ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ...………………………..…….……………..5

ВСТУП………………………………………………………………….………….…6

1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ ОГЛЯД…………………………………...7

1.1 Сучасний стан проблеми………………………………………………………..7

1.2 Аналіз аналогів…………………………………………………………….…….7

1.3 Постанова задачі………………………………………………………………..10

2 ДОСЛІДЖЕННЯ ПРОЦЕСІВ ОБРОБЛЕННЯ ІНФОРМАЦІЇ…………………12

2.1 Контекстна діаграма процесу…………………………………………………..12

2.2 Декомпозиція контекстної діаграми…………………………………………...14

2.3 Діаграма декомпозиції роботи “Відправлення та постачання”………………15

2.4 Побудування ER-діаграми……………………………………………………..17

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ФУНКЦІОНАЛЬНОСТІ…………19

3.1 Актори та прецеденти інформаційної системи………………………………..19

3.2 Потік подій прецеденту інформаційної системи……………………………...21

3.3 Діаграма послідовності подій………………………………………………….22

3.4 Діаграма кооперації об’єктів…………………………………………………..22

3.5 Співвіднесення об’єктів із класами……………………………………….…...23

3.6 Співвіднесення повідомлень із операціями…………………………………..24

3.7 Концептуальна схема даних…………………………………………………...26

3.8 Модель інтерфейсу прикладення………………………………………...……29

4 РЕАЛІЗАЦІЯ ПРИКЛАДЕННЯ……………………………………………...….31

4.1 Проектування архітектури програмного виробу……………………………..31

4.2 Вибір засобів розробки та виконання…………………………………………31

4.3 Створення фізичного проекту даних……………………………….………….33

4.4 Опис складу та модулів програмного виробу……………………….………..36

4.5 Розгортання програмного виробу……………………………………………...39

4.6 Тестування програмного виробу……………………………………………....39

4.7 Визначення апаратних та програмних вимог виробу…………………………40

5 ЕКСПЛУАТАЦІЯ ПРИКЛАДЕННЯ………………………………………...….42

5.1 Розгортання системи…………………………………………………………....42

5.2 Опис функціоналу для покупця…………………………………………….….43

5.3 Опис функціоналу для менеджера…………………………………………….45

6 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ………………………….……48

6.1 Обґрунтування доцільності розробки ПЗ……………………………………..48

6.2 Оцінка конкурентоспроможності ПЗ у порівнянні з аналогом………………48

6.3 Планування комплексу робіт з розробки програмного забезпечення і оцінка трудомісткості робіт………………………………………………………………..51

6.4 Розрахунок проектних витрат на розробку ПЗ……………………………….56

6.5 Розрахунок витрат на впровадження ПЗ………………………………………59

6.6 Розрахунок витрат на закупівлю та впровадження програмного продукту – аналога ПЗ, яке розроблено………………………………………………………...61

6.7 Розрахунок поточних експлуатаційних витрат ПЗ та його аналога…………61

6.8 Розрахунок показника економічного ефекту від розробки ПЗ……………….66

7 ОХОРОНА ПРАЦІ І НАВКОЛИШНЬОГО СЕРЕДОВИЩА………………….69

7.1 Загальні питання охорони праці…………………………………………….…69

7.2 Охорона навколишнього природного середовища…………………………...73

ВИСНОВКИ………………………………………………………………………...75

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ………………………………………………..76

ДОДАТОК А………………………………………………………………………..78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ – Електронно-обчислювальна машина

ДР – Дипломна робота

ОС – Операційна система

ПЗ – Програмне забезпечення

СКБД – Система керування базами даних

CSS – Cascading Style Sheets

EE – Enterprise Edition

HTML – HyperText Markup Language

IDE – Integrated Development Environment

IDEF – Integrated DEFinition

JDBC – Java Database Conectivity

JRE – Java Runtime Environment

JSP – Java Server Pages

JVM – Java Virtual Machine

PHP - Hypertext Preprocessor

POM – Project Object Model

SQL – Structured Query Language

UML - Unified Modeling Language

URL – Uniform Resource Location

ВСТУП

В даний час неймовірно поширеними є веб-сервіси різного напрямку: ігрові сайти, Інтернет магазини, сайти державних установ, соціальні мережі, відео- та аудіо- хостинги, інформаційні портали, веб-системи обліку товарів тощо. З цього переліку розглянемо напрямок інтернет-магазинів, які є дуже поширеними і мають свої напрямки.

Актуальність цього питання полягає в тому, що оскільки Інтернет став ефективним інструментом для здійснення комерційної діяльності, а створення Інтернет магазину – це один з найбільш вигідних і перспективних інструментів онлайн бізнесу

Системи управління веб-додатками надають можливість реалізувати будь-які побажання покупців. А метою створення веб-додатку є збільшення об’єму продажів.

У такому веб-додатку обов’язково повинні бути реалізовані:

1. Споживчий кошик
2. Пошук по сайту
3. Форма реєстрації покупців

1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ ОГЛЯД

**1.1 Сучасний стан проблеми**

За останні 10-15 років веб-сервіси з продажу товарів перетворилися з екзотичного варіанту покупки, у один з основних напрямків збуту товарів масового споживання. На цей процес вплинуло багато факторів, основні з них це розвиток інформаційних технологій, розповсюдження Інтернет мережі, мобільного Інтернету, а також доступність та різноманітність гаджетів, які мають вихід до мережі Інтернет.

Сектор продажу товарів для домашніх тварин серед Інтернет продажів є менш розвинутим у порівнянні з продажем смартфонів чи побутової техніки. Причинами такого стану є, окрім просто більшого ринку електроприладів, мала зацікавленість власників домашніх тварин у більш комплексному підході до раціону свої улюбленців, недостатня варіативність товарів, відсутність продуктів для окремих домашніх тварин в зоомагазинах. Тому було вирішено створити веб-сервіс з продажу товарів для домашніх тварин аби закрити цю прогалину на нашому ринку.

**1.2 Аналіз аналогів**

Для вирішення цього питання було проаналізовано веб-сервіси, які можуть вирішити дану проблему. Для аналізу було вибрано три веб-сервіси zoo-baza.com.ua, zoomapa.com.ua та pethouse.ua.

1. zoo-baza.com.ua

Харківський Інтернет-магазин зоотоварів. В цілому має приємний інтерфейс і не приїдається. Є деякі непогані рішення - виведення мінливого вмісту в певному блоці. Але мається також і великий мінус - це необхідність прокрутки сторінки для пошуку потрібної категорії товарів.

Рішення буде не використовувати даний ресурс як зразок для роботи.

Головна сторінка zoo-baza.com.ua представлена ​​на рисунку 1.1.

