МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

КУРСОВИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Моделювання та аналіз програмного забезпечення»

на тему:

«Магазин дитячого одягу та взуття»

студента IV курсу групи ІПЗ-19-1
спеціальності 121 «Інженерія програмного забезпечення»
Драка Тараса Сергійовича
(прізвище, ім'я та по-батькові)

Керівник: доцент кафедри ІПЗ
Сугоняк І.І.

Дата захисту: " ___ " _____ 20__ р.
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії _____ І.І. Сугоняк
(підпис) (прізвище та ініціали)
_____ О.В. Власенко
(підпис) (прізвище та ініціали)
_____ С.М. Кравченко
(підпис) (прізвище та ініціали)

Житомир – 2022

Зміст

ВСТУП

Тема онлайн магазинів стає все популярнішою останнім часом на цей час майже усі підприємства мають свій веб додаток. З'являється все більше спільнот, форумів, YouTube-каналів, веб-сайтів та інших джерел інформації і місць для обговорення свіжих тенденцій у дитячій моді та покупках для дітей. Проте існуючі ресурси не завжди вирішують деякі проблеми, які можуть відштовхнути людей від покупок у цьому сегменті ринку, особливо новачків, які хочуть детальніше ознайомитися зі світом дитячої моди. Саме тому було вирішено розробити власний інтернет-магазин дитячого одягу з унікальною системою.

Метою курсового проекту є дослідження особливостей моделювання та аналізу програмних комплексів для інтернет-магазину дитячого одягу.

Завданням на курсову роботу є:

- Аналіз теоретичних відомостей щодо моделювання програмного забезпечення;
- Вивчення та опис вимог користувачів;
- Використання методів моделювання функцій та поведінки системи;
- Проектування об’єктної структури системи;
- Моделювання програмних комплексів;
- Автоматична генерація коду з моделей.

Предметом дослідження є можливості застосування CASE-засобів у процесі проектування програмного забезпечення для інтернет-магазину дитячого одягу.

Об’єктом дослідження є методи та засоби проектування програмного забезпечення та уніфікація процесу проектування.

1 АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1 Технічне завдання на розробку системи

1. Назва системи:

   Повне найменування системи: Інтернет-магазин дитячого одягу

   Коротке найменування системи: Дитячий одяг, Онлайн-магазин, Дитяча мода (ДО), Дитячий магазин (ДМ), Інтернет-магазин (ІМ).

2. Призначення системи:

   Програмний комплекс створений для забезпечення зручного та швидкого доступу до дитячого одягу, сприяння обговоренню модних тенденцій та спрощення процесу покупки товарів для дітей. Онлайн-магазин надає можливість переглядати, вибирати та замовляти дитячий одяг.

3. Вимоги до системи:

   Програмний продукт повинен мати наступні функціональні характеристики:

   - приємний та інтуїтивний дизайн;
   - публікація, редагування, перегляд товарів;
   - можливість одночасної роботи з системою як мінімум 70 користувачів;
   - наявність трьох ролей користувачів (адміністратор, гість, користувач).

4. Вимоги до програмного забезпечення:

   Програмний комплекс повинен бути побудований на клієнт-серверній мікросервісній архітектурі з використанням веб-технологій, що не вимагають додаткового ліцензування. Серверна частина програмного комплексу буде працювати на веб-сервері з Windows і Php та системі керування базами даних MySql.

5. Вимоги до інтерфейсу:

   Користувач має взаємодіяти з продуктом за допомогою браузерів. Відповідно, інтерфейс має бути функціональним, незалежно від браузера, що використовується, за умови що браузер є сучасним і популярним. Інтерфейс має бути зручним і функціональним. Час відклику на дії користувача має бути достатньо малим, щоб не викликати дискомфорту.

   До інтерфейсу є наступні вимоги

   - Мінімальний можливий час відгуку;
   - повідомлення про помилки/критичні ситуації системи;

1.2 Обґрунтування вибору засобів моделювання

На даний момент існує велика кількість додатків для моделювання ПЗ, в тому числі тих, які підтримують створення UML-діаграм

Для вибору кращого інструменту для вирішення заданої задачі було обрано наступні редактори: StarUML, Draw.IO.

StarUML

StarUML – це редактор, що доступний як для Windows так і для Linux. Для не комерційного використання він є безкоштовним, що, однозначно є плюсом. Він підтримує усі стандарти UML і оновлюється відповідно до еволюції цієї мови. Також, існує купа плагінів для цього редактору. Наприкад ті, що дозволяють генерувати програмний код відповідно до створеної діаграми. Його інтерфейс є інтуїтивно зрозумілим і зручним у використанні

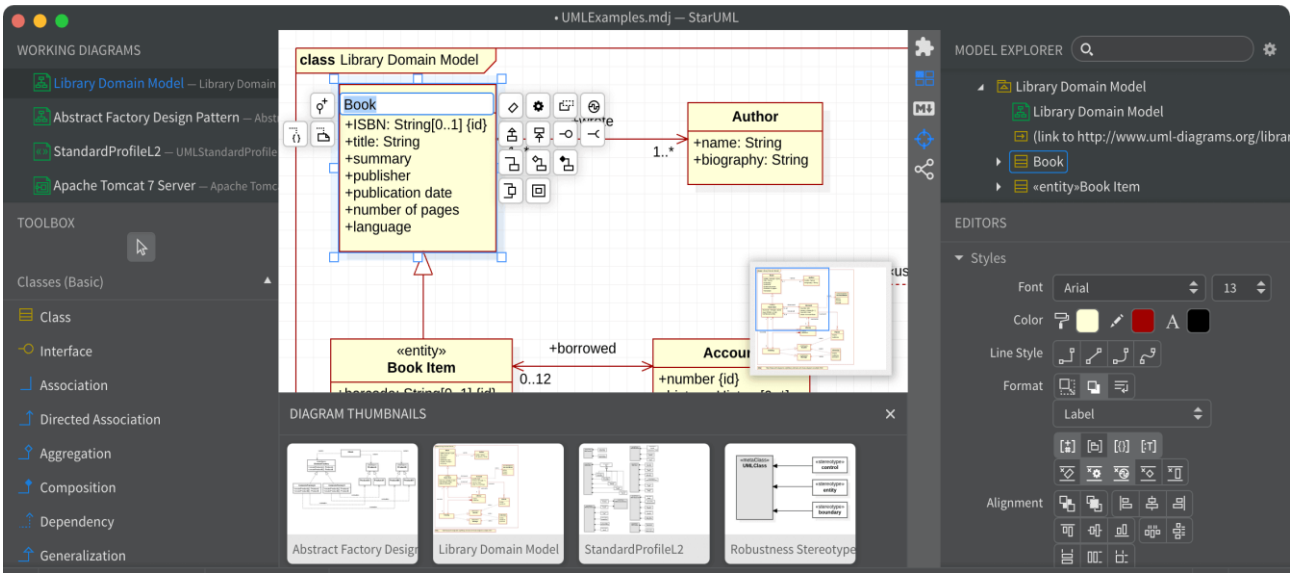

Рис. 1.1 – Інтерфейс програми StarUML

Draw.IO

Draw.IO – це безкоштовний кростплатформений веб- і десктопний додаток для створення діаграм, схем, графіків і так далі з відкритим кодом, розроблений за допомогою HTML і JavaScript. Спектр схем і діаграм, що можуть бути створені є досить широким. Підтримується експорт і імпорт, формати файлів png, jpeg, svg, drawio, xml, pdf. Відповідно, drawio і xml використовуються для імпорту і експорту, а всі інші-виключно для експорту. Його функціонал менший ніж у StarUML, а також він не є таким зручним у використанні як попередній додаток.

