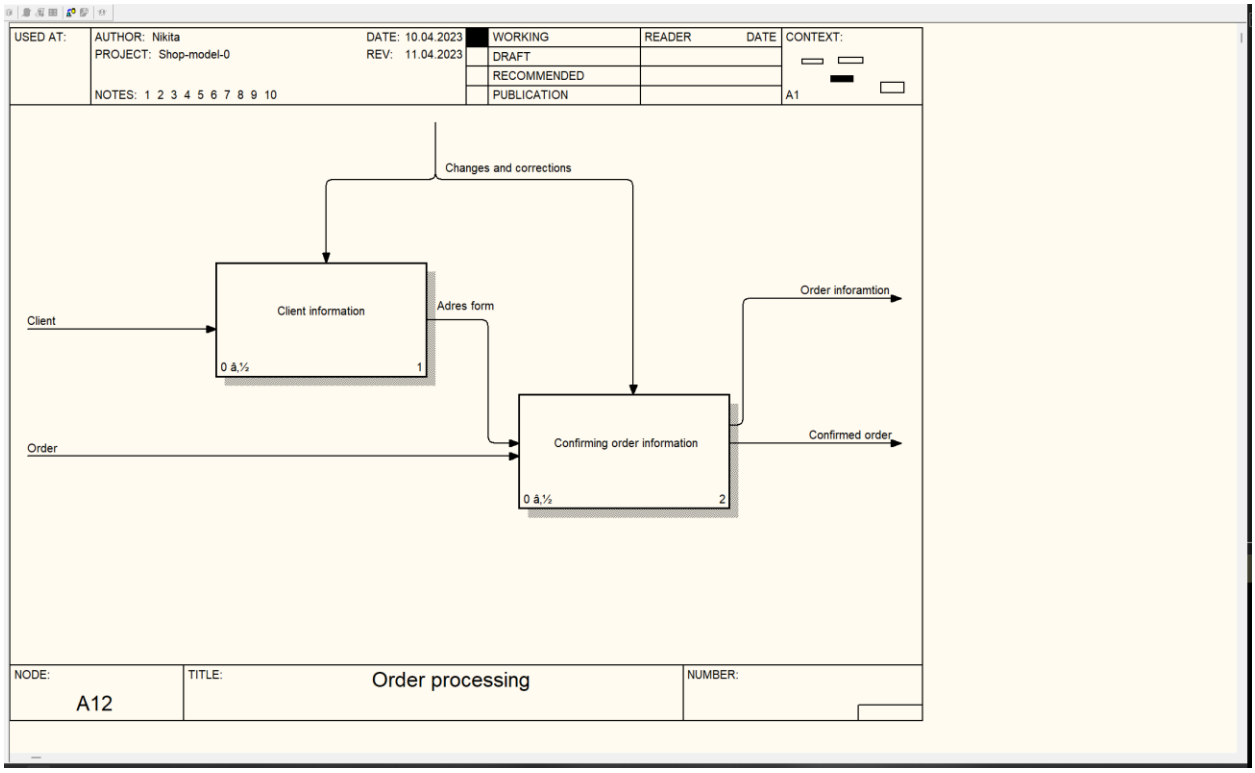


Illustration 5: Схема-5

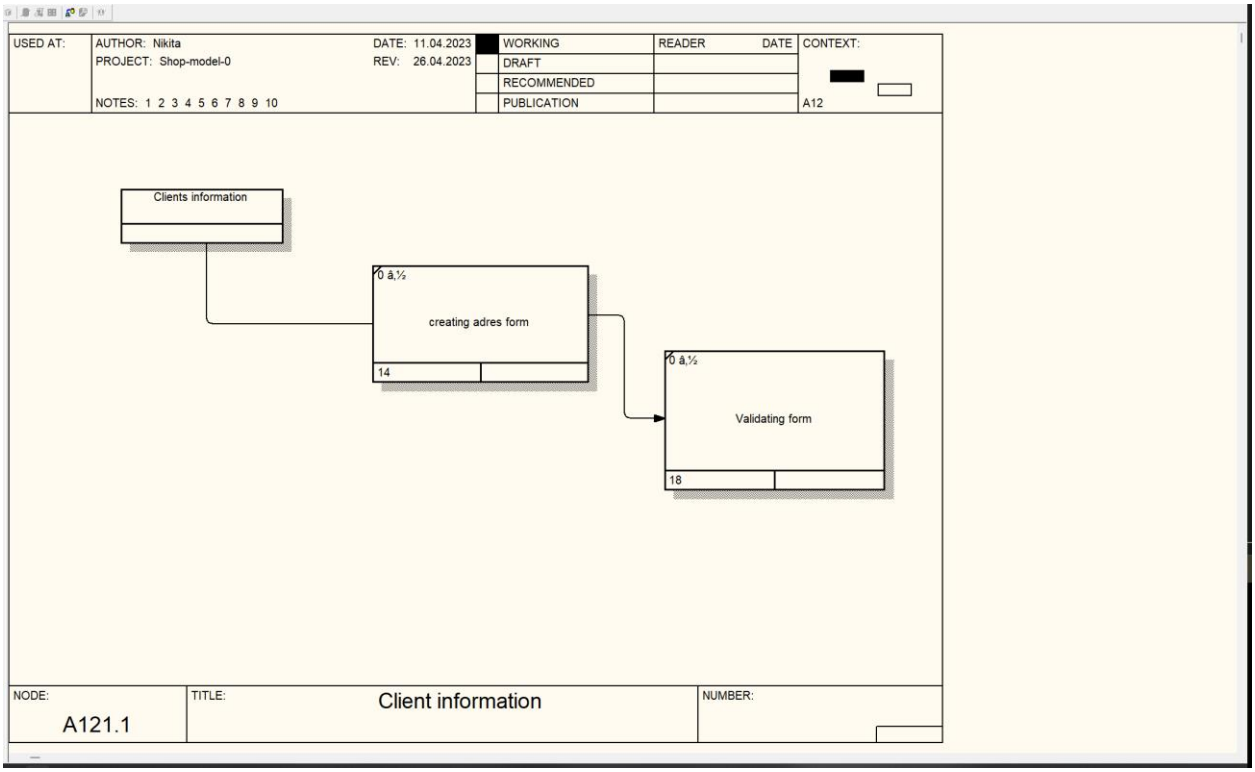


Illustration 6: Схема-6

3. Корзина

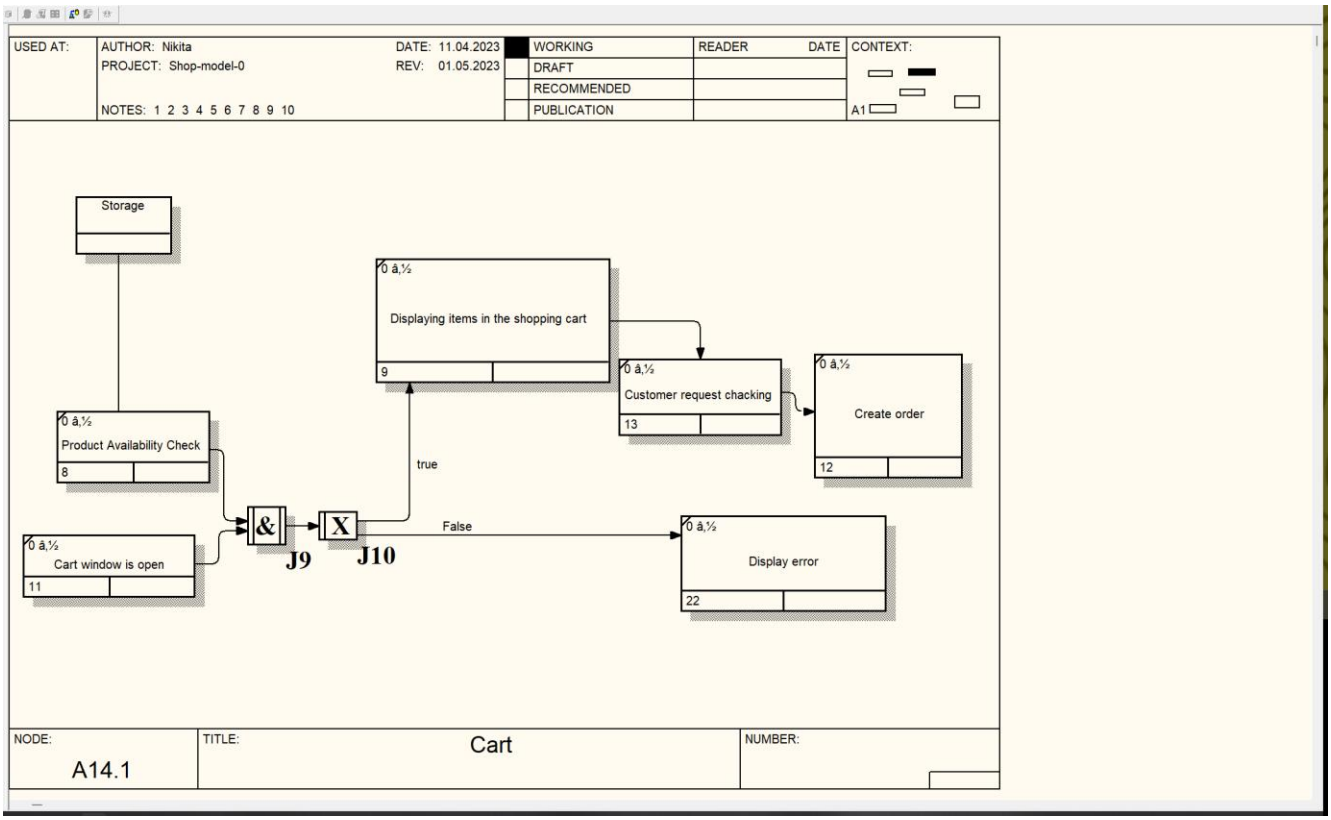


Illustration 7: Схема-7

4. Форма оплата

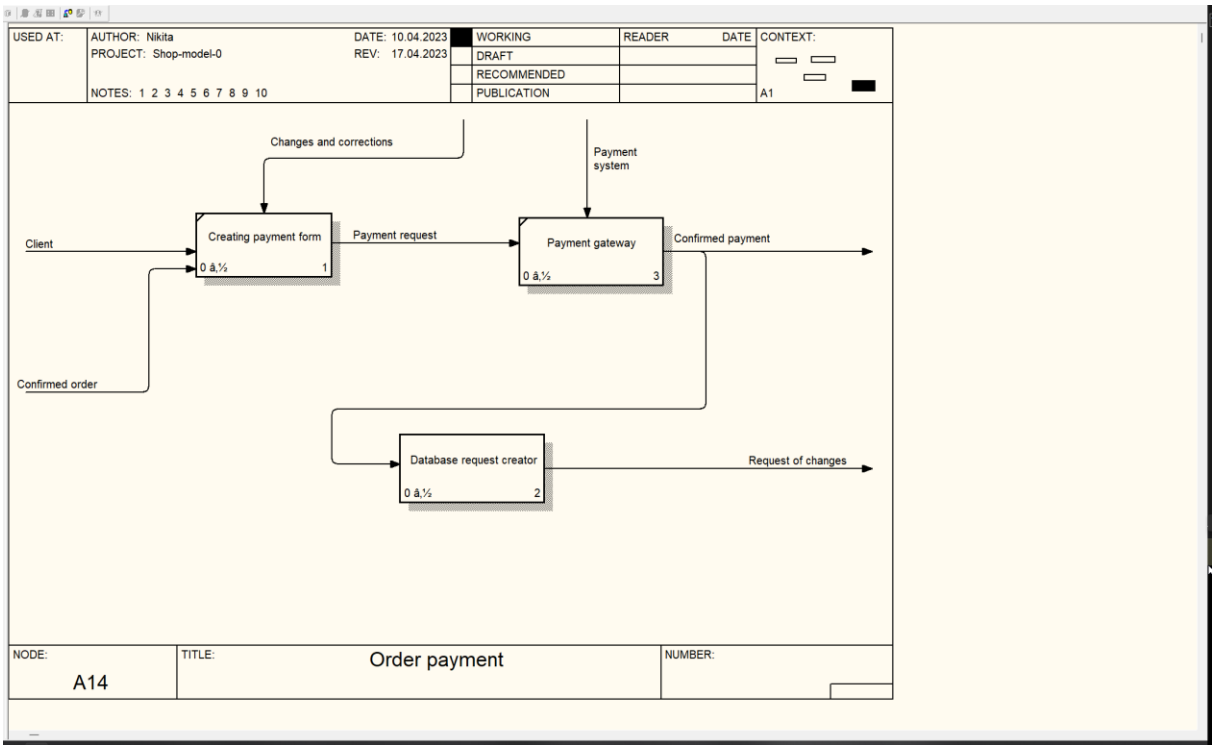