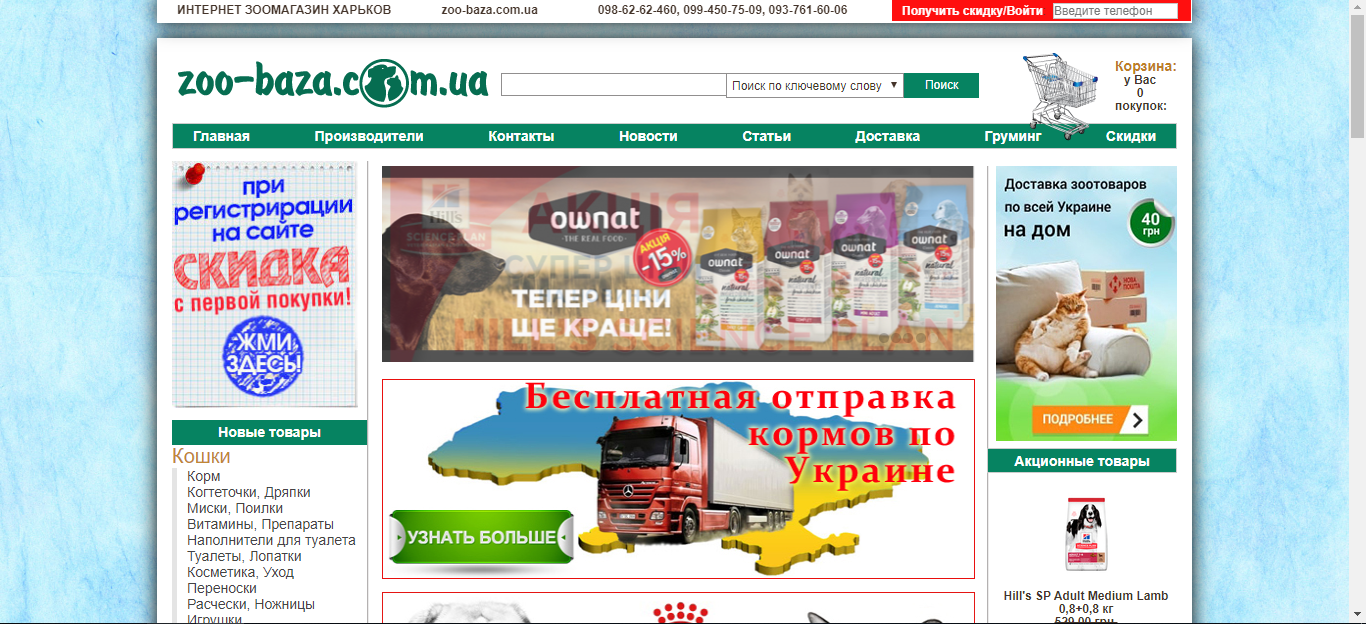


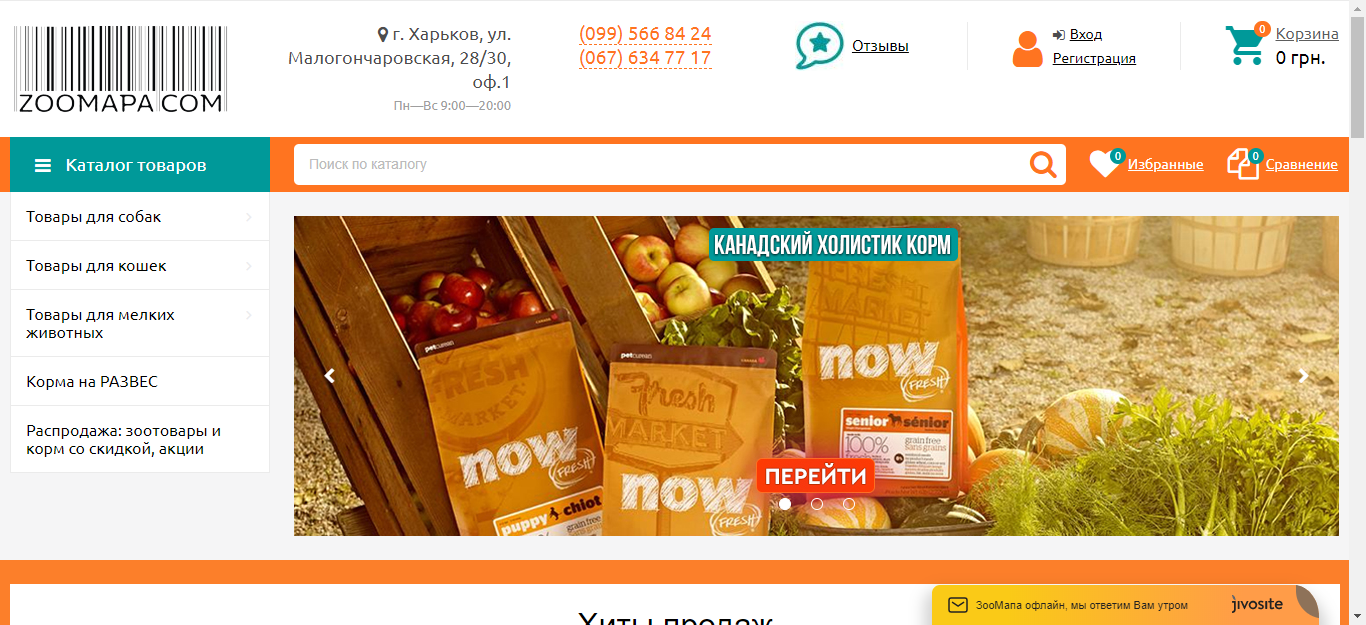
Рисунок 1.1 – Головна сторінка сайту zoo-baza.com.ua

1. zoomapa.com.ua

Харківський Інтернет-магазин зоотоварів. Має приємний інтерфейс, є інтерактивні об'єкти і в цілому зрозумілий інтерфейс. Але є один, але дуже великий мінус - список, що випадає вміст якого можна повністю побачити лише прокрутивши сторінку, що призведе до згортання цього списку.

Рішення буде не використовувати даний ресурс як зразок для роботи.

Головна сторінка zoomapa.com.ua представлена на рисунку 1.2.

Рисунок 1.2 – Головна сторінка zoomapa.com.ua

3. pethouse.ua

Інтернет-магазин зоотоварів. Має приємний зовнішній вигляд, добре пророблений дизайн і хороший розподіл на групи товарів. З мінусів тільки наявність списку, але в цьому випадку працює більш коректно, ніж в прикладі вище.

Рішення буде придивитися до рішень, реалізованих на цьому сайті: кошику, головній сторінці.

Головна сторінка сайту pethouse.ua представлена на рисунку 1.3.

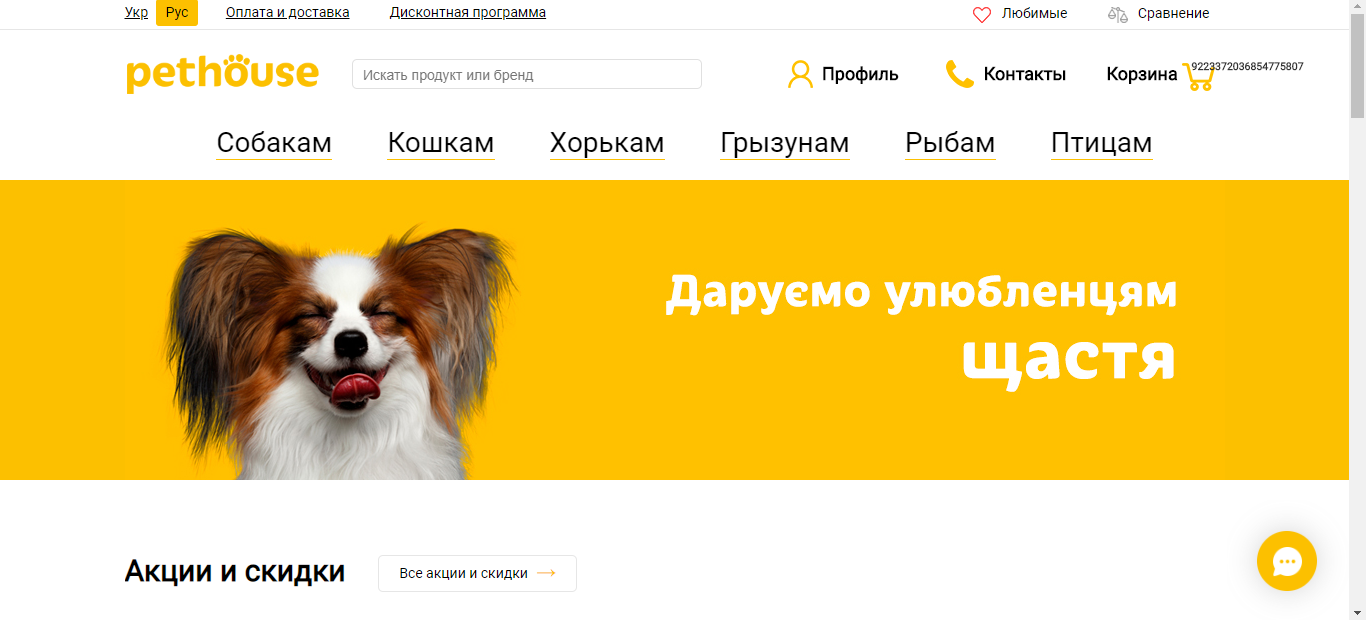


Рисунок 1.3 - Головна сторінка сайту pethouse.ua

Властивості виробів-аналогів наведені в таблиці 1.1.

Таблиця 1.1 – Властивості виробів-аналогів

|  |  |  |  |
| --- | --- | --- | --- |
|  | zoo-baza.com.ua | zoomapa.com.ua | pethouse.ua |
| Сфера застосування | Продаж товарів для домашніх тварин | Продаж товарів для домашніх тварин | Продаж товарів для домашніх тварин |
| Функціональність | Є можливість переглянути, вибрати та замовити товари | Є можливість переглянути, вибрати та замовити товари | Є можливість переглянути, вибрати та замовити товари |
| Методи та засоби | Розроблено на мові PHP(Hypertext Preprocessor) і JavaScript | Розроблено на мові PHP і JavaScript | Розроблено на мові Node.js і JavaScript |
| Апаратні вимоги | Немає | Немає | Немає |
| Вимоги до ПЗ | Підтримка мови HTML5 | Підтримка мови HTML5 | Підтримка мови HTML5 |

**1.3 Постанова задачі**

При аналізі виробів-аналогів були сформулювані конкретні вимоги до роботи, яка має стати результатом ДР, а саме веб-сервіс повинен бути з продажу товарів для домашніх тварин, також в ньому повинно бути реалізовано:

1. зрозумілий і приємний для користувача інтерфейс
2. реєстрація і авторизація користувача
3. можливість пошуку товарів з будь-якої сторінки
4. можливість додання і видалення товару з кошику
5. можливість замовити товар
6. можливість змінити мову
7. можливість перегляду особистих даних
8. можливість вибору товару
9. можливість перегляд інформації про товар
10. можливість для адміністратора змінювати і додавати товар

2 ДОСЛІДЖЕННЯ ПРОЦЕСІВ ОБРОБЛЕННЯ ІНФОРМАЦІЇ

Моделювання діяльності магазину з продажу товарів для домашніх тварин буде здійснюватися за допомогою IDEF (Integrated DEFinition) діаграм.

IDEF діаграма [1] – це методологія для вирішення задач моделювання складних систем, які дозволяють відображати та аналізувати моделі діяльності широкого спектру складних систем.

Суб’єктом моделювання виступає підприємство з продажу товарів для домашніх тварин, а саме процеси, які відбуваються в ньому.

Мета моделювання - відтворити бізнес-процеси, які відбуваються на підприємстві (модель AS-IS).

Точка зору - з позиції директора як особи, яка знає структуру підприємства в цілому.

**2.1 Контекстна діаграма процесу**

Назва процесу – Діяльність інтернет-магазину з продажу товарів для домашніх тварин.

Стрілки входу:

* Замовлення клієнтів – перелік товарів, які клієнт бажає придбати
* Продукція від постачальників - товари, отриманні від постачальників
* Гроші клієнтів – гроші за придбаний товар.

Стрілки виходу:

* Замовлення постачальникам - перелік товарів, які підприємство закуповує у постачальників
* Оплата товарів - гроші постачальникам за отриманий товар
* Маркетингові матеріали - прайс-листівки, реклама на інших сайтах і т.п.
* Відправлення товару до клієнта – товари, які замовив клієнт і є у наявності, відправляють за вказаною адресою

Стрілки управління:

* Законодавство – законодавчі документи, якими керується підприємство під час своєї діяльності
* Правила та процедури - правила та процедури, якими керується підприємство під час своєї діяльності (наприклад, процедура спілкування з клієнтами і т.п.)

Стрілки механізмів:

* Бухгалтерська система
* Персонал

Контекстна діаграма процесу представлена на рисунку 2.1.



Рисунок 2.1 – Контекстна діаграма процесу

**2.2 Декомпозиція контекстної діаграми**

Основний процес складається з:

1. Управління
2. Продажі та маркетинг
3. Відправлення та постачання

1. Робота “Управління” включає в себе загальне управління підприємством тому її входом буде “Гроші клієнтів”, а її результатами буде “Управлінська інформація”, яка поступає на вхід управління всіх інших робіт та “Оплата товарів”.