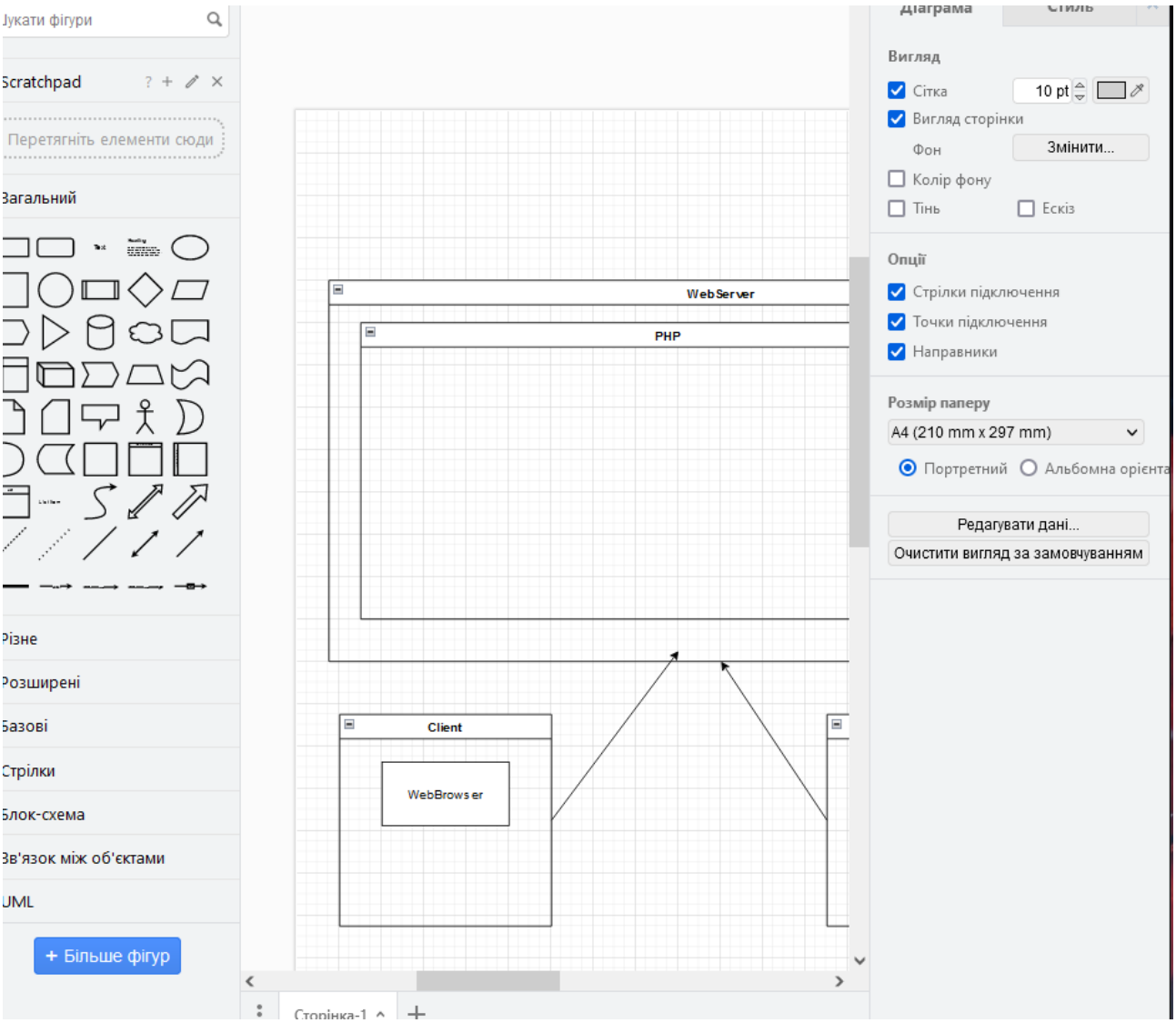

Рис. 1.2 – Інтерфейс програми Draw.IO

Для наочності порівняння отримані результати подані в таблиці 1.1.

Таблиця 1.1

	Простота використання	Зручність	Наявність всіх необхідних можливостей	Кодогенерація	Експорт/Імпорт	Підтримка нових версій UML
StarUML	+	+	+	+	+	+
Draw.IO	+	+-	-	-	+	+-

1.3 Аналіз вимог до програмного продукту

Розглянемо вимоги, щоб система інтернет-магазину дитячого одягу задовольняла потреби користувачів і виконувала свої функції.

Бізнес-вимоги:

1. Основні цілі: Розробка інтернет-магазину дитячого одягу з метою надання зручного доступу до широкого асортименту якісного дитячого одягу та сприяння поширенню продукції.
2. Представлення проекту: Реалізація проекту у вигляді функціонального веб-сайту, що надає можливість переглядати і купувати дитячий одяг, а також забезпечує зручну навігацію та функціонал для забезпечення задоволення покупців.
3. Час відгуку сайту на дії користувача: Максимальний час завантаження сторінки - до 3 секунд
4. Можливість розширення функціоналу та обсягу товарів у магазині.

Вимоги користувачів

- Зовнішні користувачі (відвідувачі сайту)
    1. Можливість перегляду каталогу дитячого одягу без авторизації.
    2. Пошук товарів за різними параметрами (колір, тип одягу і т.д.).
    3. Перегляд детальної інформації про товар (опис, ціна, наявність у складі, фото).
- Авторизовані користувачі
    1. Усі можливості зовнішніх користувачів.

2. Авторизація в системі за допомогою облікового запису.
3. Додавання товарів у кошик для подальшої покупки.
4. Оформлення замовлення та введення необхідних даних для доставки.
5. Перегляд статусу замовлення та історії покупок.

- Внутрішній користувач-адміністратор, модератор, автор
  1. Усі можливості авторизованих користувачів.
  2. Додавання, редагування та видалення товарів, категорій та брендів.

Функціональні вимоги

1. Авторизація користувачів в системі.
   - Система повинна надавати можливість реєстрації нових користувачів з обов'язковим заповненням необхідних полів.
   - Після реєстрації користувач отримує обліковий запис з унікальним ідентифікатором та паролем.
   - Користувачі повинні мати різні ролі: адміністратор, гість , користувач.
2. Збереження інформації.
   - Збереження профілів користувачів з особистою інформацією та історією покупок.
   - Збереження каталогу дитячого одягу з описом, цінами та наявністю товарів.

Нефункціональні вимоги

1. Сприйняття.

- Час, необхідний для навчання користувачів роботі з системою - до 30 хвилин для звичайних користувачів та до 15 хвилин для досвідчених.
- Максимальний час відповіді системи на запити - до 5 секунд для запитів.
- Інтуїтивно зрозумілий та зручний інтерфейс користувача, не вимагаючий додаткової підготовки.

2. Надійність
   - Доступність системи повинна становити не менше 90% часу роботи.
   - Середній час безперервної роботи системи - не менше 20 робочих днів.

Системні вимоги

1. Вимоги до середовища виконання:

Система повинна задовільняти вимогам на компьютері, що знаходиться в наступній мінімальній комлектації:

- 4ГБ оперативної пам'яті
- Процессор з тактовою частотою 1.5GHz
- Операційна система Ubuntu 20, 22, Windows 7, 8, 10, 11
- Вимоги до операційного середовища (для сервера UNIX, до web-сервера Apache).
- Браузери (Chrome, Firefox, …).

2. Вимоги до СУБД та доступу до данних

- У ядрі системи повинна бути представленна СУБД реляційного доступу

Бізнес логіка

1. Продаж дитячого одягу: Основною метою існування інтернет-магазину є продаж дитячого одягу. Клієнти мають змогу переглядати каталог, вибирати товари, додавати їх до кошика та здійснювати покупки.