Illustration 8: Схема-8

Діаграма інтерфейсу адміністратора містить процеси:

- Управління замовленнями
- Управління товарам
- Статистика продаж
- Налаштування магазину

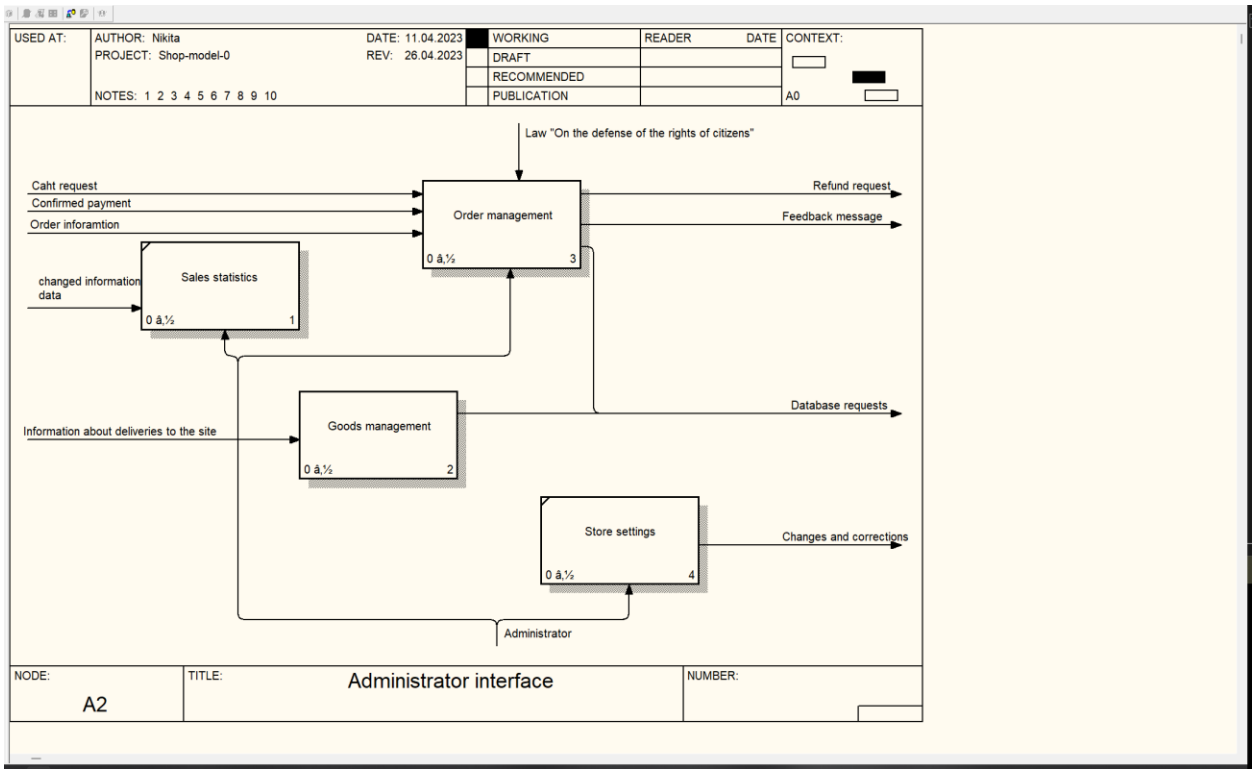


Illustration 9: Схема-9

Опишемо дані процеси більш детально.