2. Робота “Продажі та маркетинг” отримує на вхід “Замовлення клієнтів” та “Необхідні товари”, а оброблену інформацію про замовлення “Оформлене замовлення” передасть роботі “Відправлення та постачання”, в якості управлінської інформації, також її результатами буде “Маркетингові матеріали” і “Звіт”.

3. Робота “Відправлення та постачання” отримує на вхід “Продукція від постачальника”, а результатами буде “Замовлення постачальникам”, “Відправлення до клієнтів”, “Необхідні товари” на вхід “Продажі та маркетинг” і “Звіт”.

Декомпозиція контекстної діаграми представлена на рисунку 2.2.

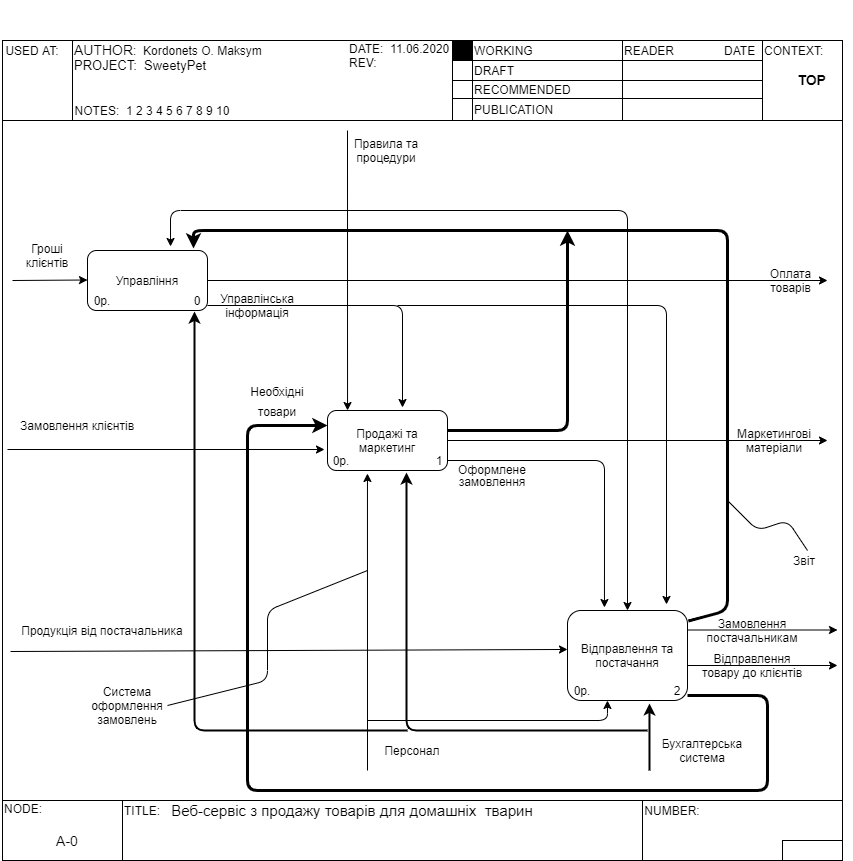


Рисунок 2.2 – Декомпозиція контекстної діаграми

**2.3 Діаграма декомпозиції роботи “Відправлення та постачання”**

Основний процес складається з:

1. Відправлення

2. Постачання

1. Робота “Відправлення” отримує на вхід “Оформлене замовлення” та “Зібрані замовлені товари”, а на вихід подає “Відправлення товару до клієнтів”, “Звіт” та “Замовлені товари”.

2. Робота “Постачання” отримує на вхід “Замовлені товари” та “Продукція від постачальника”, а на вихід подає “Замовлення постачальникам”, “Звіт”, “Необхідні товари” та “Зібрані замовлені товари”.

Діаграма декомпозиції роботи “Відправлення та постачання” представлена на рисунку 2.3.

****

Рисунок 2.3 – Діаграма декомпозиції роботи “Відправлення та постачання”

**2.4 Побудування ER-діаграми**

Основні сутності: Товар, Замовлення, Співробітники, Постачальники, Склад, Покупці.

Інформаційні процеси етапів наведені у таблиці 2.1.

Перелік зв’язків між сутностями наведено у таблиці 2.2.

Таблиця 2.1 – Інформаційні процеси етапів

|  |  |
| --- | --- |
| Етап | Інформаційні процеси |
| 1. Пошук необхідного товару | - Пошук товару на складі за назвою, ціною чи виробником |
| 2. Формування списку товарів | - Виведення обраних товарів; |
| 3. Оформлення документів клієнта | - Збереження інформації у базі даних; |
| 4. Оформлення продажу | - Вибір кількості продаваного товару; |

Таблиця 2.2 – Перелік зв’язків між сутностями

|  |  |
| --- | --- |
| № | Зв’язок |
| 1 | Постачальники ПОСТАЧАЮТЬ Товари |
| 2 | Товари ЗНАХОДЯТЬСЯ Склад |
| 3 | Склад ПРАЦЮЮТЬ Співробітники |
| 4 | Співробітники ОФОРМЛЮЮТЬ Замовлення |
| 5 | Замовлення Роблять Покупці |

ER-діаграма представлена на рисунку 2.4

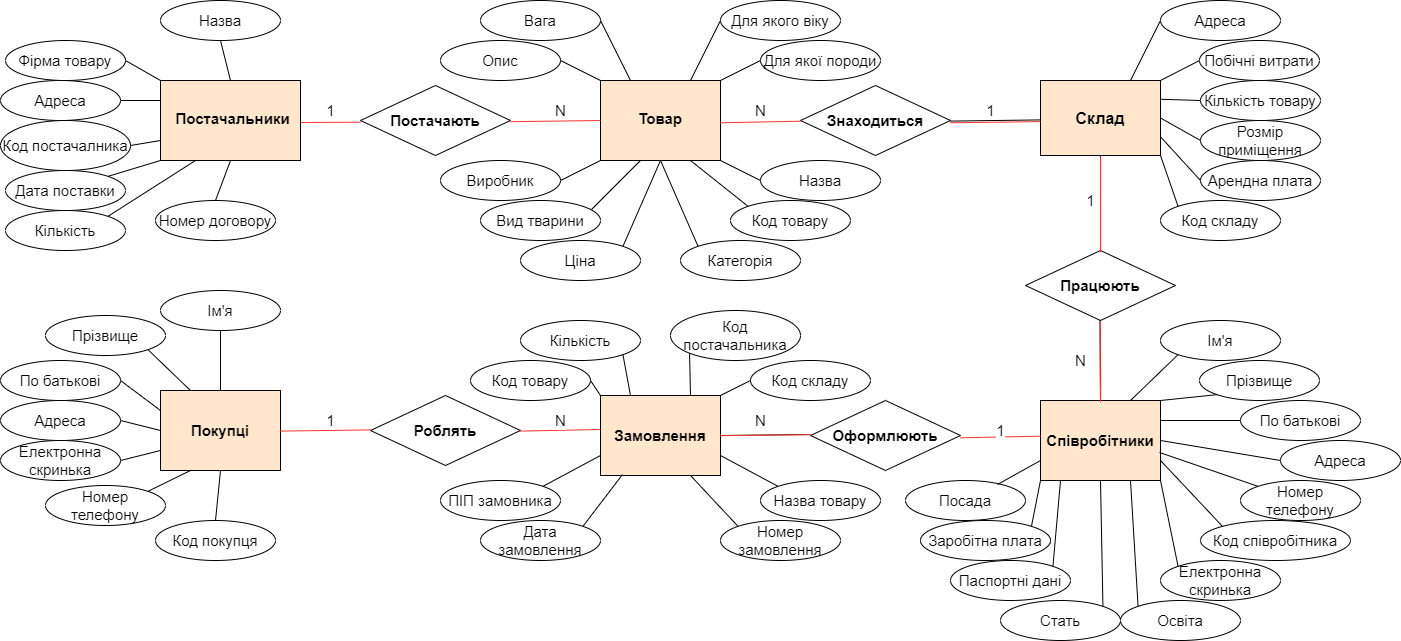


Рисунок 2.4 – ER-діаграма

Типові запити:

1. Вибрати співробітників, заробітна плата яких менше 4700.
2. Вибрати коди складу для товарів в кількості 10 або більше.
3. Вибрати покупця, у якого електронна скринька kordonetsmax@gmail.com.
4. Вибрати назву та кількість товару, який був замовлений до 12.05.2020.
5. Вибрати товар у якого назва виробника “Whiskas” і ціна не більше 200.

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ФУНКЦІОНАЛЬНОСТІ

Моделювання [2] – це спосіб дослідження будь-яких явищ, процесів або об'єктів шляхом побудови й аналізу їх моделей. У широкому розумінні моделювання є однією з основних категорій теорії пізнання і мало не єдиним науково обґрунтованим методом наукових досліджень систем і процесів будь-якої природи в багатьох сферах людської діяльності.

Метою цього розділу є розроблення структури функцій прикладення – автоматизованої системи оброблення інформації. Для моделювання функціональності буде використовуватися мова моделювання UML (Unified Modeling Language).

UML [3] - це мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, для моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

**3.1 Актори та прецеденти інформаційної системи**

Актор (діюча особа) — це зовнішня стосовно ІС сутність, яка може взаємодіяти з системою. Акторами можуть бути люди, зовнішні ІС або пристрої.

В роботі були виділені наступні актори:

1. Покупець

2. Персонал магазину

Прецедент — опис множини послідовностей виконуваних системою дій (транзакцій) з вироблення певного результату, що становить цінність для певного зовнішнього об’єкта (актора).

Прецеденти системи, що проектується:

1. Створення аккаунта

2. Вибір товару з каталогу

3. Зробити замовлення

4. Змінити статус замовлення

5. Скасувати замовлення

6. Переглянути замовлення

Діаграма прецедентів наведена на рисунку 3.1.