2. Керування інвентарем: Система повинна вести облік наявності товарів у магазині. При купівлі товару його кількість в інвентарі зменшується автоматично, щоб уникнути продажу вичерпаного товару.

Підсумовуючи, побудуємо діаграму варіантів використання (Use Case Diagram).

Специфікацію прецендентів наведено в додатку А.

На рисунку 1.4 зображено побудовану діаграму варіантів використання

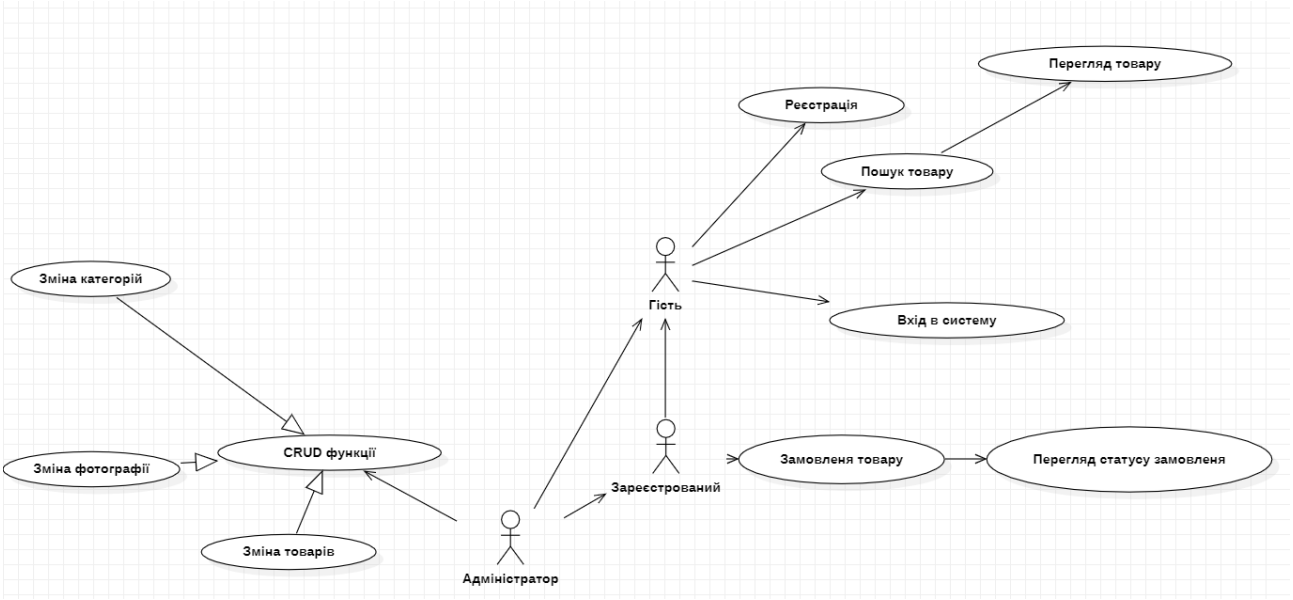

Рис. 1.3 – Діаграма варіантів використання

2 РОЗРОБКА МОДЕЛІ ПРОГРАМНОГО КОМПЛЕКСУ НА ЛОГІЧНОМУ РІВНІ

2.1 Алгоритм роботи та стани програмної системи

Для зображення алгоритму роботи та можливих станів програмної системи добре підійде діаграма активностей (activity diagram).

Діаграма активності за своєю суттю нагадує блок-схему, але з більш конкретним моделюванням потоку виконання від однієї дії до іншої.

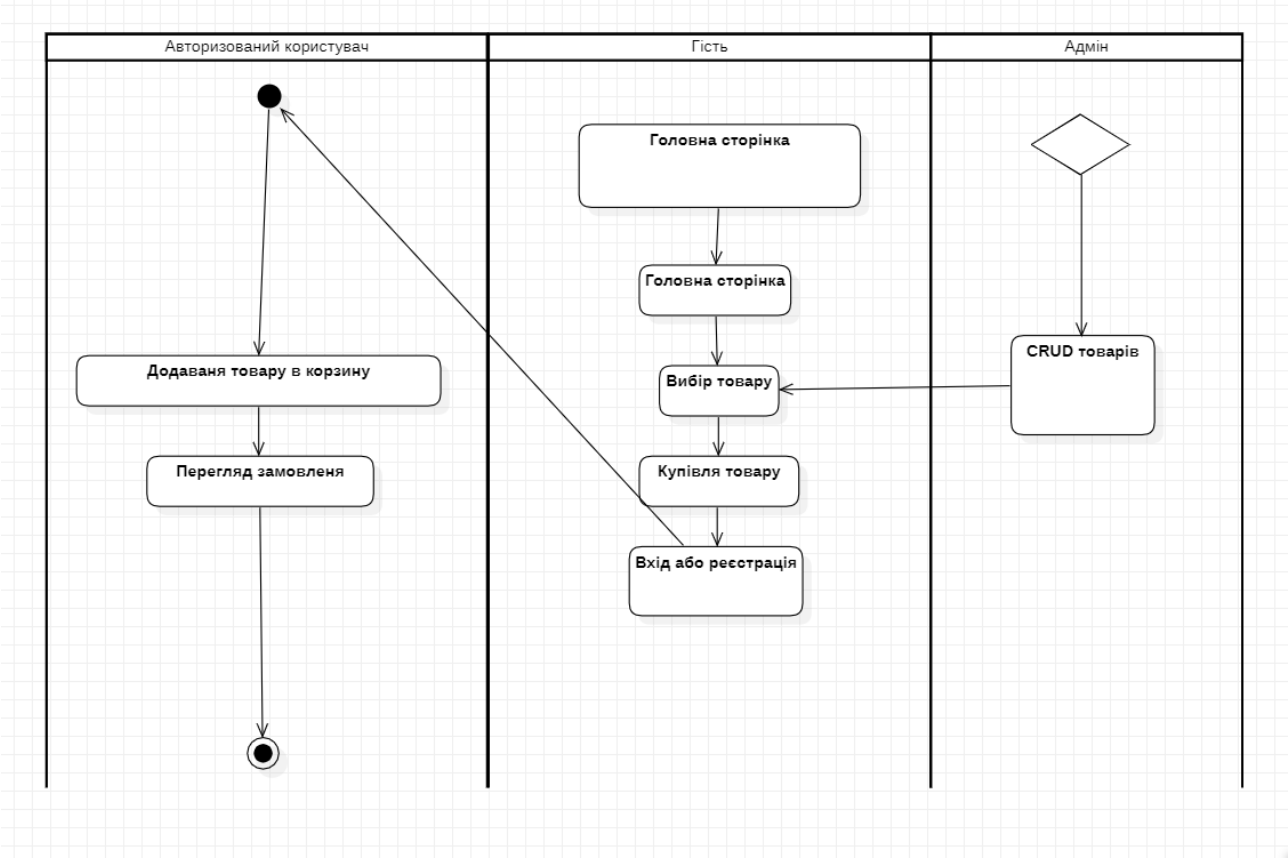

Рис. 2.1. Діаграма активності для реєстрації і придбання товару

Опис діаграми активностей:

1. Головна сторінка: Користувач відкриває інтернет-магазин.
2. Вибір товару: Користувач переглядає доступні категорії товарів, наприклад, "Дитячі футболки", "Штани для дітей" тощо;
3. Купівля товару: Користувач додає товар до свого кошика для покупки.
4. Проводиться запит на вхід для користувача у разі потреби він проходить реєстрацію.
5. Додавання товару до корзини: Користувач переглядає зміст свого кошика, кількість доданих товарів та проводить купівлю.

6. Перегляд замовлення: Користувач заповнює необхідну інформацію (адреса доставки, контактні дані тощо) для оформлення замовлення.
7. Завершення: Процес покупки завершується, і користувач може повернутися до початкового стану або продовжити покупки.