1. Управління товарам

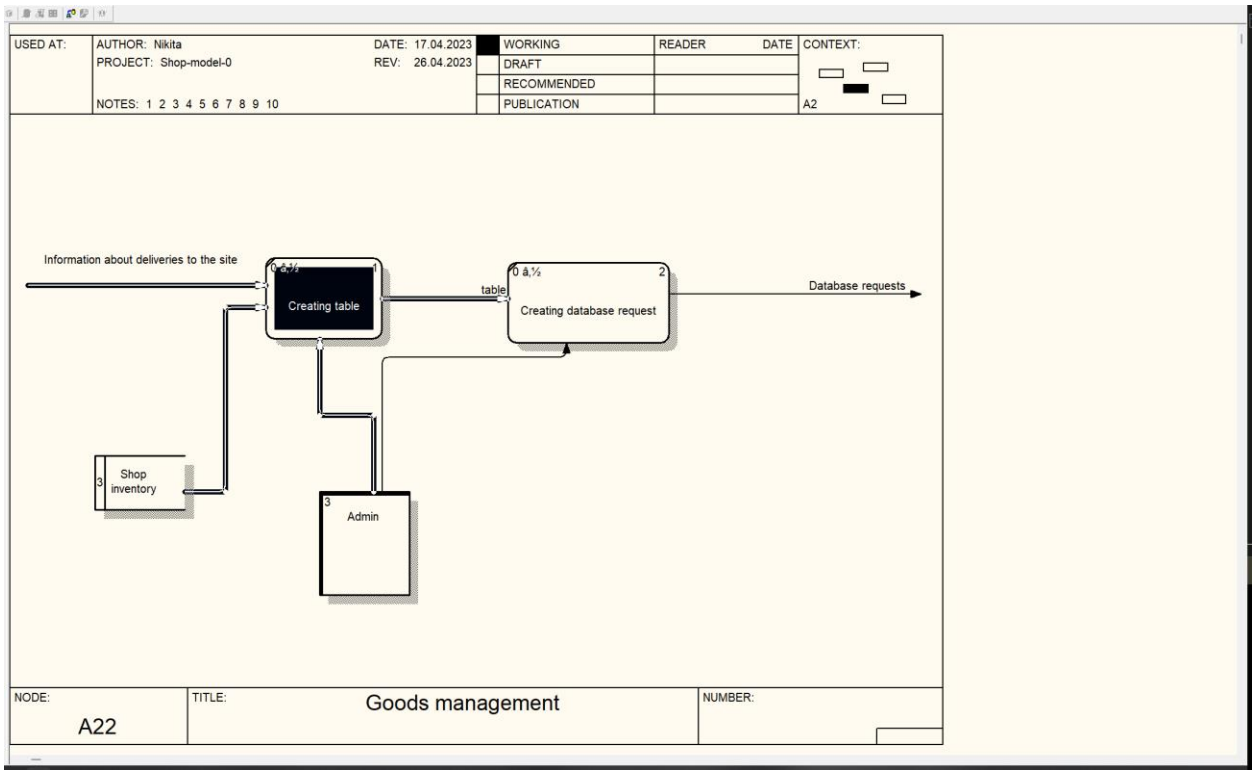


Illustration 10: Схема-10

2. Управління замовленнями

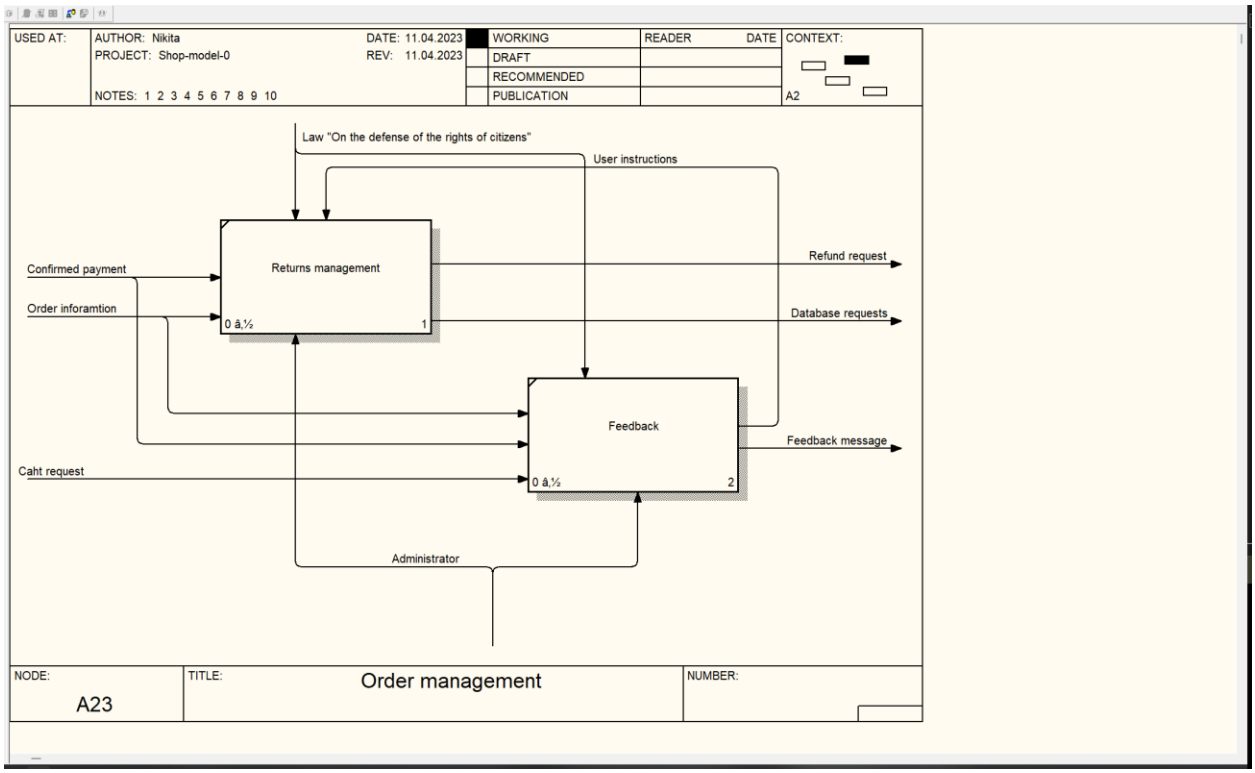


Illustration 11: Схема-11

Діаграма база-даних містить процеси:

- Структура даних
- Зберігання даних
- Індексація

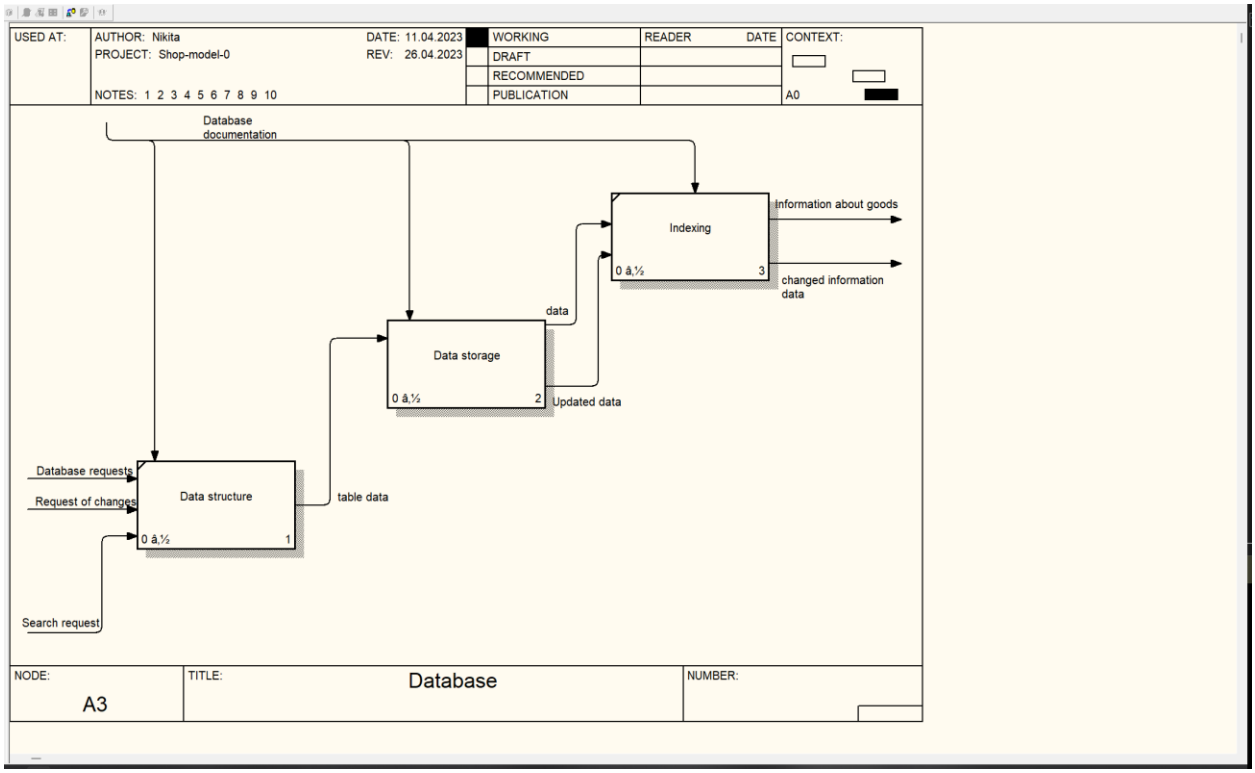


Illustration 12: Схема-12

Опишемо дані процеси більш детально.

1. Зберігання даних

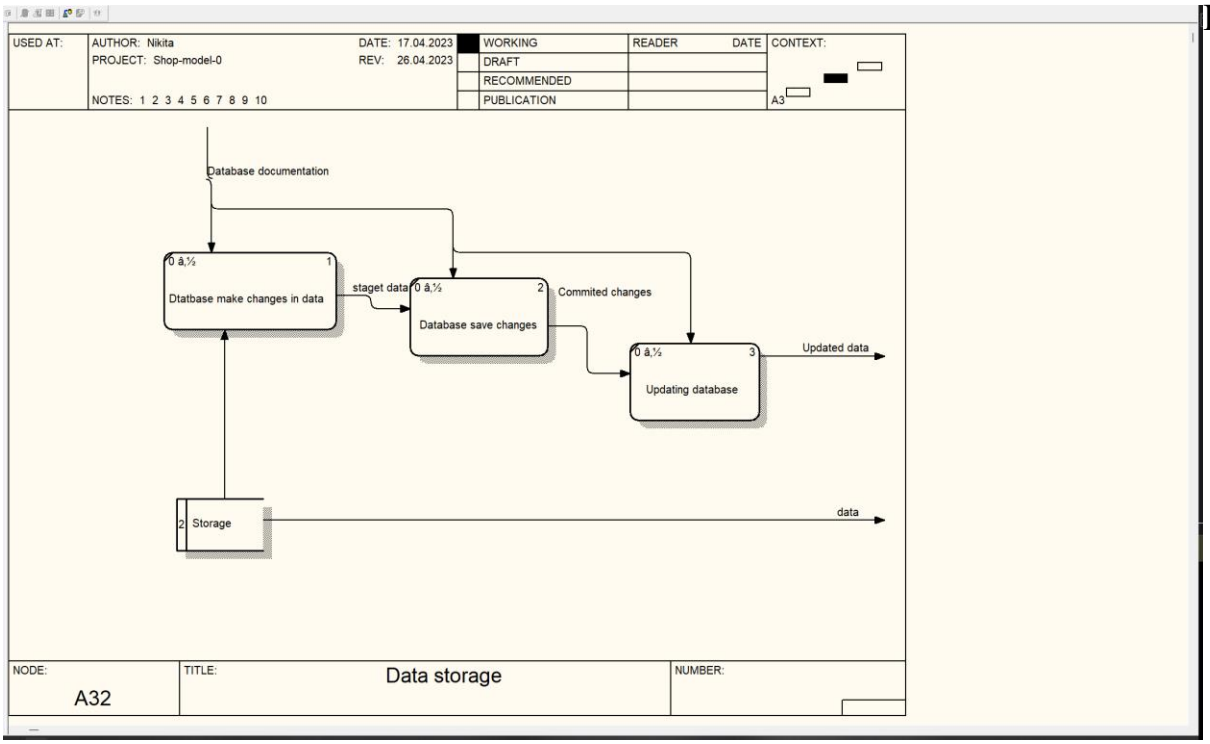


Illustration 13: Схема-13

Ця діаграма бізнес-процесів дозволяє краще зрозуміти, як саме працює інформаційна система, яка була розроблена для цього інтернет-магазину.

2.Проектування архітектури системи

2.1 Проектування архітектури ІС

Після проведення аналізу предметної області та бізнес-процесів, можна розробити архітектуру інформаційної системи.

Для системи керування базами даних Strapi рекомендується використовувати таку архітектуру:

1. Клієнтська частина (frontend)

- Веб-сторінка (HTML, CSS, JavaScript)
- Бібліотеки для роботи з API

2. Серверна частина (backend)

- Node.js
- Бібліотека для роботи з базою даних Strapi SDK

3. База даних

- Система керування базами даних Strapi

Веб-сторінка буде взаємодіяти з сервером за допомогою API, яке буде забезпечене backend-ом системи. Backend буде отримувати запити від клієнтської частини, обробляти їх та здійснювати взаємодію з базою даних. Для зручності розробки та підтримки системи можна використовувати фреймворк Express.js або Nest.js, які дозволяють зменшити час розробки та забезпечують хорошу структуру проекту.