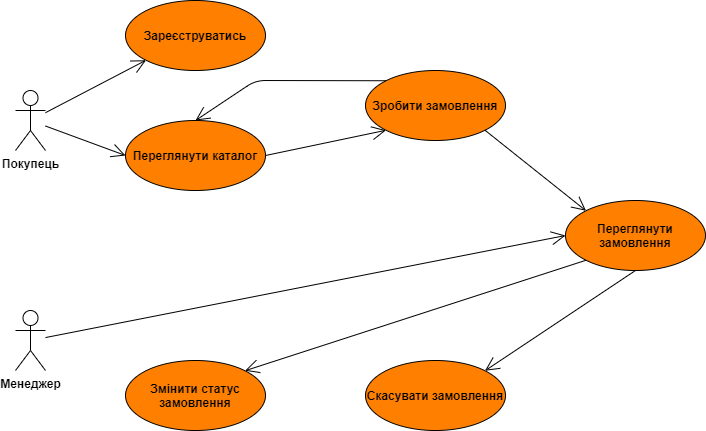


Рисунок 3.1 – Діаграма прецедентів

**3.2 Потік подій прецеденту інформаційної системи**

Між суб’єктами та прецедентами є сталий зв’язок. На рисунку 3.1 можна побачити, що користувач може реєструватись, а може відразу перейти до каталогу, а менеджер, дивлячись на рішення клієнта, або змінює статус замовлення, або його скасовує.

Опис події прецеденту наведено у таблиці 3.1.

Таблиця 3.1. – Опис прецеденту

|  |  |
| --- | --- |
| Прецедент | Зробити замовлення |
| Короткий опис | Покупець дає замовляє обраний товар |
| Суб’єкт | Покупець |
| Передумови | 1. Створення аккаунта 2. Додавання товару до кошика |
| Основний потік | 1. Зайти під свій профіль на сайті 2. Перейти до вкладки кошик и натиснути кнопку “Замовити” 3. Або вибрати кількість якогось одного товару и натиснути кнопку “Замовити товар” |
| Альтернативний потік | Якщо робиться замовлення без реєстрації на сайті, то переходить до вкладки Кошик і тисне кнопку  “Замовити” і вказує необхідні дані. |
| Після умова | Замовлення йде на обробки, клієнт має можливість повернутися до каталоги, або зробити ще замовлення. |

**3.3 Діаграма послідовності подій**

Для того аби зробити замовлення Покупець може зареєструватися, щоб його товари в кошику зберігалися, або розпочати без реєстрації, якщо це одноразове придбання, або просто для перегляду пропозицій. Діаграма послідовності подій наведена на рисунку 3.2.

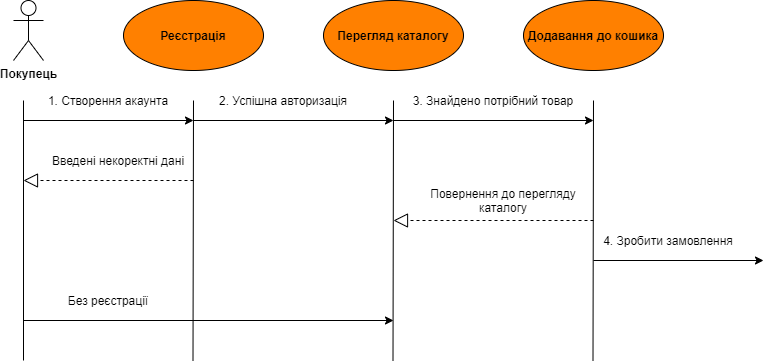


Рисунок 3.2 – Діаграма послідовності подій

**3.4 Діаграма кооперації об’єктів**

Діаграма кооперації об’єктів, яка наведена на рисунку 3.3, зображає тільки відносини між об'єктами, що грають певні ролі у взаємодії. На цій діаграмі не вказується час у вигляді окремого вимірювання. Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, що беруть участь в цій взаємодії.



Рисунок 3.3 – Діаграма кооперації об’єктів

**3.5 Співвіднесення об'єктів із класами**

Розглянемо процес створення аккаунту. Для цього покупцю потрібно завітати на головну сторінку сайту, перейти у розділ профілю і натиснути “зареєструватися” та перейти за посиланням. Після чого ввести необхідні дані, натиснути “Зареєструватися” і перейти до сторінки вибору товарів.

Діаграма послідовності з класами наведена на рисунку 3.4.



Рисунок 3.4 – Діаграма послідовності з класами

**3.6 Співвіднесення повідомлень із операціями**

Для оформлення замовлення потрібно вибрати товар, занести його у кошик і натиснути кнопку “Замовити”. Після цього менеджер отримує замовлення і перевіряє товар на наявність, якщо товар в наявності, тоді дозволити проведення операції, якщо немає, тоді Скасувати і повідомити про це Покупця.

Діаграма послідовності з класами та операціями наведена на рисунку 3.5

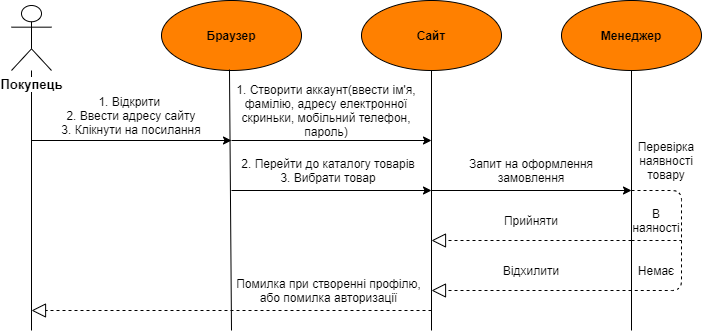


Рисунок 3.5 – Діаграма послідовності з класами та операціями

* 1. **Концептуальна схема даних**

Опис схеми даних представлена на рисунку 3.6.

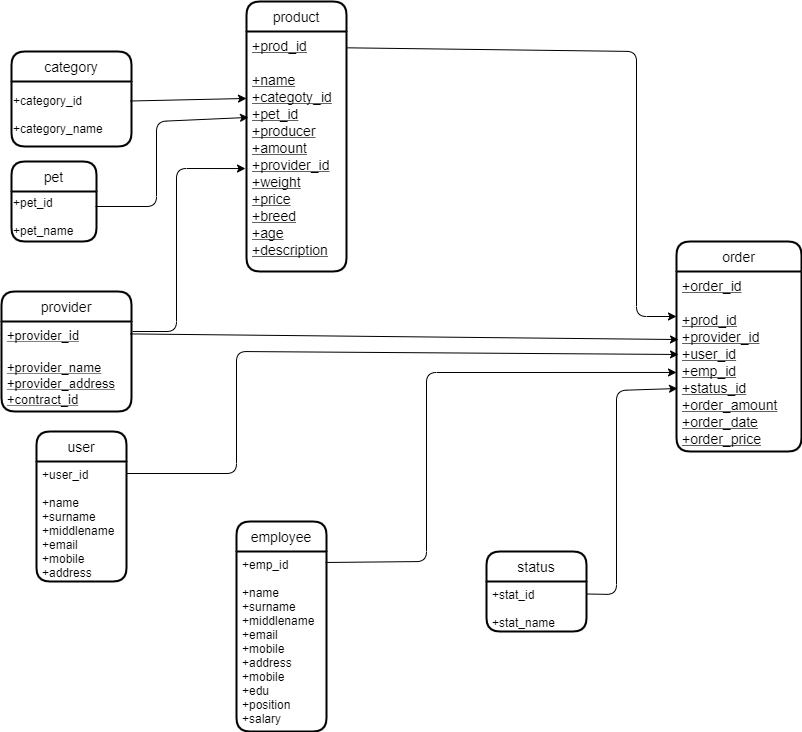


Рисунок 3.6 – Концептуальна схема даних

Таблиця user відповідає сутності “Покупець”. Її атрибути:

* user\_id – ідентифікаційний код покупця, первинний ключ, тип int;
* name – ім’я покупця, тип text;
* surname – прізвище покупця, тип text;
* middlename – по батькові покупця, тип text;
* email – адреса поштової скриньки покупця, тип text;
* mobile – номер мобільного телефону покупця, тип text;
* address – поштова адреса покупця, тип text.

Таблиця product відповідає сутності “Товар”. Її атрибути:

* prod\_id – код товару, первинний ключ, тип int;
* name – назва товару, тип text;
* category\_id – код категорії товару, тип int;
* pet\_id – код виду тварини, для якої призначений товар, тип int;
* producer – назва виробника, тип text;
* amount – кількість товару, тип int;
* provider\_id – код постачальника товару, тип int;
* weight – вага товару, тип int;
* price – ціна товару, тип int;
* breed – порода, для якої призначено товар, тип text;
* age – вік, для якого призначено товар, тип text;
* description – опис товару, тип text;

Таблиця employee відповідає сутності “Співробітник”. Її атрибути:

* emp\_id – ідентифікаційний код співробітника, первинний ключ, тип int;
* name – ім’я співробітника, тип text;
* surname – прізвище співробітника, тип text;
* middlename – по батькові співробітника, тип text;
* email – адреса поштової скриньки співробітника, тип text;
* mobile – номер мобільного телефону співробітника, тип text;
* address – поштова адреса співробітника, тип text.
* edu – рівень освіти співробітника, тип text;
* position – посада співробітника, тип text;
* salary – заробітна плата співробітника, тип int;

Таблиця order відповідає сутності “Замовлення”. Її атрибути:

* order\_id – код замовлення, первинний ключ, тип int;
* prod\_id – код товару, тип int;
* provider\_id – код постачальника товару, тип int;
* user\_id – ідентифікаційний код покупця, тип int;
* emp\_id – ідентифікаційний код співробітника, тип int;
* status\_id – код статусу, тип int;
* order\_amount – кількість товару в замовлені, тип int;
* order\_date – дата замовлення, тип date;
* order\_price – ціна замовлення, тип int;

Таблиця provider відповідає сутності “Постачальник”. Її атрибути:

* provider\_id – код постачальника, первинний ключ, тип int;
* provider\_name – назва постачальника, тип text;
* provider\_address – адреса постачальника, тип text;
* contract\_id – код договору, тип int;

Таблиця category відповідає сутності “Категорії”. Її атрибути:

* category\_id – ідентифікаційний код категорії, первинний ключ, тип int;
* category\_name – назва категорії, тип text (допустимі 4 значення: main food, medical, goodies, canned food, кожне значення відповідає цифрі 0,1,2,3 відповідно);

Таблиця status відповідає сутності “Статус”. Її атрибути:

* stat\_id – ідентифікаційний код статусу, первинний ключ, тип int;
* stat\_name – назва статусу, тип text (допустимі 3 значення: accepted, denied, completed, кожне значення відповідає цифрі 0,1,2 відповідно);

Таблиця pet відповідає сутності “Категорії”. Її атрибути:

* pet\_id – ідентифікаційний код категорії, первинний ключ, тип int;
* pet\_name – назва категорії, тип text (допустимі 5 значень: dog, cat, rodent, bird, fish, кожне значення відповідає цифрі 0,1,2,3,4 відповідно);
  1. **Модель інтерфейсу прикладення**

Складемо діаграму станів для класу Замовлення, графічне представлення діаграми на рисунку 3.7. Замовлення може знаходитися в декількох станах:

* при створені замовлення воно переходить до стану Ініціалізація*,* в якому виконуються збереження дати створення замовлення та внесення інформації про клієнта, товар, співробітника і постачальника;
* після завершення ініціалізації замовлення переходить до стану Очікування підтвердження;
* якщо замовлення підтверджено, переходить до стану Закрито, в якому виписується рахунок;
* якщо замовлення скасовано, тоді зі стану Очікування підтвердження переходить до стану Скасовано, в якому буде записано дату скасування замовлення;

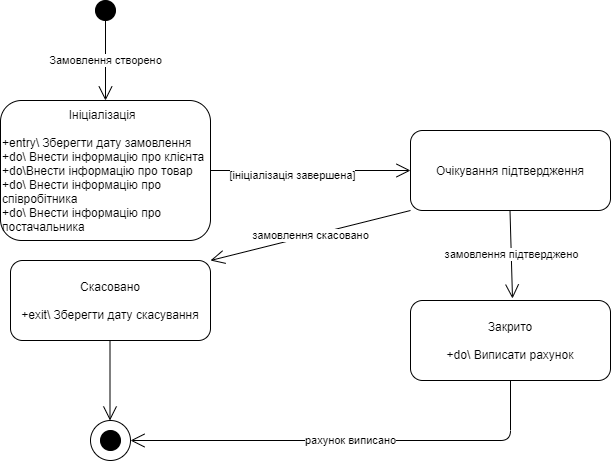


Рисунок 3.7 – Діаграма станів

4 РЕАЛІЗАЦІЯ ПРИКЛАДЕННЯ

**4.1 Проектування архітектури програмного виробу**

Для виконання поставленого завдання, а саме створення веб-сервісу з продажу товарів для домашніх тварин, була обрана архітектура веб-додатку.

Веб-додаток [4] – розподілений застосунок, в якому клієнтом виступає браузер, а сервером – веб-сервер.

Тобто логіка розподілена між сервером і клієнтом, зберігання даних здійснюється на сервері, обмін інформацією відбувається по мережі. До переваг треба віднести те, що клієнти не залежать від конкретної ОС (Операційна система).

**4.2 Вибір засобів розробки та виконання**

При виборі мови програмування було прийнято рішення вибрати Java 8.0 [5] оскільки це об’єктно-орієнтована мова програмування, в якій Java-код компілюється у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи, тобто ця мова програмування є кроссплатформенною. А також Java, а точніше її специфікація Jakarta EE (Enterprise Environment) надає API та виконавче середовище для розробки і виконання у тому числі і веб-сервісів.

СКБД (Система керування базою даних) [6] – це совокупність програмних та лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням і використанням баз даних.

Найрозповсюдженою СКБД є реляційна СКБД (РСКБД), особливістю якої є простота структури даних і табличне представлення, де кожна така таблиця є двовимірним масивом.

Переглядаючи можливі РСКБД вибір зупинився на MySQL 5.7.10 [7]. Вибір було зумовлено тим, що ця РСКБД є безкоштовною, має докладний та інтуїтивно зрозумілий інтерфейс користувача і насамперед є досить популярною та здебільшого використовується для створення динамічних веб-сторінок.

Вибираючи інтегроване середовище розробки вибір пав на Eclipse IDE (Integrated Development Environment) for Java EE Developers [8], оскільки це безкоштовний додаток з дуже приємним і зрозумілим інтерфейсом, також ця версія підтримує мови Java 8, HTML(HyperText Markup Language) 5, SQL (Structured Query Language):2016.

Сервером прикладення було обрано Apache Tomcat 7.2.2 [9] , оскільки цей контейнер сервлетів, повністю написаний мовою програмування Java і реалізує JSP та сервлети, на яких буде проводитися розробка веб-застосунку.

Також у роботі буде використовуватися стандартизована мова розмітки документів у Всесвітній павутині – HTML 5 [10]. Оскільки більшість веб-сторінок містять опис розмітки на цій мові.

Для опису зовнішнього вигляду сторінок, написаних на мові розмітки HTML 5, була обрана спеціальна мова стилю сторінок CSS (Cascading Style Sheets). Оскільки вона є основної технологією Всесвітньої павутини.

Для здійснення доступу до бази даних був використаний прикладний програмний інтерфейс Java JDBC (Java DataBase Connectivity) [11]. Перевагами є можливість під’єднатись до бази даних через легко описуваний URL (Uniform Resource Locator), легкість розробки, відсутність потреби встановлювати клієнтські програми.

Технологією динамічного генерування HTML сторінок було обрана JSP (JavaServer Pages) [12], оскільки це високопродуктивна технологія, де код сторінки транслюється в java-код сервлету за допомогою компілятора, після чого компілюється в байт-код JVM (Java Virtual Machine).

Для забезпечення автоматизованої зборки проектів було обрано фреймворк Apache Maven [13], який збирає проект на основі опису його структури в файлах на мові POM (Project Object Model).

Також в ході цієї роботи було використано MySQL Embedded Server, який є розрахованою на одного користувача локальною базою даних, яка повністю сумісна с MySQL Server. До того ж не потребує інсталяції, налаштування і може бути запускатися зі змінних накопичувачів.

**4.3 Створення фізичного проекту даних**

SQL-скрипт для створення таблиць:

set names utf8;

drop database if exists petshop;

create database petshop character set utf8;

use petshop;

create table roles (

id integer not null primary key,

role\_name varchar(15) not null unique);

create table users(

id integer not null primary key auto\_increment,

name varchar(20) not null,

surname varchar(20) not null,

pass varchar(30) not null,

phone varchar(20) not null ,

email varchar(30) not null unique,

role\_id integer not null default 0,

foreign key (role\_id) references roles(id)

on delete cascade

on update restrict);

create table pets(

id integer not null primary key,

name varchar(15) not null unique);

create table type\_product(

id integer not null primary key,

name varchar(20) not null);

create table products (

id integer not null primary key auto\_increment,

name varchar(50) not null unique,

price integer not null,

description text not null,

amount integer not null,

weight integer not null,

producer varchar(30) not null,

type\_id integer not null,

age varchar(20) default null,

breed varchar(50) default null,

pet\_id integer not null,

photo\_link text not null,

foreign key (pet\_id) references pets(id)

on delete cascade

on update restrict,

foreign key (type\_id) references type\_product(id)

on delete cascade

on update restrict);

create table orders(

id integer not null primary key auto\_increment,

user\_id integer not null,

product\_id integer not null,

count\_product integer not null default 1,

foreign key (user\_id) references users(id)

on delete cascade

on update restrict,

foreign key (product\_id) references products(id)

on delete cascade

on update restrict);

SQL-скрипти для типових запитів

1. Вибрати всіх користувачів

SELECT \* FROM users

1. Вибрати покупця, у якого електронна скринька kordonetsmax@gmail.com.

SELECT \* FROM users WHERE email = ‘kordonetsmax@gmail.com’

1. Вибрати товари, назва яких “Природа Колосок” і ціна менше 200

SELECT \* FROM products WHERE name = “Природа Колосок” AND price < 200

1. Вибрати товари, у яких номер виду тварин дорівнює 2

SELECT \* FROM products WHERE pet\_id = 2

5. Вибрати товари, у якого назва виробника “Whiskas” і ціна не більше 200.

SELECT \* FROM products WHERE producer = ‘Whiskas’ AND price < 200

**4.4 Опис складу та модулів програмного виробу**

Компонентна діаграма серверної частини додатку представлена на рисунку 4.1.

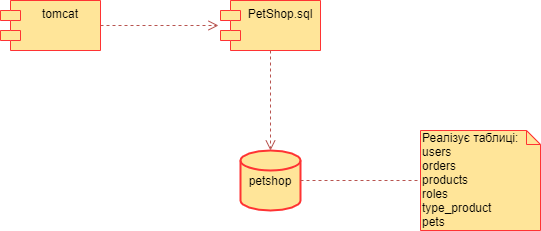


Рисунок 4.1 – Компонентна діаграма серверної частини додатку

Опис модулів серверної частини:

1. PetShop.sql – скрипт, який створює базу даних та її таблиці, заповнює таблиці контентом.
2. petshop– база даних, яка реалізовує таблиці та зберігає інформацію.

Компонентна діаграма серверної частини додатку, відповідальної за візуалізацію HTML сторінок представлена на рисунку 4.2.