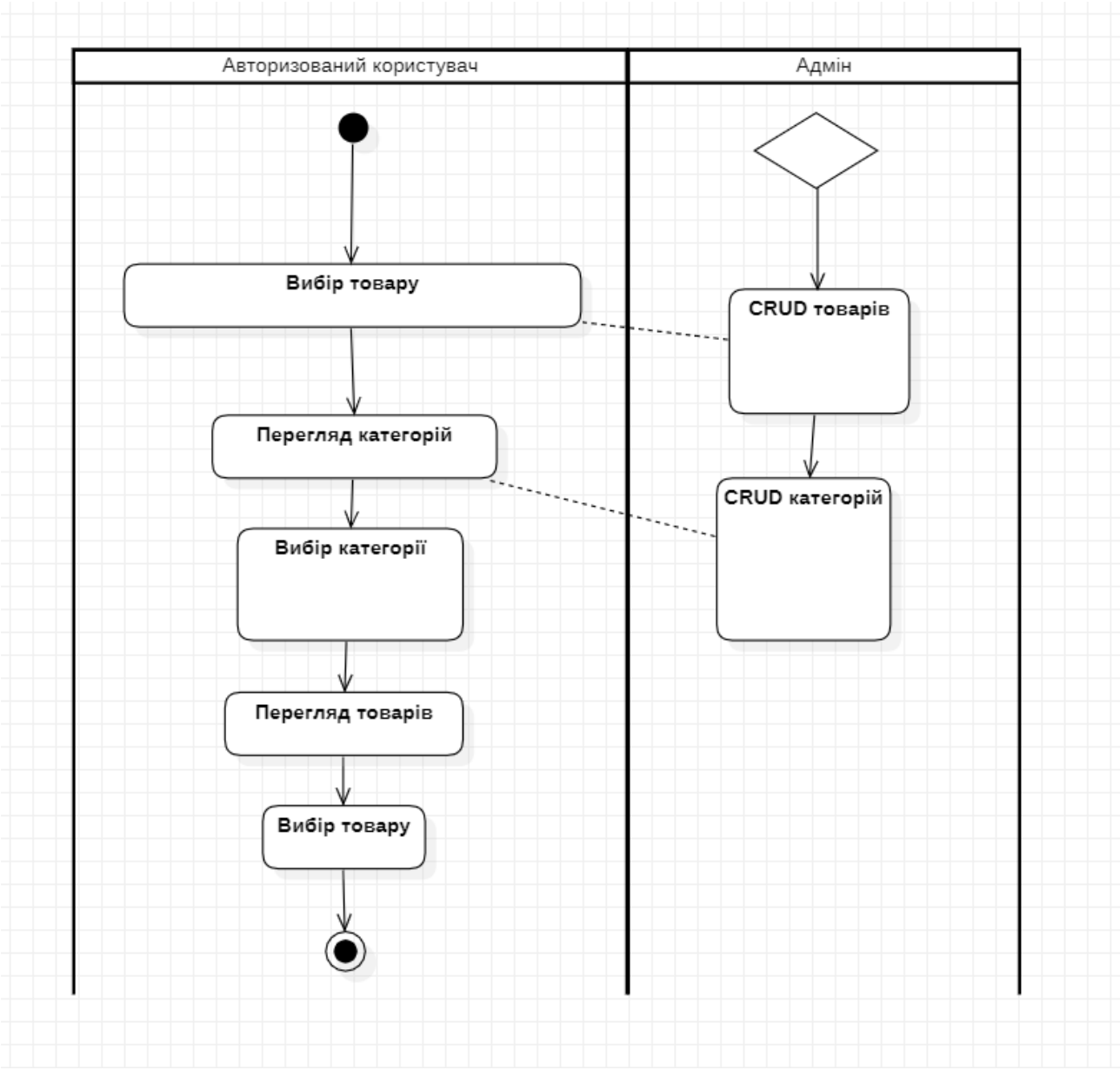

Рис. 2.2. Діаграма дій з товаром

Опис діаграми активностей:

1. Користувач переглядає товари.

2. Перегляд категорій: Користувач переглядає доступні категорії товарів, наприклад, "Дитячі футболки", "Штани для дітей" і т. д.
3. Вибір категорії: Користувач вибирає певну категорію, яку бажає дослідити більше.
4. Перегляд товарів: Користувач переглядає список доступних товарів у вибраній категорії.
5. Вибір товару: Користувач обирає конкретний товар з переліку.
6. CRUD категорій(для адміністратора): Адміністратор має можливість редагувати категорії товарів, додавати нові категорії, видаляти чи змінювати наявні категорії
7. CRUD товарів (для адміністратора): Адміністратор може редагувати властивості товарів, такі як назва, опис, ціна, зображення, наявність тощо. Він також може додавати нові товари або видаляти ті, що більше не доступні

2.2 Об’єктно-орієнтована модель системи

Об’єктно-орієнтовану модель системи прийнято розглядати у вигляді діаграми класів (class diagram). Для побудови даної діаграми спочатку потрібно розібратися, що воно собою представляє. Діаграма класів – це тип діаграми статичної структури, який описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) і зв’язки між об’єктами.