Для зв'язку з базою даних Strapi, яка є NoSQL базою даних, можна використовувати Strapi SDK. База даних Strapi забезпечує можливість використання гнучкої схеми даних та масштабування системи в майбутньому.

Така архітектура дозволяє створити систему, яка буде забезпечувати ефективну роботу з базою даних та забезпечувати можливість розширення та підтримки системи в майбутньому.

2.2 Архітектура системи

Архітектура системи Strapi базується на архітектурі, що використовує мікросервіси. Це означає, що функціональність системи розподілена між невеликими незалежними сервісами, які можуть бути розгорнуті і масштабовані окремо.

Структура мікросервісів Strapi включає наступні компоненти:

1. Клієнтський інтерфейс - це веб-інтерфейс, який дозволяє користувачам керувати своїми даними. Клієнтський інтерфейс розроблений з використанням технологій React та Redux.

2. API-сервіс - це основний компонент системи Strapi, який надає доступ до даних через REST API. API-сервіс побудований на Node.js та використовує фреймворк Express.js.

3. Сервіс аутентифікації - відповідає за автентифікацію та авторизацію користувачів в системі. Сервіс автентифікації базується на бібліотеці Passport.js.

4. Сервіс авторизації - відповідає за визначення рівня доступу до різних функціональних частин системи для користувачів з різними ролями.

5. Сервіс управління контентом - відповідає за зберігання та керування контентом, який використовується в системі.

6. База даних - це реляційна база даних, яка використовується для зберігання даних, що стосуються користувачів та контенту.

Така архітектура дозволяє забезпечити високу масштабованість, доступність та надійність системи Strapi.

Висновок до першої частини

Було розглянуто процес розробки системи керування базами даних Strapi. У результаті аналізу було проведено моделювання бізнес-процесів та створено структуру бази даних, що відповідає вимогам додатку. Було досліджено функціональні та нефункціональні вимоги до системи, проведено аналіз сучасного стану ринку та вибрано оптимальний стек технологій для розробки.
Отже, відповідно до проведеного дослідження, можна зробити висновок, що розробка системи керування базами даних на основі Strapi є актуальною і перспективною задачею в галузі програмної інженерії. Вона дозволяє ефективно керувати даними, забезпечуючи високий рівень безпеки та зручний інтерфейс для користувачів.

Друга частина

3.Розробка інтерфейсу

3.1 Вимоги до інтерфейсу

Для забезпечення зручності використання інформаційної системи необхідно дотримуватись наступних вимог до її інтерфейсу:

1. Простота та зрозумілість інтерфейсу. Інтерфейс повинен бути легким у використанні та зрозумілим для користувачів будь-якого рівня підготовки.

2. Приємний дизайн. Інтерфейс повинен мати приємний дизайн, що дозволить користувачам зосередитись на основній функціональності системи.

3. Інтуїтивно зрозумілі елементи управління. Усі елементи управління повинні бути легко розпізнавальними та доступними для взаємодії з користувачем.

4. Адаптивність. Інтерфейс повинен адаптуватись до різного типу пристроїв та розмірів екранів, щоб користувачі могли зручно працювати з системою на будь-якому пристрої.

5. Доступність. Інтерфейс повинен бути доступним для користувачів з різними фізичними обмеженнями та забезпечувати можливість взаємодії з системою з використанням спеціальних пристроїв.

6. Можливість персоналізації. Користувач повинен мати можливість налаштувати інтерфейс системи відповідно до своїх потреб та вподобань.

7. Надійність та стабільність. Інтерфейс повинен працювати без перебоїв та завантажень навіть за умови великої кількості запитів користувачів.

3.2 Проектування інтерфейсу

Для проектування інтерфейсу використовувалися засоби HTML, CSS та JavaScript. Було розроблено прототип інтерфейсу, який був подальшим розвитком проекту.

Основною метою інтерфейсу є забезпечення зручного та ефективного користування інформаційною системою. З урахуванням цієї мети, було виконано наступні кроки проектування інтерфейсу:

1. Розроблення загальної концепції дизайну інтерфейсу з використанням засобів HTML та CSS. Дизайн інтерфейсу мав бути простим та зрозумілим для користувачів.

2. Розроблення функціональної моделі інтерфейсу, яка включала в себе всі необхідні функції та можливості системи.

3. Розроблення скриптів на JavaScript для реалізації функціональної моделі та додаткових можливостей інтерфейсу.

4. Перевірка коректності та зручності інтерфейсу шляхом тестування та виправлення виявлених помилок та недоліків.

3.3 Демонстрація роботи інтерфейсу.