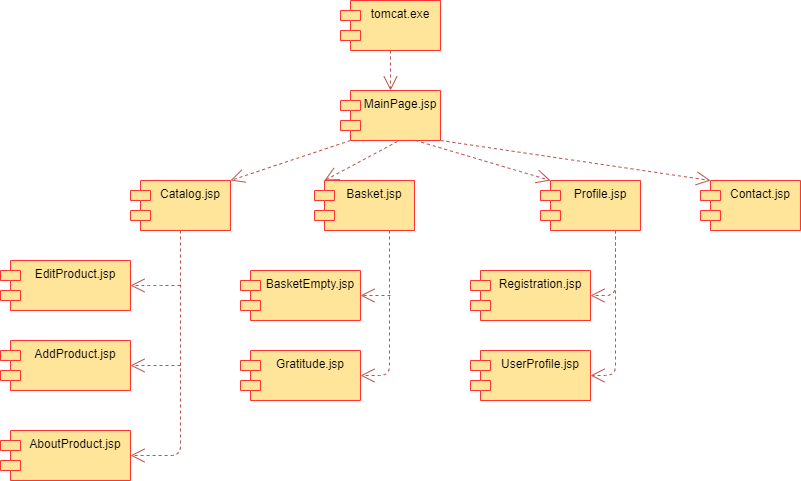


Рисунок 4.2 - Компонентна діаграма серверної частини додатку, відповідальної за візуалізацію HTML сторінок

Опис модулів серверної частини відповідальної за візуалізацію HTML сторінок:

1. MainPage.jsp – Головна сторінка додатку, з якої можливо перейти до інших сторінок. Вибравши потрібний вид тварини і категорію товару перейти на сторінку каталогу.
2. Catalog.jsp – Сторінка з переліком товарів, з якої можливо додати товар до кошика, а для адміністратора ще є можливість додати чи змінити будь-який товар.
3. Basket.jsp – Сторінка, яка відображає товар, який додав до кошика покупець і надає можливість замовити товари.
4. Profile.jsp – Сторінка, де користувачу необхідно авторизуватись.
5. Contact.jsp – Сторінка контактів магазину.
6. EditProduct.jsp – Сторінка, на якій адміністратор може змінити значення атрибутів товару.
7. AddProduct.jsp – Сторінка, на якій адміністратор може додати новий товар.
8. AboutProduct.jsp – Сторінка детального описання товару.
9. BasketEmpty.jsp – Пуста сторінка кошику.
10. Gratitude.jsp – Сторінка вітання з успішно оформленим замовленням.
11. Registation.jsp – Сторінка реєстрації користувача.
12. UserProfile.jsp – Сторінка профілю користувача, де вказуються його дані.

**4.5 Розгортання програмного виробу**

Діаграма розгортання представлена на рисунку 4.3.

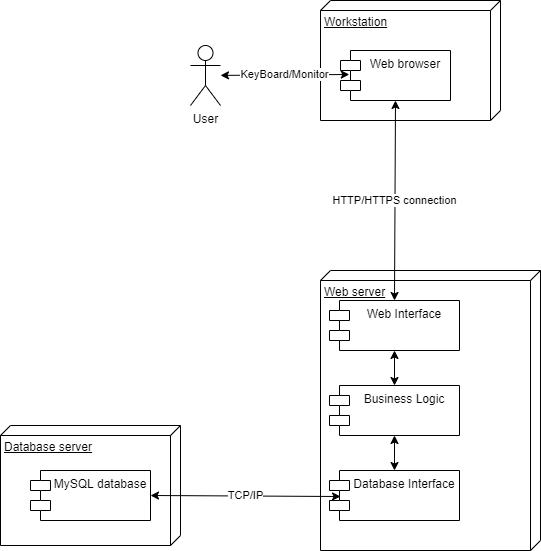


Рисунок 4.3 – Діаграма розгортання

**4.6 Тестування програмного виробу**

Для тестування програмного виробу я обрав ручне тестування, де я відіграв роль користувача для знаходження помилок у роботі програми.

Результати тестування представлені в таблиці 4.1.

Таблиця 4.1 – Результати тестування

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування | Очікуваний результат | Результат перевірки | Помилки |
| Функціональне тестування | Всі кнопки працюють, читабельні та відповідають заявленому функціоналу | Всі кнопки працюють коректно  Всі запити обробляються коректно  Всі посилання працюють коректно  Всі дані з бази даних відображаються коректно | Немає |
| Всі запити, які посилаються до бази даних мають бути правильно оброблені. Дані, які повині бути занесені в базу даних, мають там бути | Немає |
| Всі посилання працюють, читабельні та відповідають заявленому функціоналу | Немає |
| Всі дані, які мають бути взяті з бази даних та представлені на сторінка повинні відображатися коректно. | Немає |
| Тестування верстки | Багато сторінковий сайт | Сайт багатосторінковий  Відправка форми здійснюється тільки якщо заповнені всі поля  Поля провалідовані | Немає |
| При заповнені форм відправка до серверу має відбуватися лише якщо були заповнені всі поля | Немає |
| Всі поля мають бути провалідовані | Немає |

**4.7 Визначення апаратних та програмних вимог виробу**

Завдяки тому, що представлений програмний виріб це веб-додаток, він може використовуватися на будь-якій ОС та на будь-якій ЕОМ (Електронно- обчислювальна машина), яка підтримує версії браузерів наведені у таблиці 4.2.

Таблиця 4.2 – Підтримка браузерами мови HTML5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Браузер | Internet Explorer | Firefox | Chrome | Safari | Opera | Safari iOS |
| Мінімальна версія | 8 | 3.5 | 5 | 4 | 10.5 | 2 |

Для розгортання серверної частини потрібно встановити необхідне ПЗ, а саме Java JRE (Java Runtime Environment) 8, Apache Tomcat 7.2.2 і MySQL 5.7.10. Тому для коректної роботи додатку необхідно забезпечити коректну роботу цього ПЗ.

Сервер повинен мати такі характеристики:

* Оперативна пам’ять: 2 ГБ
* Дисковий простір: 1 ГБ
* Процесор: мінімальні вимоги – Pentium 2266 МГц
* ОС: Windows, macOS або Linux(наприклад, Debian, Ubuntu)

5 ЕКСПЛУАТАЦІЯ ПРИКЛАДЕННЯ

**5.1 Розгортання системи**

Для розгортання веб-додатку попередньо треба встановити Java 8, Apache Maven 3 та MySQL Server.

Далі потрібно відкрити консоль операційної системи, вибрати директорію, куди буде клоновано проект і прописати:

git clone <https://github.com/Maksym1996/PetShop.git>

Потім потрібно зібрати проект, для чого потрібно прописати:

mvn clean install

І на останок, запустити додаток виконавши команду, як зображено на рисунку 5.1 і 5.2:

mvn tomcat7:run

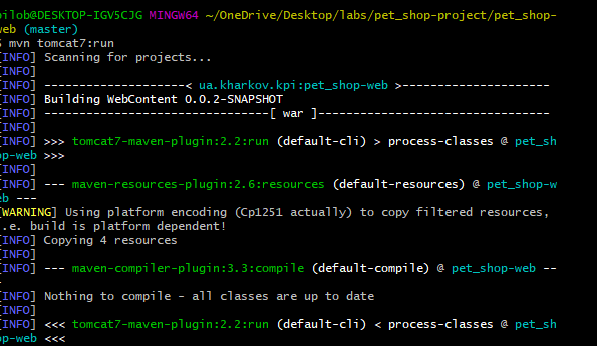


Рисунок 5.1

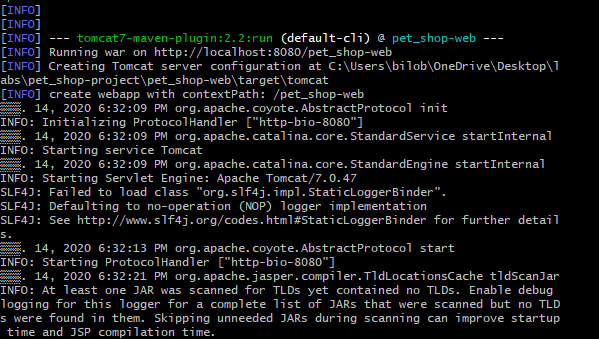


Рисунок 5.2

**5.2 Опис функціоналу для покупця**

1. Завітати до головної сторінки рисунок 5.3.

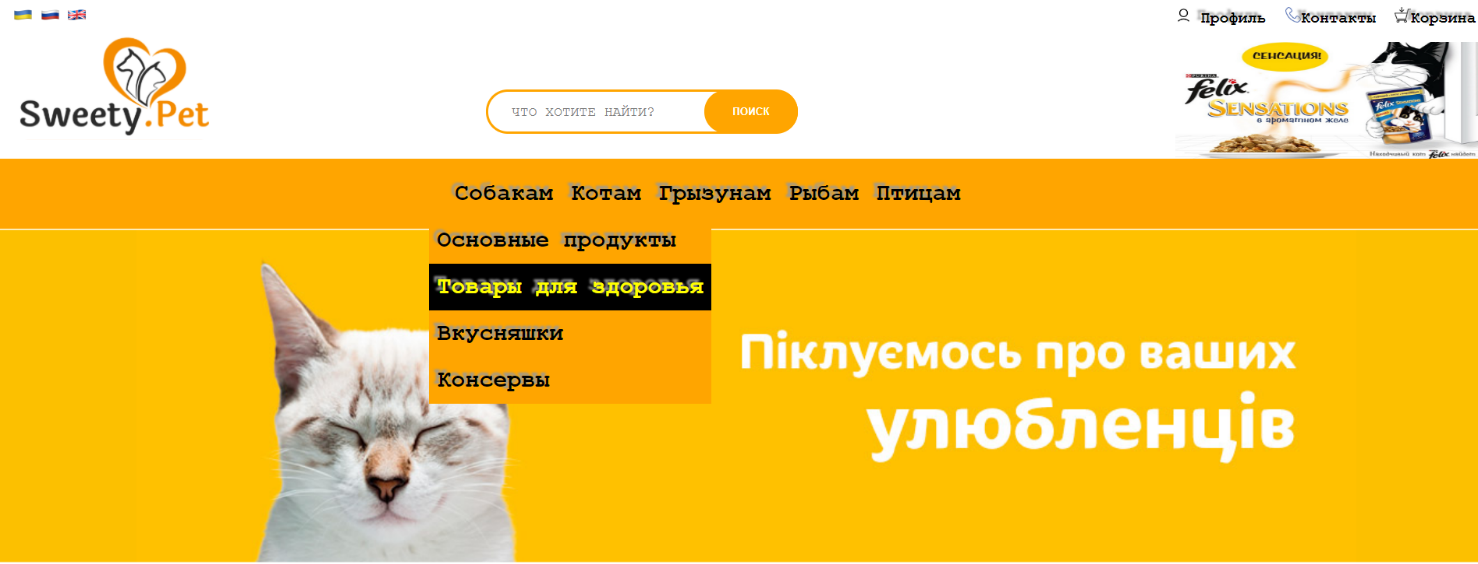


Рисунок 5.3 – Головна сторінка сайту

1. Вибрати вид тварини і категорію товару
2. Переглянути перелік товарів і продивитись інформацію про обраний товар, як це показано на рисунку 5.4.