На рисунку 2.3 зображено побудовану діаграму класів

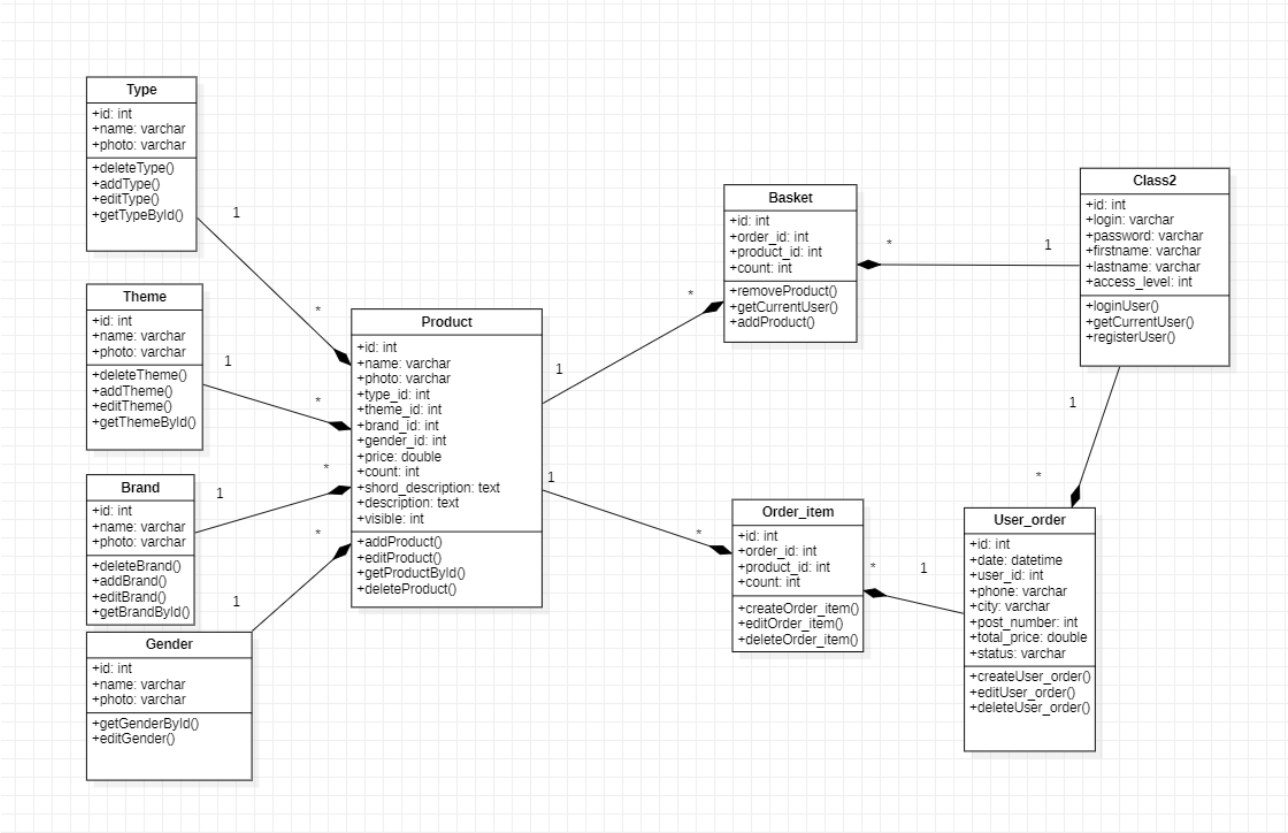

Рис. 2.3. Діаграма класів

Патерни проектування

Як вже було згадано раніше, патерни проектування є рішеннями для поширених проблем, які можуть виникнути під час розробки. Вони допомагають прискорити процес розробки, оскільки вирішують відомі проблеми ще до їх появи під час реалізації. Патерн - це загальний принцип вирішення проблеми, і його застосування потребує налаштування під конкретну програму, оскільки копіювання коду не є відповідним підходом. Хоча знання патернів не є обов'язковими, можна успішно працювати і без них, оскільки ви можливо вже використовували деякі патерни без свідомості про це.

Однак, знання патернів дозволяє ефективніше вирішувати задачі та розробляти програмне забезпечення. Вони надають перевірені рішення, стандартизують код та створюють загальний словник програмістів. Патерни

різняться за складністю, деталізацією та охопленням проектованої системи. Розглянемо основні групи патернів:

- Породжуючі патерни: забезпечують гнучке створення об'єктів без зайвих залежностей у програмі.
- Структурні патерни: пропонують різні способи побудови зв'язків між об'єктами.
- Поведінкові патерни: зосереджуються на ефективній комунікації між об'єктами.

Один з прикладів породжуючого патерну - прототип. Він дозволяє копіювати об'єкти без необхідності детального знання їхньої реалізації.

2.3 Взаємодія об’єктів системи

Взаємодію об’єктів системи в UML прийнято зображати у вигляді діаграми послідовності.

Діаграми послідовності – це діаграми взаємодії, які детально описують, як виконуються операції. Вони фіксують взаємодію між об’єктами в контексті співпраці. Діаграми послідовності візуально показують порядок взаємодії і те, які саме дії будуть виконані.

Було вирішено побудувати діаграму послідовності для написання статті

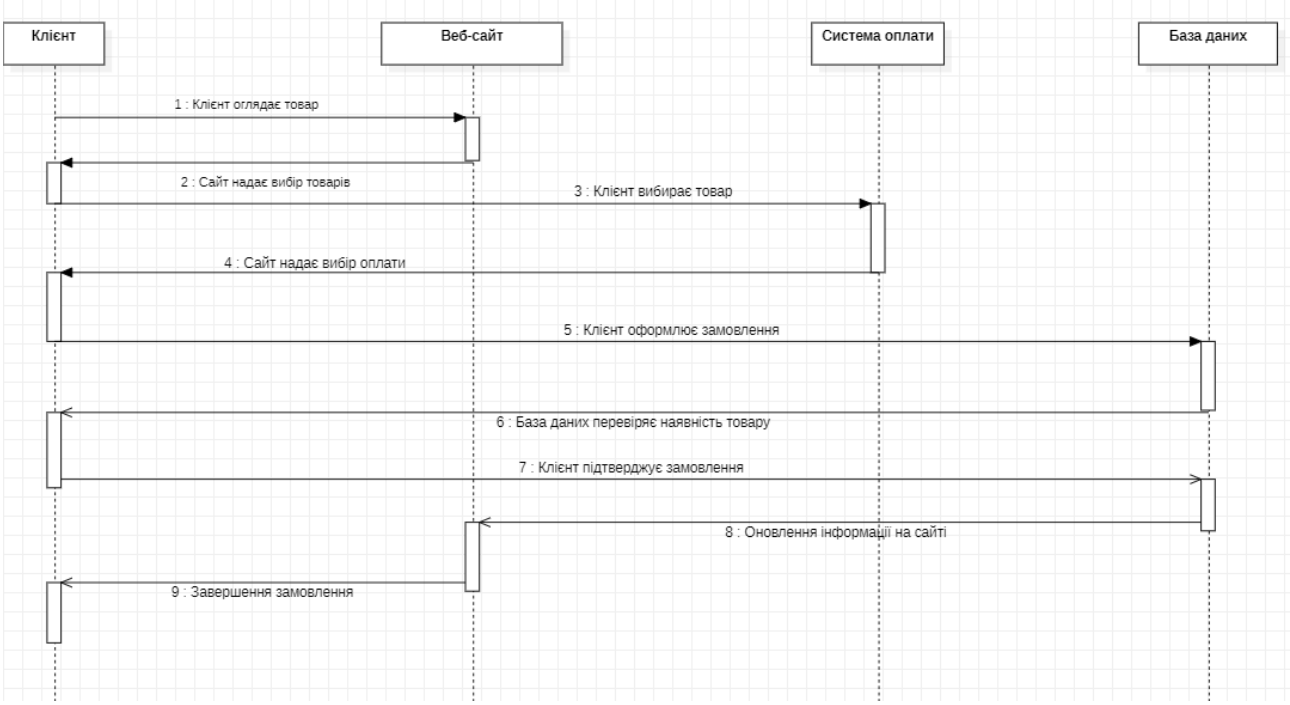

Рис. 2.5. Діаграма послідовності для купівлі товару

Опис діаграми послідовності:

1. Клієнт оглядає товар
2. Сайт надає вибір товарів
3. Клієнт обирає товар
4. Система оплати надає запит вибору оплати
5. Клієнт обирає тип оплати
6. Бд повторно перевіряє наявність товару
7. Клієнт підтверджує замовлення
8. Бд оновлює інформацію про товар
9. Користувач завершує замовлення

3 ФІЗИЧНА МОДЕЛЬ ТА ПРОТОТИП ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Взаємодія компонентів системи

Для зображення взаємодії компонентів системи мовою UML передбачено окремий вид діаграм – діаграму компонентів (component diagram).

Цей тип діаграм використовується для моделювання фізичних аспектів об’єктно-орієнтованих систем. Діаграми компонентів – це, по суті, діаграми класів, які зосереджуються на компонентах системи, які часто використовуються для моделювання статичної реалізації системи.

На рисунку 3.1 зображено діаграму компонентів.