1. Інтерфейс Strapi SDK
   - Content manager

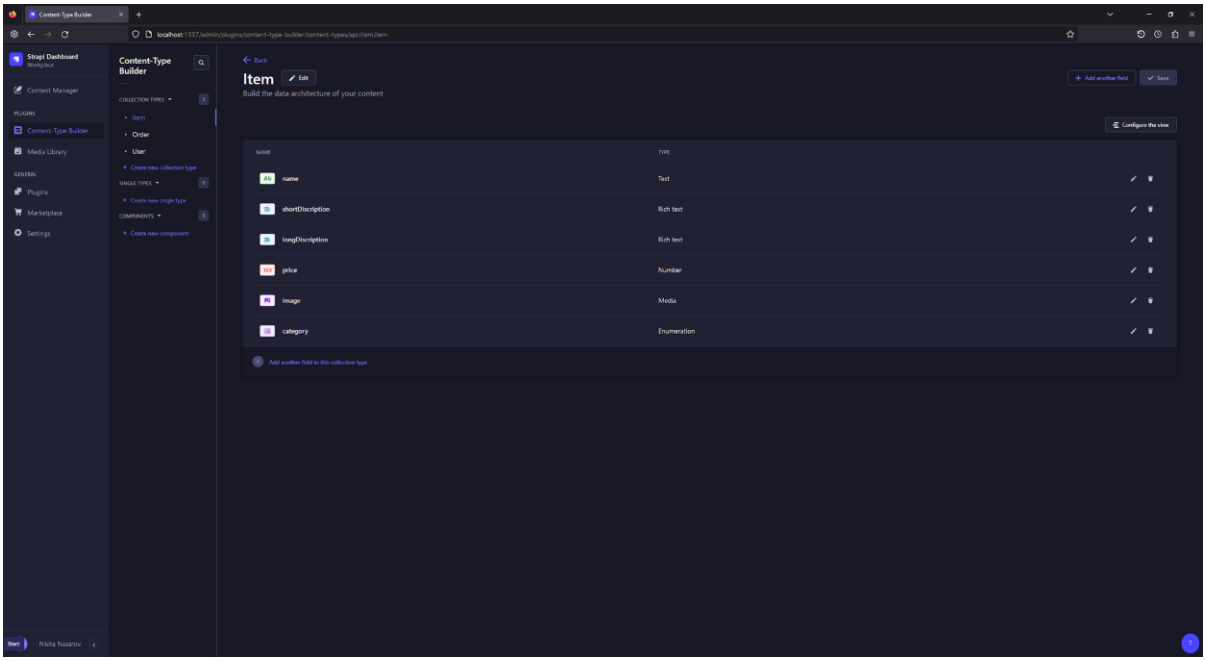


Illustration 14: Інтерфейс створення об'єктів

   - Content-type Builder

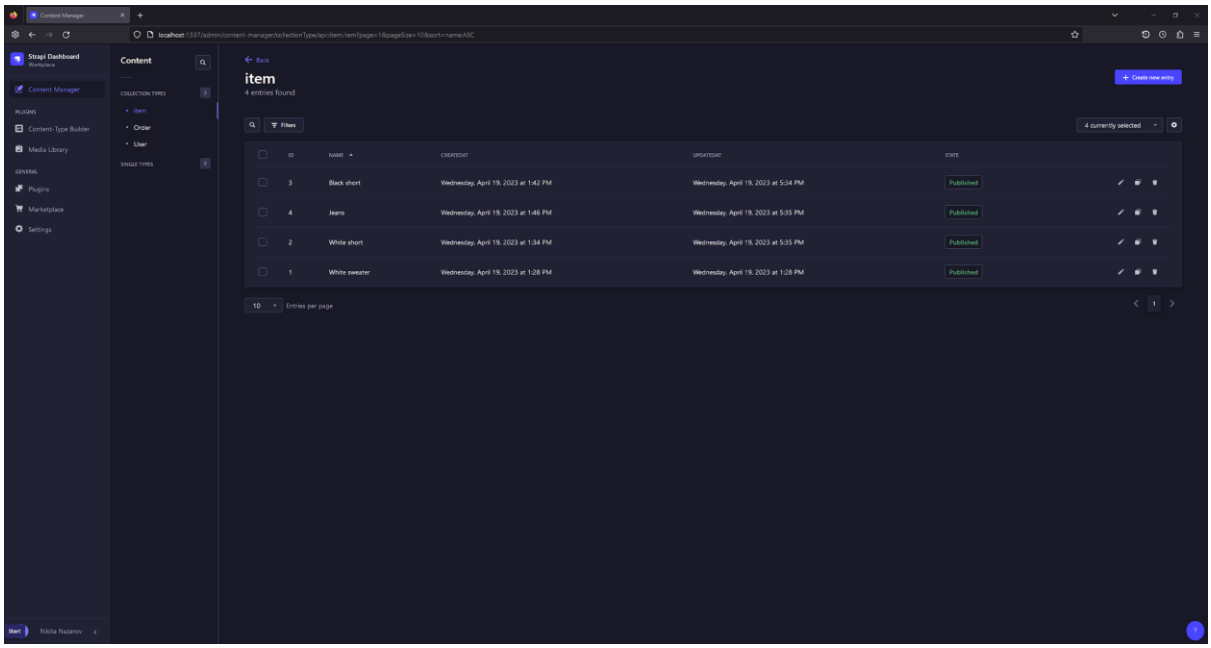


Illustration 15: Інтерфейс керування об'єктів

- Media Library

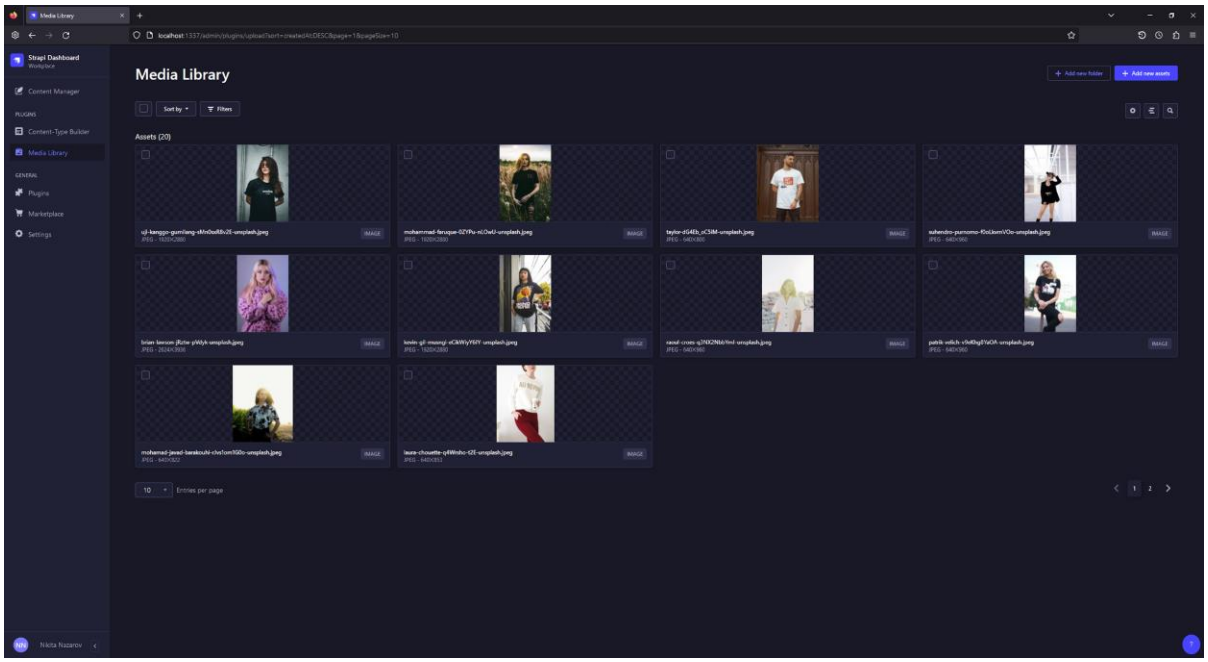


Illustration 16: Інтерфейс додавання медія контенту

- Payment dashboard

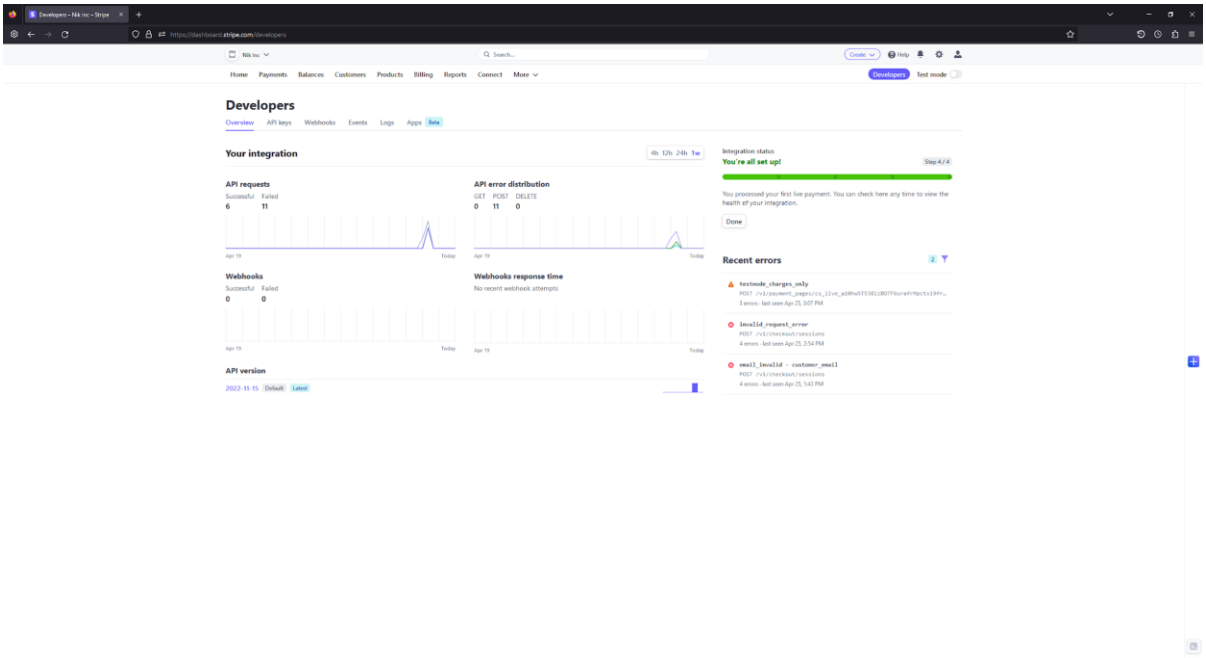


Illustration 17: Інтерфейс оплати

4.РОЗРОБКА ОСНОВНИХ АЛГОРИТМІВ

4.1 МОДЕЛЮВАННЯ ДАНИХ

- Користувач (User) - сутність, що відображає дані про користувача системи. Містить такі поля як ім'я, прізвище, email, пароль та роль користувача.

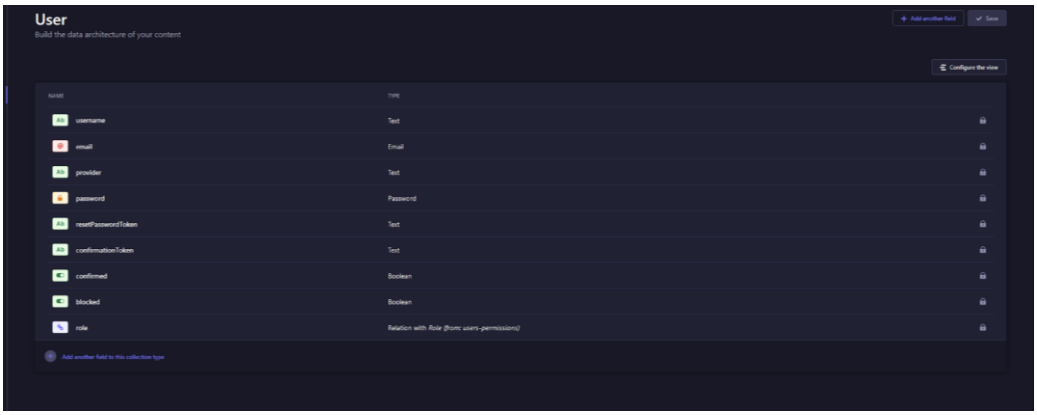


Illustration 18: Об'єкт користувач

```
const initialValues = {
  billingAddress: {
    firstName: "",
    lastName: "",
    country: "",
    street1: "",
```

```
    street2: "",
    city: "",
    state: "",
    zipCode: "",
  },
```

- Товар (Product) - сутність, що відображає дані про товар. Містить такі поля як назва товару, опис, ціна, категорія товару, кількість на складі та зображення товару.

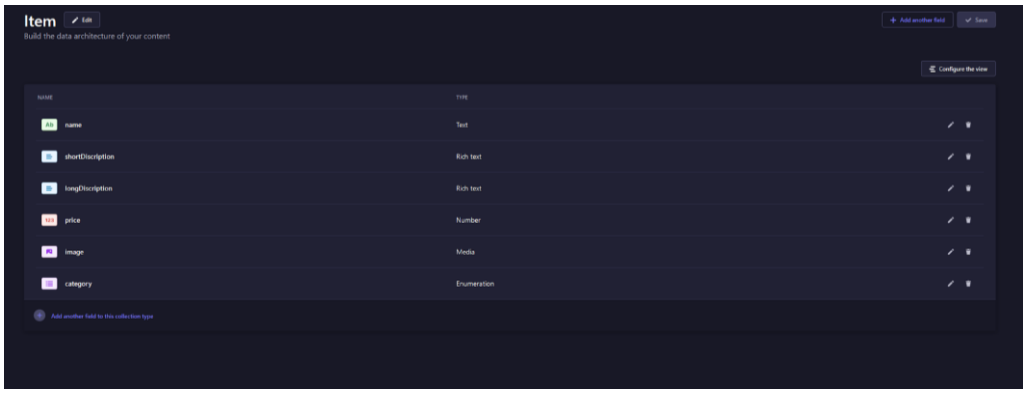


Illustration 19: Об'єкт товар

```
async function getItem() {
    const item = await fetch(
      `http://localhost:1337/api/items/${itemId}?populate=image`,
      {
        method: "GET",
      }
    );
    const itemJson = await item.json();
    setItem(itemJson.data);
  }
```

- Замовлення (Order) - сутність, що відображає дані про замовлення. Містить такі поля як ідентифікатор користувача, дата замовлення, список товарів, сума замовлення та статус замовлення.