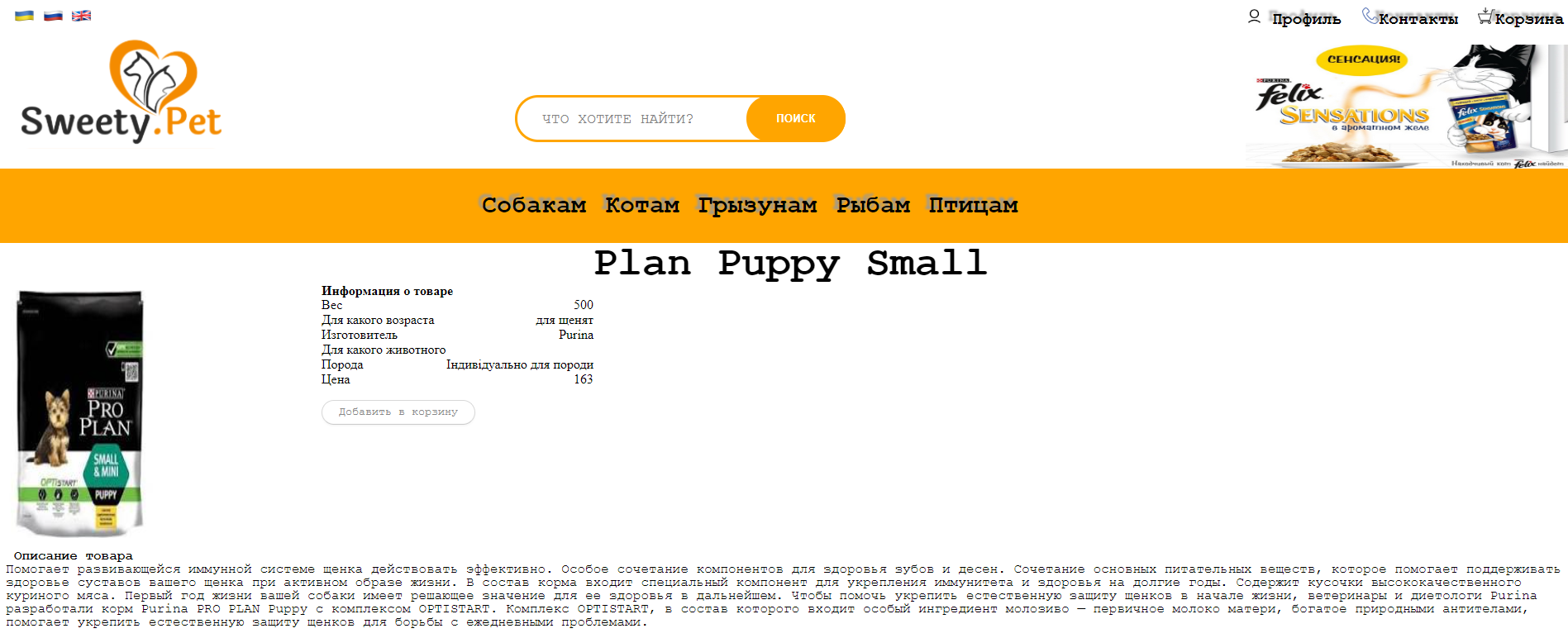


Рисунок 5.4

1. Натиснути додати у кошик
2. При бажані повторити операцію
3. Обравши необхідні товари, перейти до вкладки Кошик у верхньому правому куті.
4. Але перед тим нам запропонують авторизуватись, або якщо ви не зареєстровані – зареєструватись, як показано на рисунку 5.5.



Рисунок 5.5 – Сторінка авторизації

1. Після авторизації нас переносять до сторінки профілю, де можна побачити свої дані
2. Знову переходимо до кошика і бачимо весь товар, який ми додали. Можемо вибрати кількість кожного товару і, або замовити один вид товару натиснувши кнопку Замовити зліва від товару, або замовити весь товар, натиснувши кнопку Замовити все, як показано на рисунку 5.6.

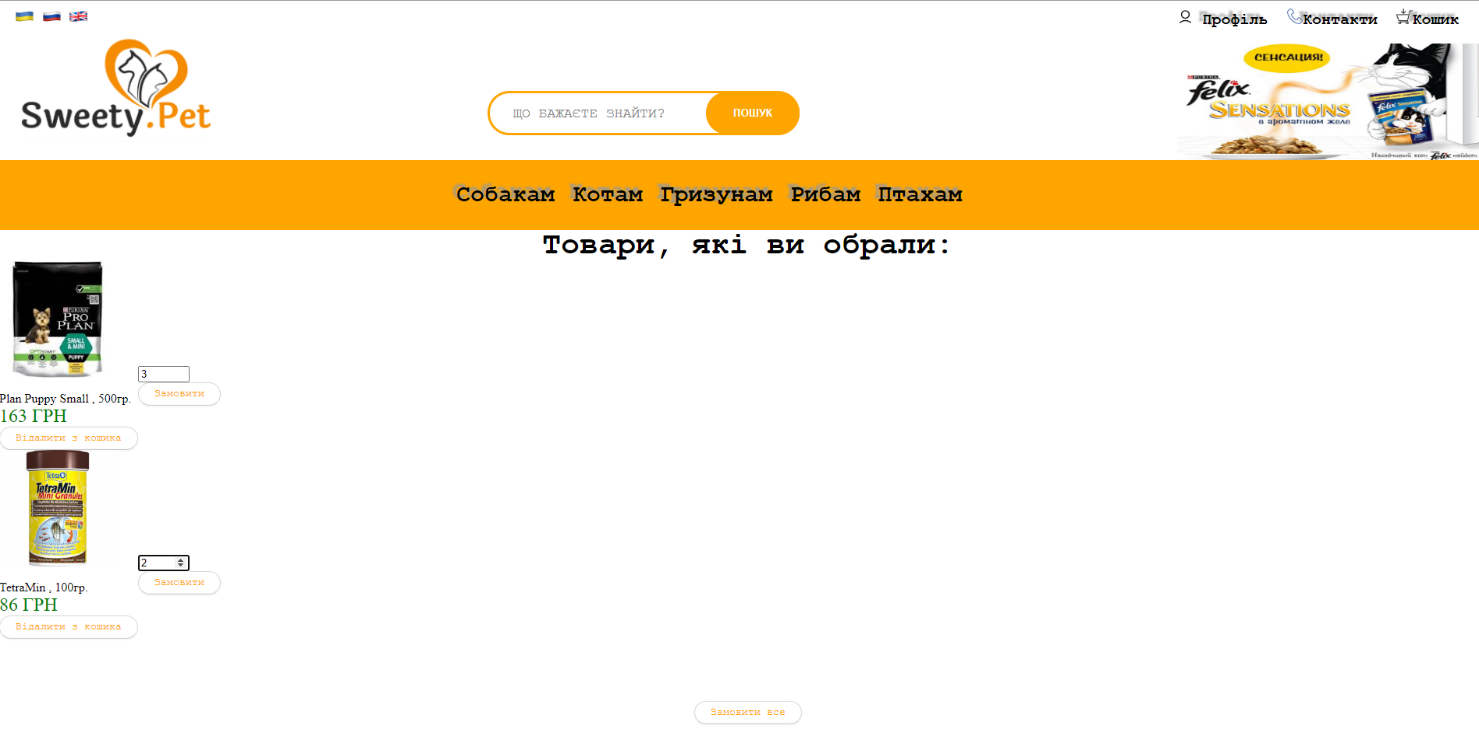


Рисунок 5.6 – Сторінка замовлення товару

1. Є можливість обрати мову інтерфейсу, вибравши відповідний прапор у верхньому лівому куті.
   1. **Опис функціоналу для менеджера**
2. Завітати до головної сторінки сайту
3. Перейти до профілю і авторизуватись
4. Далі вибрати вид тварини і категорію товару
5. На сторінці перегляду товарів менеджер може або продивитись відомості про товар, або додати товар, або змінити його, як показано на рисунку 5.7.