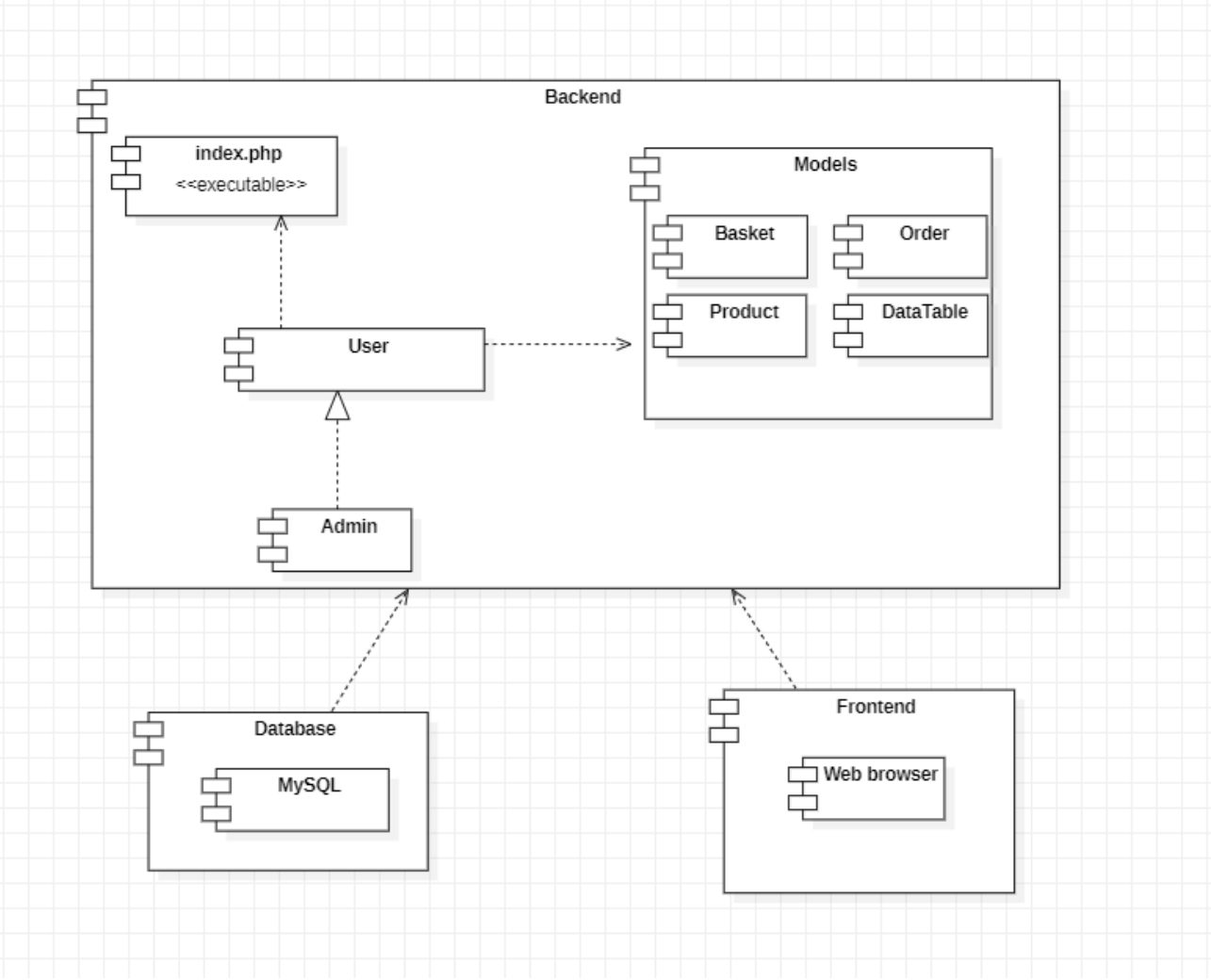

Рис. 3. 1. Діаграма компонентів

Опис діаграми компонентів:

- Models – компоненти, які є модулями в програмному коді
- Залежності.
- Відношення залежності.
- Database Server – локальний сервер бази даних:MySql
- Frontend – пристрій користувчач з якого він може користуватись додатком:Css

3.2 Архітектура програмного комплексу та його розгортання

При проектуванні системи було вирішено використовувати мікросервісну архітектуру для більшої гнучкості.

До переваг цієї архітектури належать:

- Модульність. Тобто частини системи не є жорстко пов’язаними між собою, а можуть змінюватись, не викликаючи при цьому фатальних проблем у інших частин
- Масштабованість. Таку систему простіше підтримувати і розвивати, додавати новий функціонал, у силу попереднього плюсу-модульності. Новий функціонал не заважає працювати вже існуючому. Також у разі вищих навантажень дуже просто розгорнути додаткові потужності
- Надійність. У разі відмови перестає працювати лише один сервіс, що у більшості ситуацій не є фатальним для роботи системи.
- Тестованість. Таку архітектуру простіше тестувати, адже досить просто локалізувати джерело помилки і виправити її

Для зображення розгортання системи існує однойменна діаграма – діаграма розгортання (deployment diagram).

Діаграма розгортання – це діаграма, яка показує конфігурацію вузлів обробки часу виконання та компонентів, які на них живуть. Діаграми розгортання – це різновид структурної діаграми, яка використовується для

моделювання фізичних аспектів об’єктно-орієнтованої системи. Вони часто використовуються для моделювання статичного вигляду розгортання системи (топології апаратного забезпечення).

На рисунку 3.2 зображено діаграму розгортання.

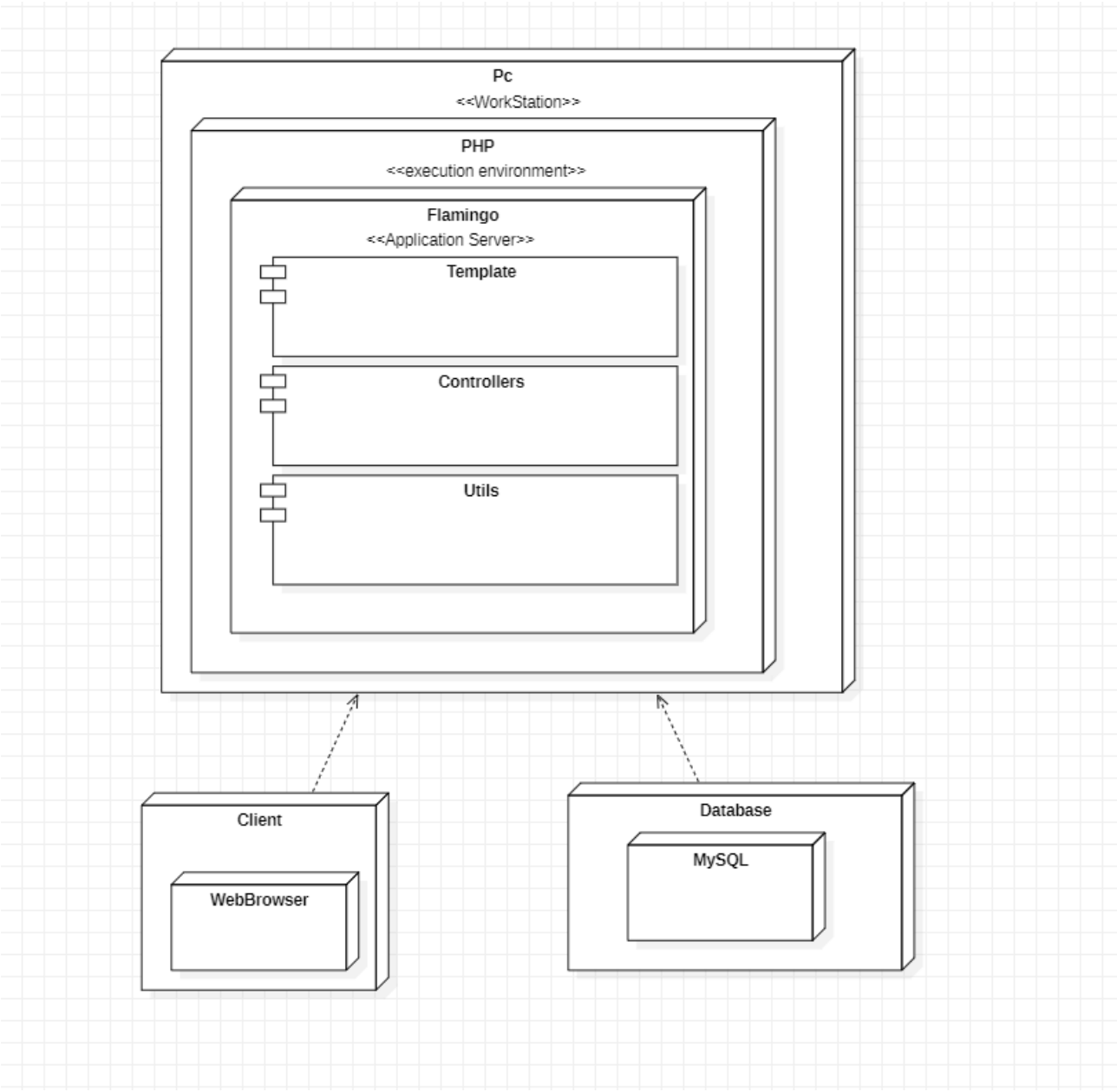

Рис. 3.2. Діаграма розгортання

3.3 Генерування програмного коду для прототипу програми

Під час вибору засобів моделювання було акцентовано на наявності кодогенерації. Зазначимо, що кодогенерація – це процес автоматичного перетворення діаграм UML у код програми.