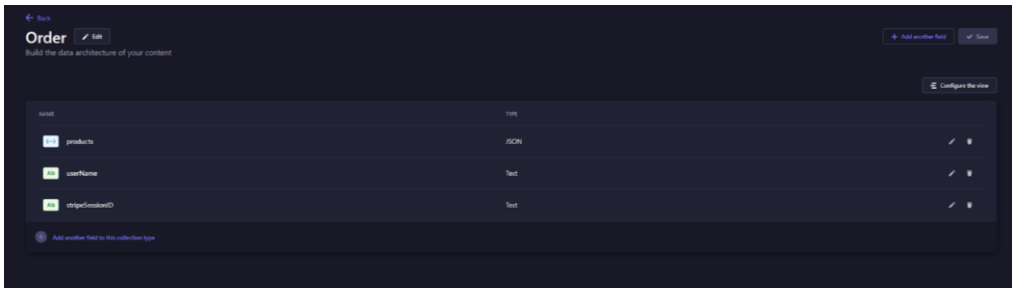


Illustration 20: Об'єкт замовлення

async function makePayment(values) {
  const stripe = await stripePromise;
  const requestBody = {
    userName: [values.firstName, values.lastName].join(" "),
    email: values.email,
    products: cart.map(({ id, count }) => ({
      id,
      count,
    })) };

  const response = await fetch("http://localhost:1337/api/orders", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(requestBody),
  });
  const session = await response.json();
  await stripe.redirectToCheckout({
    sessionId: session.id,
  });
}

4.2 РОЗРОБКА ФУНКЦІОНАЛЬНОСТІ

- Функція додавання товару до кошика

```
const CartMenu = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const cart = useSelector((state) => state.cart.cart);
  const isCartOpen = useSelector((state) => state.cart.isCartOpen);

  const totalPrice = cart.reduce((total, item) => {
    return total + item.count * item.attributes.price;
```

```
}, 0);

return (
  <Box
    display={isCartOpen ? "block" : "none"}
    backgroundColor="rgba(0, 0, 0, 0.4)"
    position="fixed"
    zIndex={10}
    width="100%"
    height="100%"
    left="0"
    top="0"
    overflow="auto"
  >
    <Box
      position="fixed"
      right="0"
      bottom="0"
      width="max(400px, 30%)"
      height="100%"
      backgroundColor="white"
    >
      <Box padding="30px" overflow="auto" height="100%">
        {/* HEADER */}
        <FlexBox mb="15px">
          <Typography variant="h3">SHOPPING BAG ({cart.length})</Typography>
          <IconButton onClick={() => dispatch(setIsCartOpen({}))}>
            <CloseIcon />
          </IconButton>
        </FlexBox>

        {/* CART LIST */}
        <Box>
          {cart.map((item) => (
            <Box key={`${item.attributes.name}-${item.id}`}>
              <FlexBox p="15px 0">
                <Box flex="1 1 40%">
                  <img
                    alt={item?.name}
```

```
                width="123px"
                height="164px"
                src={`http://localhost:1337${item?.attributes?.image?.data?.attributes?.for
mats?.medium?.url}`}
               />
              </Box>
              <Box flex="1 1 60%">
               <FlexBox mb="5px">
                <Typography fontWeight="bold">
                 {item.attributes.name}
                </Typography>
                <IconButton
                 onClick={() =>
                  dispatch(removeFromCart({ id: item.id }))
                 }
                >
                 <CloseIcon />
                </IconButton>
               </FlexBox>
               <Typography>{item.attributes.shortDescription}</Typography>
               <FlexBox m="15px 0">
                <Box
                 display="flex"
                 alignItems="center"
                 border={`1.5px solid ${shades.neutral[500]}`}
                >
                 <IconButton
                  onClick={() =>
                   dispatch(decreaseCount({ id: item.id }))
                  }
                 >
                  <RemoveIcon />
                 </IconButton>
                 <Typography>{item.count}</Typography>
                 <IconButton
                  onClick={() =>
                   dispatch(increaseCount({ id: item.id }))
                  }
                 >
```

```
                <AddIcon />
              </IconButton>
            </Box>
            <Typography fontWeight="bold">
              ${item.attributes.price}
            </Typography>
          </FlexBox>
        </Box>
      </FlexBox>
      <Divider />
    </Box>
  ))}
</Box>

{/* ACTIONS */}
<Box m="20px 0">
  <FlexBox m="20px 0">
    <Typography fontWeight="bold">SUBTOTAL</Typography>
    <Typography fontWeight="bold">${totalPrice}</Typography>
  </FlexBox>
  <Button
    sx={{
      backgroundColor: shades.primary[400],
      color: "white",
      borderRadius: 0,
      minWidth: "100%",
      padding: "20px 40px",
      m: "20px 0",
    }}
    onClick={() => {
      navigate("/checkout");
      dispatch(setIsCartOpen({}));
    }}
  >
    CHECKOUT
  </Button>
</Box>
</Box>
</Box>
```

```
    </Box>
  );
};
```

- Функція пошуку

```
const ShoppingList = () => {
  const dispatch = useDispatch();
  const [value, setValue] = useState("all");
  const items = useSelector((state) => state.cart.items);
  const breakPoint = useMediaQuery("(min-width:600px)");

  const handleChange = (event, newValue) => {
    setValue(newValue);
  };

  async function getItems() {
    const items = await fetch(
      "http://localhost:1337/api/items?populate=image",
      { method: "GET" }
    );
    const itemsJson = await items.json();
    dispatch(setItems(itemsJson.data));
  }

  useEffect(() => {
    getItems();
  }, []); // eslint-disable-line react-hooks/exhaustive-deps

  const topRatedItems = items.filter(
    (item) => item.attributes.category === "topRated"
  );
  const newArrivalsItems = items.filter(
    (item) => item.attributes.category === "newArrivals"
  );
  const bestSellersItems = items.filter(
    (item) => item.attributes.category === "bestSeller"
  );

  return (
    <Box width="80%" margin="80px auto">
```

```
<Typography variant="h3" textAlign="center">
  Our Featured <b>Products</b>
</Typography>
<Tabs
  textColor="primary"
  indicatorColor="primary"
  value={value}
  onChange={handleChange}
  centered
  TabIndicatorProps={{ sx: { display: breakPoint ? "block" : "none" } }}
  sx={{
    m: "25px",
    "& .MuiTabs-flexContainer": {
      flexWrap: "wrap",
    },
  }}
>
  <Tab label="ALL" value="all" />
  <Tab label="NEW ARRIVALS" value="newArrivals" />
  <Tab label="BEST SELLERS" value="bestSellers" />
  <Tab label="TOP RATED" value="topRated" />
</Tabs>
<Box
  margin="0 auto"
  display="grid"
  gridTemplateColumns="repeat(auto-fill, 300px)"
  justifyContent="space-around"
  rowGap="25px"
  columnGap="1.33%"
>
  {value === "all" &&
    items.map((item) => (
      <Item item={item} key={`${item.name}-${item.id}`} />
    ))}
  {value === "newArrivals" &&
    newArrivalsItems.map((item) => (
      <Item item={item} key={`${item.name}-${item.id}`} />
    ))}
  {value === "bestSellers" &&
```

```
        bestSellersItems.map((item) => (
          <Item item={item} key={`${item.name}-${item.id}`} />
        ))}
      {value === "topRated" &&
        topRatedItems.map((item) => (
          <Item item={item} key={`${item.name}-${item.id}`} />
        ))}
    </Box>
  </Box>
);};
```

5.ТЕСТУВАННЯ ТА ВАЛІДАЦІЯ СИСТЕМИ

1. Запуск серверу з базою даних

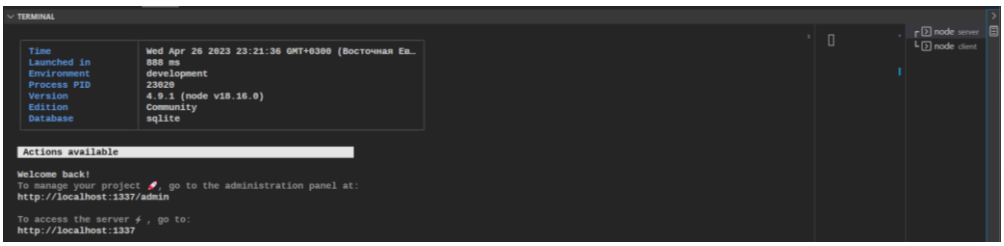
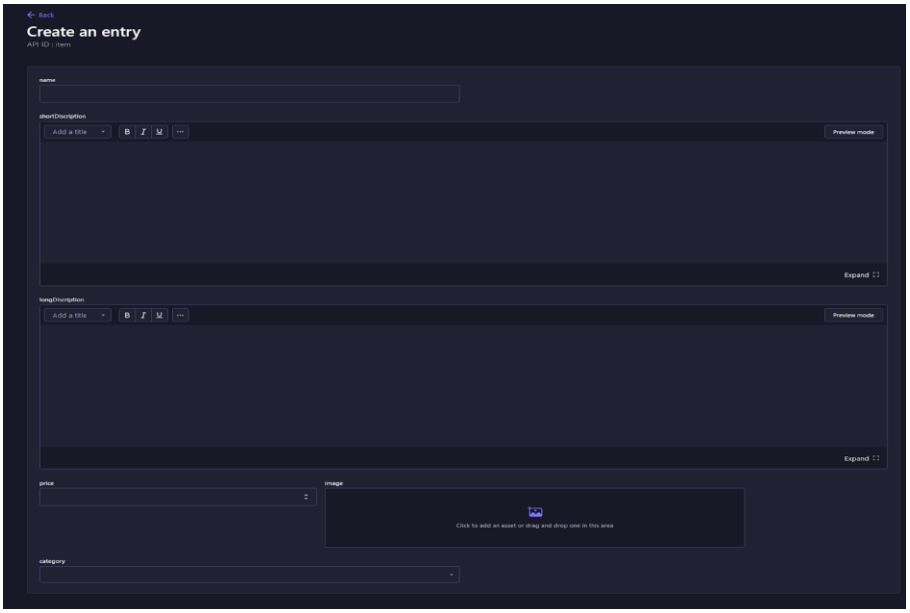


Illustration 21

2.Додавання продукту до бази даних



3. Запуск клієнтської сторінки

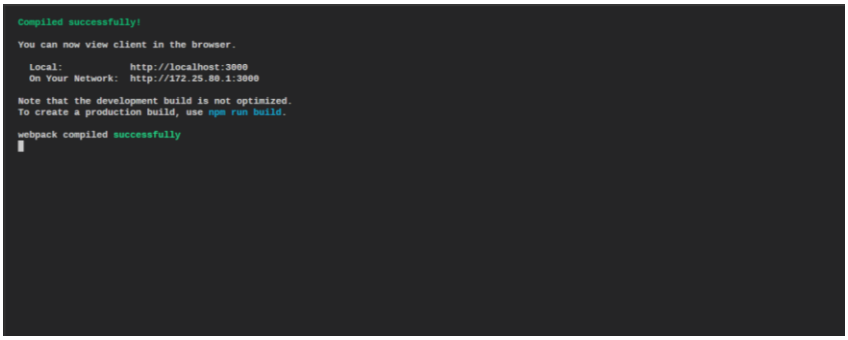


Illustration 23

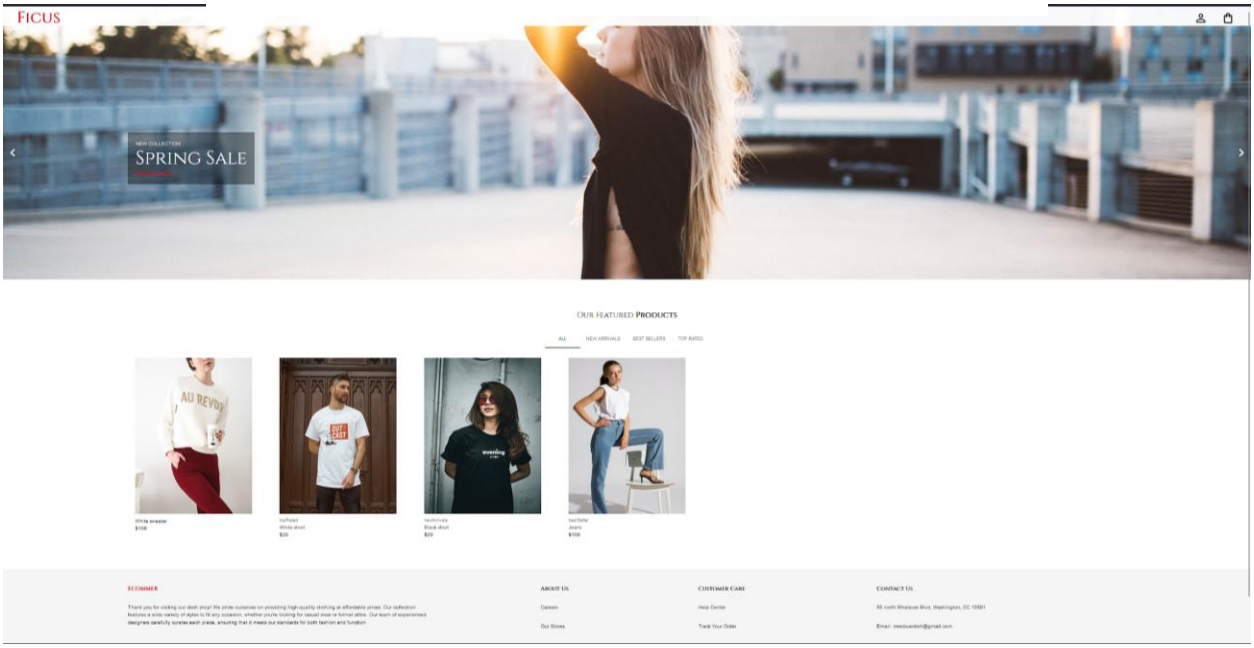


Illustration 24: Головна сторінка сайту

4. Додавання товару до кошика

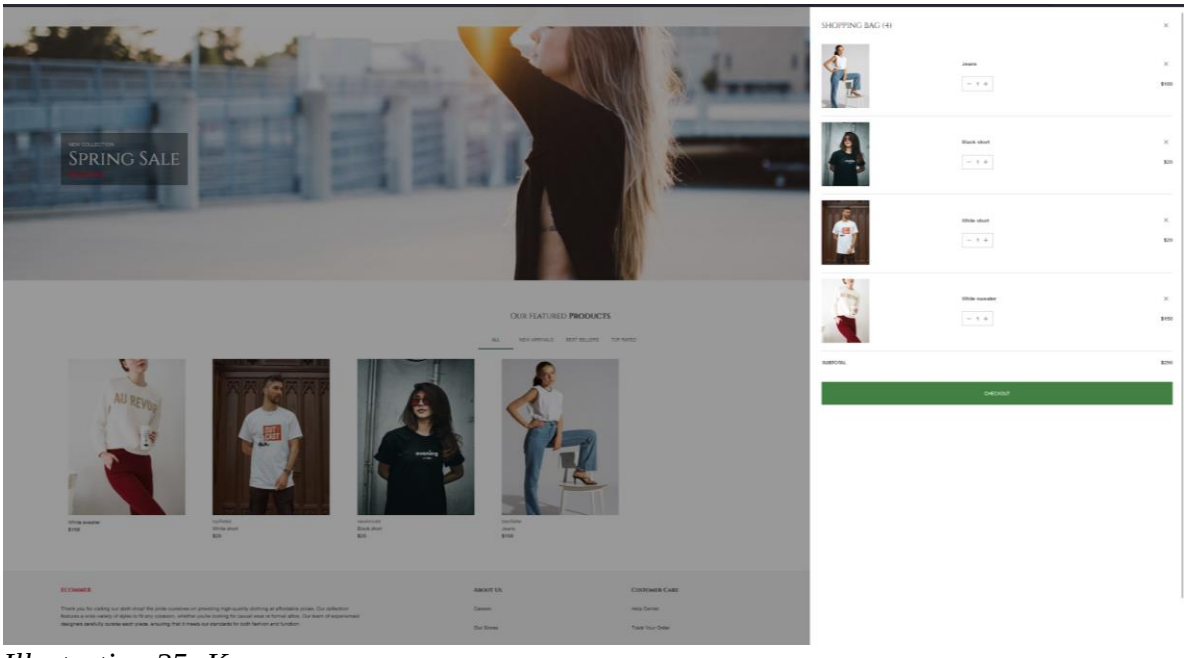


Illustration 25: Кошик

5. Перевірка категорій та детальної інформації

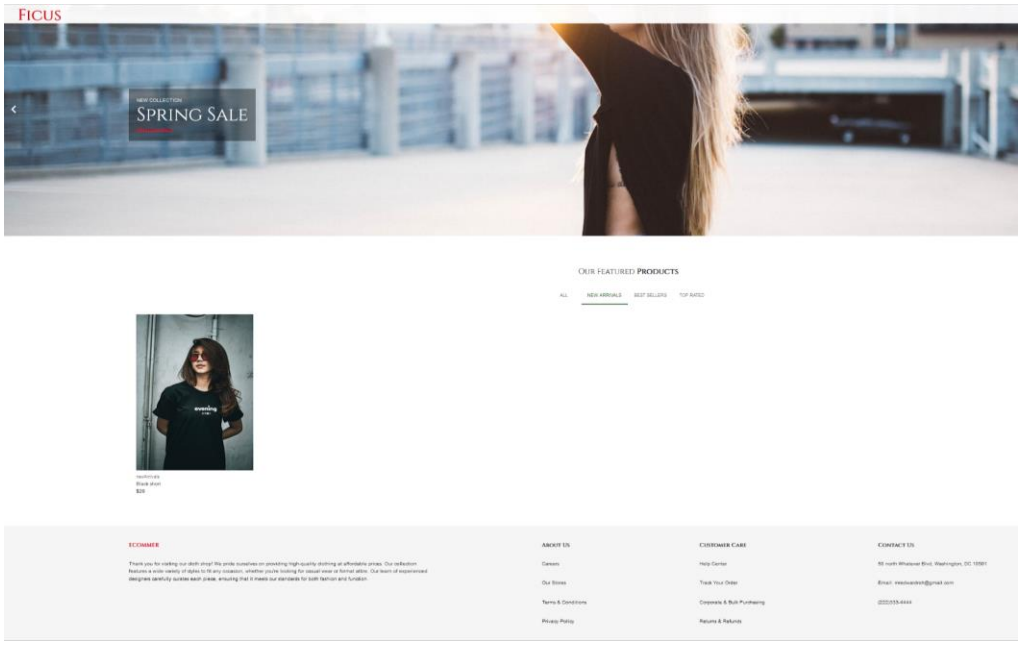


Illustration 26: Категорії сайту

6. Перевірка створення замовлення та оплати
   - Платіжна інформація

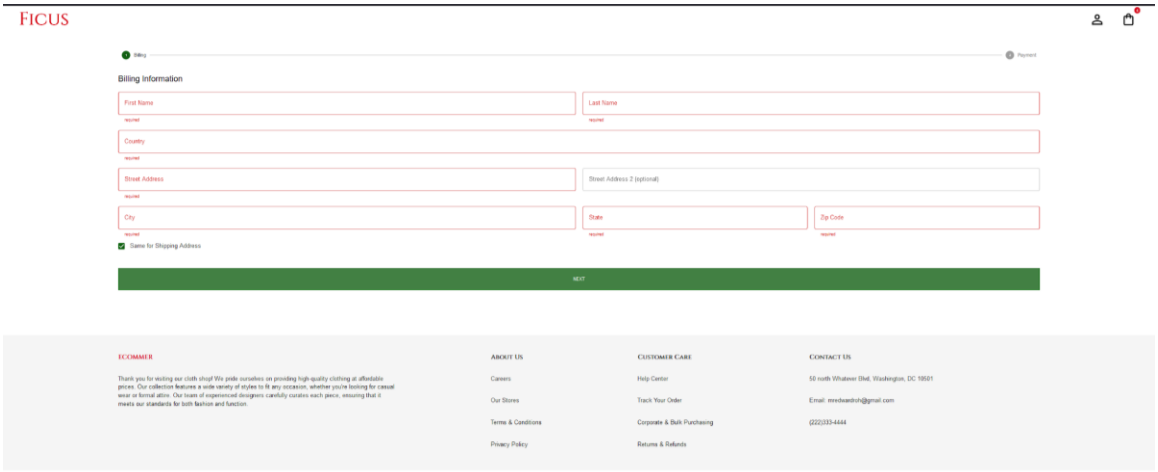


Illustration 27: Форма доставка

   - Контактна інформація

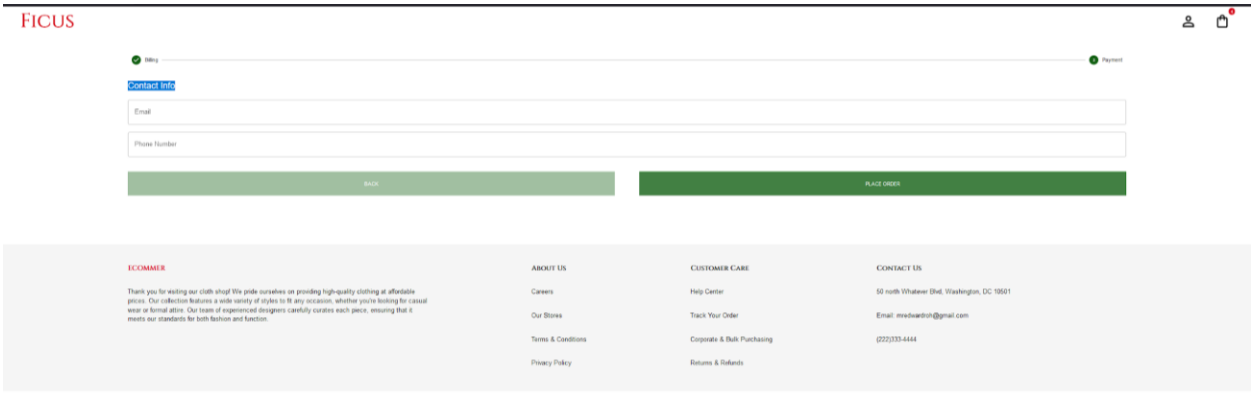


Illustration 28: Форма контактної інформації

- Платіжний запит

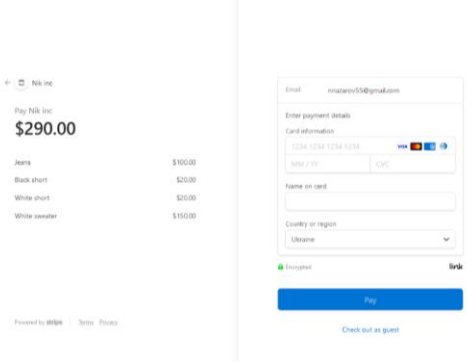


Illustration 29: Форма сплати

- Підтвердження замовлення

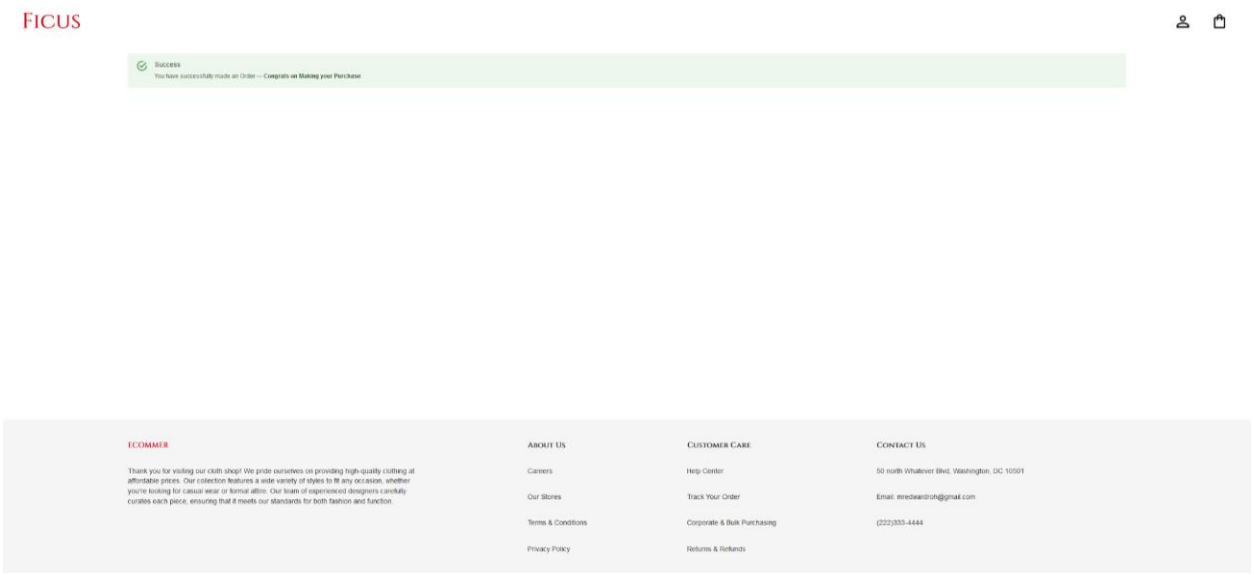


Illustration 30: Підтверджене замовлення

- Перевірка створеного замовлення у базі даних

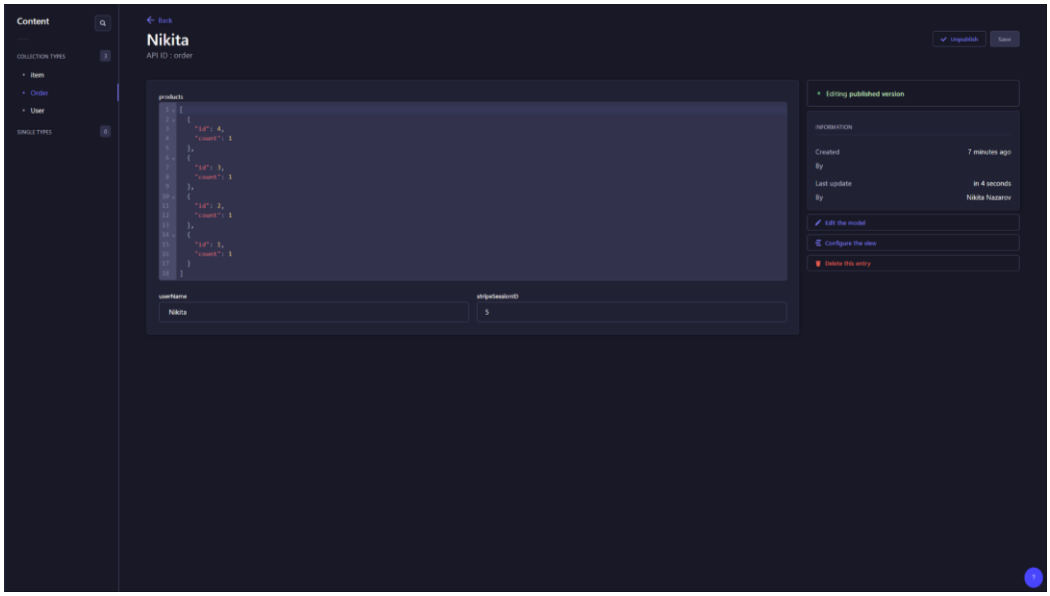


Illustration 31: Приклад створеного об'єкту замовлення

Висновок до другої частини

У розділах 3, 4 та 5 було проведено детальний аналіз вимог до інтерфейсу та моделювання даних. Були розроблені вимоги до інтерфейсу, створена діаграма компонентів, проведено проектування інтерфейсу та демонстрація його роботи. Для моделювання даних було використано сучасні методи та засоби, зокрема ER-діаграми, які допомогли зробити проект ефективнішим та більш структурованим.

Результатом роботи стало створення повноцінної інформаційної системи з високоякісним інтерфейсом та оптимізованою моделлю даних, що дозволить користувачам зручно та швидко працювати з системою та забезпечує якісну збереження та обробку даних.