Для генерації коду необхідна наявність діаграми класів та розширення для необхідної мови програмування, в нашому випадку PHP.

Для генерації PHP у StarUML необхідно виконати такий порядок дій:

1. Відкрити файл з створеною раніше діаграмою класів;
2. Перейти у StarUML – Tools – Extension Manager та встановити розширення PHP;
3. Перезавантажити StarUML;
4. Перейти у StarUML – Tools – PHP – Generate code та обрати класи, для яких потрібно згенерувати код;
5. Згенерований код з’явиться у обраній вами папці, яку запропонує обрати редактор.

Згенеровані класи спроектованої системи зображені на рисунку 3.3,

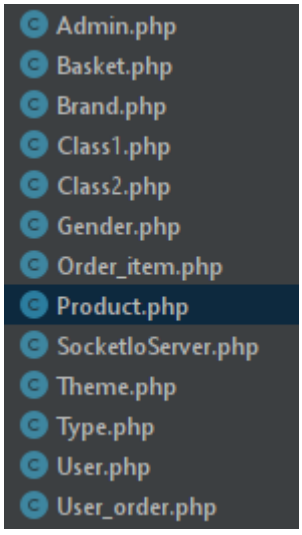

Рисунок 3.3 – Файли згенерованих класів.

ВИСНОВКИ

Отже, під час виконання даної курсової роботи було закріплено пройдений матеріал, який вивчався протягом курсу, а саме: моделювання та аналіз програмного забезпечення за допомогою мови UML. Виконання курсового проекту було розділено на 3 етапи.

У першому розділі було отримано технічне завдання на розробку системи. Було обрано засіб моделювання для вирішення поставленого завдання. Також, було виокремлено та проаналізовано вимоги до програмного продукту.

У другому розділі було розроблено модель програмного комплексу на логічному рівні. Було змодельовано діаграму активності для демонстрації алгоритму роботи та можливих станів програмної системи та діаграму класів для презентування об’єктно-орієнтованої моделі системи. Імплементовано патерни проектування. Також, було спроектовано діаграму послідовності для зображення взаємодії об’єктів системи.

У третьому розділі було створено фізичну модель та прототип програмного комплексу. Було продемонстровано взаємодію компонентів системи за допомогою побудови діаграми компонентів. Було обґрунтовано вибір архітектури програмного комплексу. Також, було побудовано діаграму розгортання. І було згенеровано програмний код з діаграми класів

Отже, виконана робота показала, що за допомогою вивченого матеріалу за курс можливо успішно моделювати програмні комплекси, використовуючи мову UML.

1 АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

ДЖЕРЕЛА ІНФОРМАЦІЇ

1. StarUML documentation: Introduction. URL: https://docs.staruml.io.
2. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – https://evergreens.com.ua/ua/articles/uml-diagrams.html.
3. Графічний редактор DRAW.IO [Електронний ресурс] – https://app.diagrams.net/.
4. Use Case Diagram [Електронний ресурс] – https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/.
5. Activity Diagram [Електронний ресурс] – https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/.
6. Class Diagram [Електронний ресурс] – https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/.
7. Factory Pattern [Електронний ресурс] https://refactoring.guru/uk/design-patterns/factory-method
8. Sequence Diagram [Електронний ресурс] – https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.
9. Deployment Diagram [Електронний ресурс] – https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/.
10. Unified Modeling Language [Електронний ресурс] – https://uk.wikipedia.org/wiki/Unified_Modeling_Language.

11. Laravel Documentation [Електронний ресурс] https://laravel.com/docs/10.x/readme
12. TypeScript Documentation [Електронний ресурс] https://www.typescriptlang.org/docs/
13. MySQL Documentation [Електронний ресурс] https://dev.mysql.com/doc/
14. Composer Documentation [Електронний ресурс] https://getcomposer.org/doc/
15. PHP Documention [Електронний ресурс] https://www.php.net/docs.php

ДОДАТКИ

Basket.php

```
<?php


namespace models;


use core\Core;

class Basket
{
    protected static $tableName = 'basket';
    public static function addProduct($product_id, $count = 1) {
        $rowProduct = Product::getById($product_id);
        $user_id = User::getCurrentAuthenticatedUser()['id'];
        if(isset($rowProduct)) {
            $rowBasket = Core::getInstance()->db-
>select(self::$tableName, '*', [
                'user_id' => $user_id,
                'product_id' => $product_id
            ])[0];
            if(isset($rowBasket)) {
                Core::getInstance()->db->update(self::$tableName,
[
                    'count' => $rowBasket['count'] + $count
                ], [
                    'id' => $rowBasket['id']
                ]);
            }
            else {
                Core::getInstance()->db->insert(self::$tableName,
[
                    'user_id' => $user_id,
                    'product_id' => $rowProduct['id'],
                    'count' => $count
                ]);
            }
        }
    }
    public static function getProducts() {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', [
            'user_id' => User::getCurrentAuthenticatedUser()['id']
        ]);
        $result = [];
        $products = [];
        $totalPrice = 0;
        foreach ($rows as $row) {
            $product = Product::getById($row['product_id']);
            $products [] = ['product' => $product, 'count' =>
$row['count']];
```

```
            if($row['count'] > $product['count'])
                continue;
            $totalPrice += $product['price'] * $row['count'];
        }
        $result['products'] = $products;
        $result['total_price'] = $totalPrice;
        return $result;
    }
    public static function getCountOfId($product_id) {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', [
            'user_id' =>
User::getCurrentAuthenticatedUser()['id'],
            'product_id' => $product_id
        ]);
        if(isset($rows))
            return $rows[0]['count'];
    }
    public static function clear() {
        Core::getInstance()->db->delete(self::$tableName, [
            'user_id' => User::getCurrentAuthenticatedUser()['id']
        ]);
    }
    public static function removeItem($product_id) {
        Core::getInstance()->db->delete(self::$tableName, [
            'user_id' =>
User::getCurrentAuthenticatedUser()['id'],
            'product_id' => $product_id
        ]);
    }
    public static function getCountItems() {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', [
            'user_id' => User::getCurrentAuthenticatedUser()['id']
        ]);
        if(isset($rows))
            return array_sum(array_column($rows, 'count'));
        else
            return 0;
    }
    public static function reduceOverflowItems() {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', [
            'user_id' => User::getCurrentAuthenticatedUser()['id']
        ]);
        foreach ($rows as $rowBasket) {
            $rowProduct =
Product::getById($rowBasket['product_id']);
            if($rowBasket['count'] > $rowProduct['count']) {
                Core::getInstance()->db->update(self::$tableName,
[
                    'count' => $rowProduct['count']
                ], [
```

```
                    'id' => $rowBasket['id']
                ]);
            }
        }
    }
    public static function isEnoughStock() {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', [
            'user_id' => User::getCurrentAuthenticatedUser()['id']
        ]);
        foreach ($rows as $rowBasket) {
            $rowProduct =
Product::getById($rowBasket['product_id']);
            if($rowBasket['count'] > $rowProduct['count']) {
                return false;
            }
        }
        return true;
    }
}
```