ВИСНОВКИ

В даній курсовій роботі було розглянуто процес розробки інформаційної системи з використанням системи керування базами даних Strapi. Було проведено аналіз предметної області, визначено об'єкт і предмет дослідження, а також встановлено мету та завдання роботи.
Було розроблено бізнес-процеси та діаграму варіантів використання, які дозволяють описати взаємодію користувачів з системою. На основі діаграм було проведено аналіз вимог до системи та визначено функціональні та нефункціональні вимоги до неї.

Далі було розроблено діаграму компонентів, яка дозволяє описати структуру системи та взаємодію компонентів між собою.

Було проведено проектування інтерфейсу та реалізовано його демонстрацію. Для цього було визначено вимоги до інтерфейсу, розроблено його дизайн та виконано імплементацію.

Крім того, було проведено моделювання даних та розроблено структуру бази даних, необхідну для роботи інформаційної системи.

Отже, розробка інформаційної системи з використанням Strapi є актуальною та корисною задачею. Реалізація такої системи дозволяє ефективно керувати базами даних, взаємодіяти з ними та забезпечувати потрібні функції користувачам. В цілому, результати даної роботи можуть бути корисні для подальшого розвитку інформаційних систем та застосування їх у практичній діяльності.

Перелік посилань

1. Офіційна документація Strapi: https://strapi.io/documentation

2. Статті на тему Strapi на сайті Medium: https://medium.com/tag/strapi
3. Документація з баз даних MySQL та PostgreSQL, які підтримуються Strapi: https://dev.mysql.com/doc/ та https://www.postgresql.org/docs/