DataTable.php

```
<?php
namespace models;

use core\Core;
use utils\Photo;

class  DataTable {
    public static function addItem($tableName, $name, $fileName) {
        Core::getInstance()->db->insert($tableName, [
            'name' => $name,
            'photo' => $fileName
        ]);
    }
    public static function getById($tableName, $id) {
        $rows = Core::getInstance()->db->select($tableName, '*', [
            'id' => $id
        ]);
        if(!empty($rows))
            return $rows[0];
        else
            return null;
    }
    public static function getNameById($tableName, $id) {
        $row = self::getById($tableName, $id);
        if(!empty($row))
            return $row['name'];
        else
            return null;
    }
    public static function deleteItem($tableName, $id) {
```

```
        $rows = Core::getInstance()->db->delete($tableName, [
            'id' => $id
        ]);
    }
    public static function updateItem($tableName, $id, $newName) {
        $rows = Core::getInstance()->db->update($tableName, [
            'name' => $newName
        ], [
            'id' => $id
        ]);
    }
    public static function getItems($tableName) {
        $rows = Core::getInstance()->db->select($tableName);
        $rows = array_reverse($rows);
        return $rows;
    }
    public static function getSortedItems($tableName) {
        $rows = self::getItems($tableName);
        $name = array_column($rows, 'name');
        array_multisort($name, SORT_ASC, $rows);
        // Move 'Default' item to the first place
        $i = 0;
        foreach ($rows as $key => $value) {
            if($value['name'] === 'Default') {
                $default = $rows[$i];
                unset($rows[$key]);
                $rows = array($default) + $rows;
                break;
            }
            $i++;
        }
        return $rows;
    }
}
```

Order.php

```
<?php


namespace models;


use core\Core;
use utils\Filter;

class Order
{
    protected static $tableName = 'user_order';
    public static function addOrder($row) {
        $products = $row['products'];

        $fieldsList = ['date', 'user_id', 'name',
```

```
            'phone', 'city', 'post_number', 'total_price'];
        $rowOrder = Filter::filterArray($row, $fieldsList);
        Core::getInstance()->db->insert(self::$tableName,
$rowOrder);

        $orderId = self::getOrderId($rowOrder['user_id'],
$rowOrder['date']);
        foreach ($products as $product) {
            if($product['product']['count'] != 0) {
                $product['product']['count'] -= $product['count'];
                Product::update($product['product']['id'],
$product['product']);
                Core::getInstance()->db->insert('order_item', [
                    'order_id' => $orderId,
                    'product_id' => $product['product']['id'],
                    'count' => $product['count']
                ]);
            }
        }
    }
    public static function getOrderId($userId, $dateTime) {
        $ids = Core::getInstance()->db->select(self::$tableName,
'*', [
            'user_id' => $userId,
            'date' => $dateTime
        ]);
        if(isset($ids))
            return $ids[0]['id'];
        else
            return null;
    }
    public static function getUserOrdersById($userId) {
        $orders = Core::getInstance()->db-
>select(self::$tableName, '*', [
            'user_id' => $userId
        ]);
        return array_reverse($orders);
    }
    public static function getOrdersByStatus($statusName) {
        $orders = Core::getInstance()->db-
>select(self::$tableName, '*', [
            'status' => $statusName
        ]);
        return array_reverse($orders);
    }
    public static function approve($id) {
        Core::getInstance()->db->update(self::$tableName, [
            'status' => 'approved'
        ],[
            'id' => $id
        ]);
    }
    public static function getOrderItems($order_id) {
```

```
        $rows = Core::getInstance()->db->select('order_item', '*',
[
            'order_id' => $order_id
        ]);
        $products = [];
        foreach ($rows as $row) {
            $product = Product::getById($row['product_id']);
            $products [] = ['product' => $product,
'count_in_order' => $row['count']];
        }
        return $products;
    }
}
```

Product.php

```
<?php
namespace models;

use core\Core;
use utils\Filter;

class Product
{
    protected static $tableName = 'product';
    public static function add($row, $filePath) {
        $fieldsList = ['name', 'photo', 'gender_id', 'theme_id',
'type_id',
            'brand_id', 'price', 'count', 'short_description',
'description', 'visible'];
        $row = Filter::filterArray($row, $fieldsList);
        $row += ['photo' => $filePath];
        Core::getInstance()->db->insert(self::$tableName, $row);
    }

    public static function delete($id) {
        Core::getInstance()->db->delete(self::$tableName, [
            'id' => $id
        ]);
    }
    public static function update($id, $row) {
        $fieldaslist = ['name', 'photo', 'gender_id', 'theme_id',
'type_id',
            'brand_id', 'price', 'count', 'short_description',
'description', 'visible'];
        $row = Filter::filterArray($row, $fieldaslist);
        Core::getInstance()->db->update(self::$tableName, $row, [
           'id' => $id
        ]);
    }
    public static function getAllProducts() {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*');
        return array_reverse($rows);
```

```
    }
    public static function count() {
        return count(self::getAllProducts());
    }
    public static function getOnePage($rows, $currentPage, $count)
{
        $startItem = ($currentPage - 1) * $count;
        $pageItems = array_slice($rows, $startItem, $count);
        return $pageItems;
    }
    public static function getById($id) {
        $row = Core::getInstance()->db->select(self::$tableName,
'*', [
            'id' => $id
        ]);
        if(!empty($row))
            return $row[0];
        else
            return null;
    }
    public static function getProductsIn($tableName, $id) {
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', [
            $tableName.'_id' => $id
        ]);
        return array_reverse($rows);
    }
    public static function setDefaultForNull($column, $id) {
        $row = self::getById($id);
        $row["{$column}_id"] = 1;
        self::update($row['id'], $row);
    }
    public static function search($searchString, $all) {
        $conditionsArray = [
            'name' => "%{$searchString}%"
        ];
        if(!$all)
            $conditionsArray += ['visible' => 1];
        $rows = Core::getInstance()->db->select(self::$tableName,
'*', $conditionsArray, 'LIKE');
        return array_reverse($rows);
    }
}
```

User.php

```
<?php
namespace  models;

use core\Utils;

class User
```

```
{
    protected  static $tableName = 'user';
    public static function addUser($login, $password, $firstname,
$lastname) {
        \core\Core::getInstance()->db->insert(
          self::$tableName, [
                'login' => $login,
                'password' => self::hashPassword($password),
                'firstname' => $firstname,
                'lastname' => $lastname
            ]
        );
    }
    public static function hashPassword($password) {
        return md5($password);
    }

    public static function isLoginExists($login) {
        $user = \core\Core::getInstance()->db-
>select(self::$tableName, '*', [
            'login' => $login
        ]);
        return !empty($user);
    }
    public static function verifyUser($login, $password) {
        $user = \core\Core::getInstance()->db-
>select(self::$tableName, '*', [
            'login' => $login,
            'password' => $password
        ]);
        return !empty($user);
    }
    public static function getUserByLoginAndPassword($login,
$password) {
        $user = \core\Core::getInstance()->db-
>select(self::$tableName, '*', [
            'login' => $login,
            'password' => self::hashPassword($password)
        ]);
        if(!empty($user))
            return $user[0];
        return null;
    }
    public static function authenticateUser($user) {
        $_SESSION['user'] = $user;
    }
    public static function logoutUser() {
        unset($_SESSION['user']);
    }
    public static function isUserAuthenticated() {
        return isset($_SESSION['user']);
    }
    public static function getCurrentAuthenticatedUser() {
```

```
        return $_SESSION['user'];
    }
    public static  function isAdmin() {
        $user = self::getCurrentAuthenticatedUser();
        return $user['access_level'] == 10;
    }
}
```