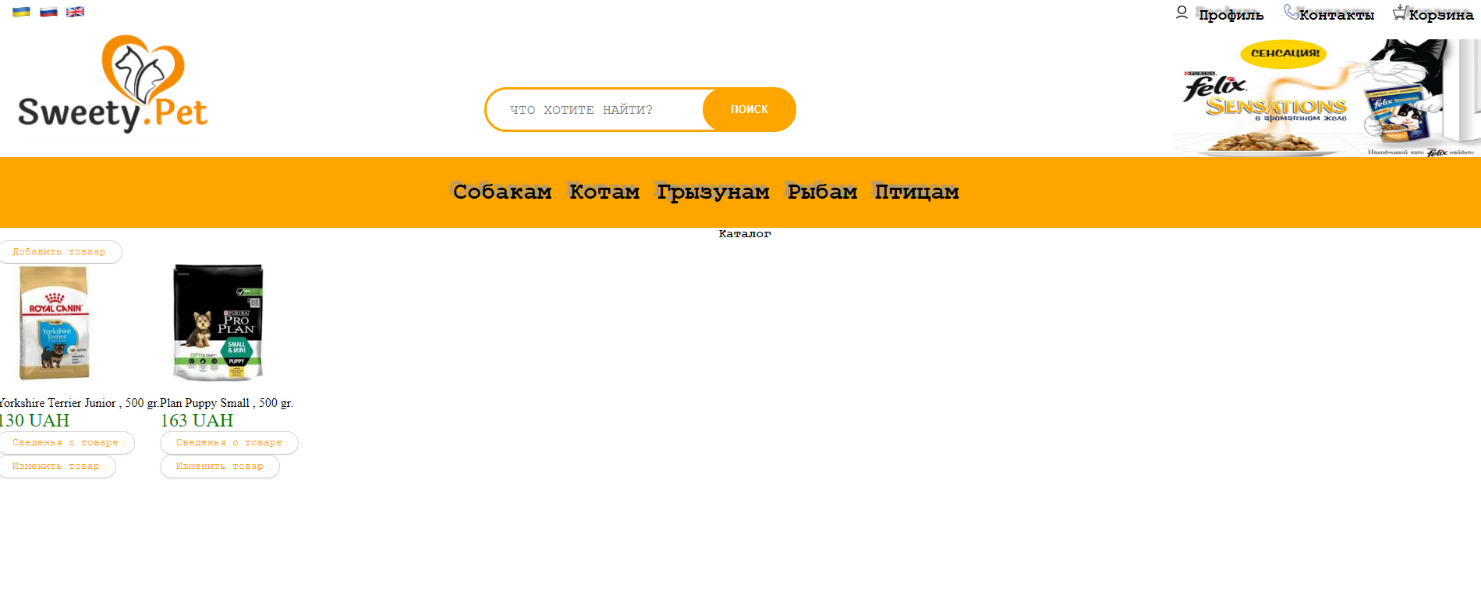


Рисунок 5.7 – Сторінка перегляду товару на стороні менеджера

1. Сторінка зміни товару зображена на рисунку 5.8.



Рисунок 5.8 – Сторінка зміни товару

1. Сторінка додавання товару зображена на рисунку 5.9, на якій необхідно заповнити всі поля для створення товару

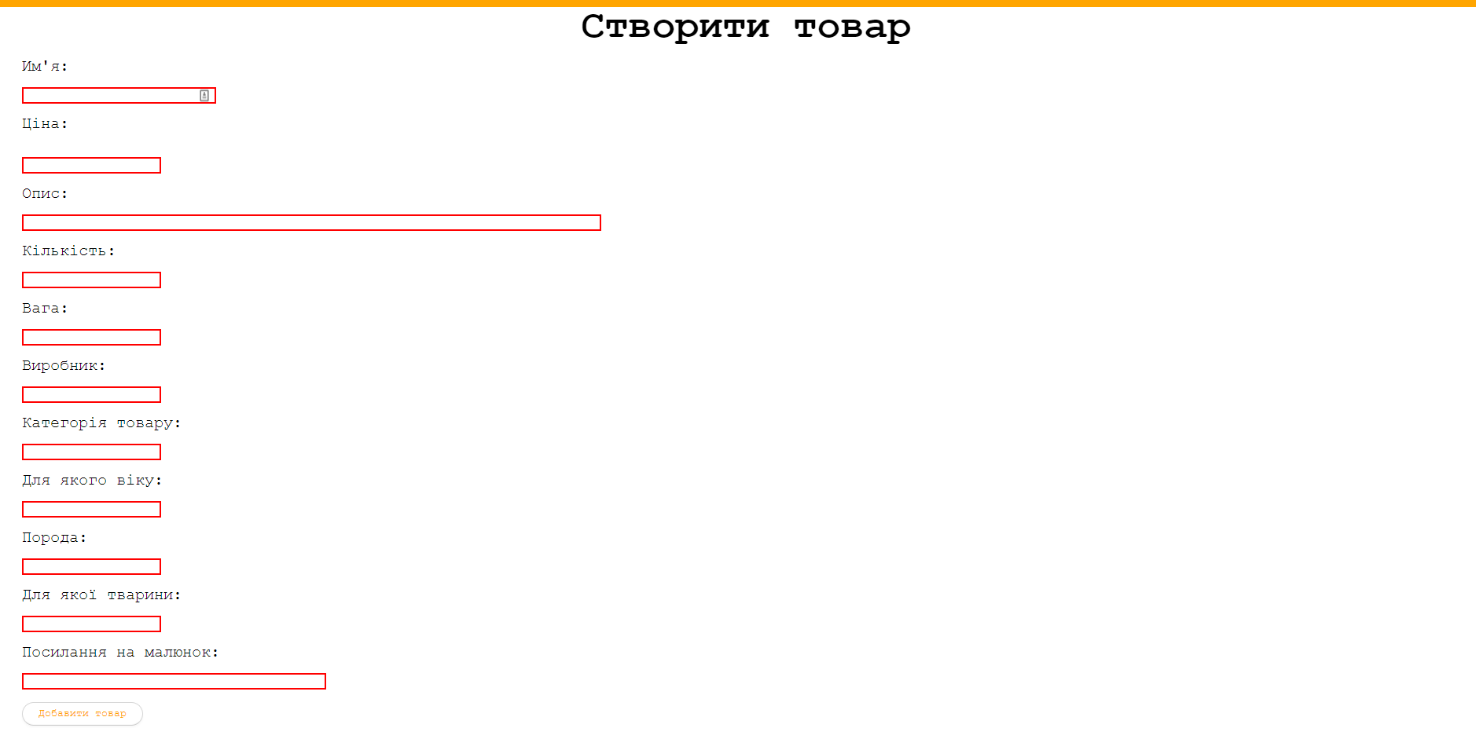


Рисунок 5.9 – Сторінка створення товару

6 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ ЩОДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**6.1 Обґрунтування доцільності розробки ПЗ**

У даному розділі представлено економічне обґрунтування розробки веб-сервісу з продажу товарі для домашніх тварин “PetShop”. Сьогоднішній ринок досить широкий асортимент різної продукції для домашніх тварин, а з використанням Інтернет магазину можливо збільшити продажі та стати більш відомим магазином з продажу товарів для домашніх тварин.

Основними функціями такого модуля є:

– представлення всього асортименту товарів поділених за видом тварин та категоріями;

– описання кожного товару та його характеристики;

– можливість замовлення потрібних товарів для клієнтів без необхідності завітати до магазину.

Застосування даного ПЗ дозволить підвищити ефективність роботи за рахунок скорочення витрат часу на замовлення для клієнтів за задовільну ціну, не виходячи з дому, або навпаки в перервах між справами.

**6.2 Оцінка конкурентоспроможності ПЗ у порівнянні з аналогом**

Для визначення доцільності розробки ПЗ, оцінки його економічної ефективності як його аналог був обраних Інтернет магазин з продажу зоотоварів zoo-bazacom.ua. Це було зумовлено такими чинниками:

– суміжний профіль – продаж зоотоварів для домашніх тварин;

– один регіон(місто Харків);

Конкурентоспроможність програмного продукту визначається властивостями його як товару на ринку. Ці властивості характеризують ступінь задоволення конкретної потреби користувачів у порівнянні з кращими аналогічними продуктами , представленими на ІТ- ринку. Перш ніж здійснювати вкладення фінансових коштів у розробку програмного продукту, розробникам необхідно зробити попередню оцінку здатності його конкурувати з продуктами-аналогами.

Для оцінки конкурентоспроможності програмного продукту, що розробляється, необхідно провести аналіз і порівняння з обраним аналогом за функціональним призначенням, основними технічними та експлуатаційними параметрами, областям використання. Подібний аналіз здійснюється за допомогою оцінки експлуатаційно-технічного рівня ПЗ, що розроблено.

Експлуатаційно-технічний рівень (ЕТУ) програмного забезпечення, що розроблено, є узагальненою характеристикою його експлуатаційних властивостей і можливостей, ступеня новизни, а це , в свою чергу, є основою визначення якості ПЗ. Для визначення ЕТУ часто використовують індекс (𝐽ЕТР) експлуатаційно-технічного рівня, який розраховується як сума приватних індексів, куди входять показники якості програмного продукту.

Комплексний показник 𝐽ЕТР якості продукту розраховується так:

𝐽ЕТР=Σ𝐵𝑗𝑛𝑗=1∙𝑋𝑗 (2.1),

де 𝐵𝑗 – коефіцієнт вагомості *j*-го показника в долях одиниці, який призначається відповідно до потреб організації-замовника програмного продукту;

𝑋𝑗 – експертна оцінка *j*-го показника якості за обраною шкалою оцінювання;

*n* – число показників.

Показники якості вибираються відповідно з деревом характеристик якості програмного виробу (ISO 8402: 94, IEEE Std 610.12-1990).

Для визначення значущості окремих параметрів застосовується бально-індексний метод. У таблиці 2.1 представлені результати розрахунку 𝐽ЕТР бально-індексним методом за п’ятибальною шкалою оцінювання.

Таблиця 6.1 – Розрахунок показника якості 𝐽ЕТР

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Показники якості програмного продукту | Коефіцієнт вагомості, 𝐵𝑗 | Проект | | Аналог | |  |
| 𝑋𝑗 | 𝑋𝑗∙𝐵𝑗 | 𝑋𝑗 | 𝑋𝑗∙𝐵𝑗 |  |
| 1. Зручність роботи (призначений для користувача інтерфейс) | 0,14 | 4 | 0,56 | 2 | 0,28 |  |
| 2.Новизна (відповідність сучасним вимогам) | 0,1 | 3 | 0,3 | 3 | 0,3 | |
| 3.Відповідність з профілем діяльності замовника | 0,2 | 4 | 0,8 | 4 | 0,8 | |
| 4.Ресурсная ефективність | 0,05 | 4 | 0,2 | 4 | 0,2 | |
| 5. Надійність (захист даних) | 0,23 | 3 | 0,69 | 3 | 0,69 | |
| 6.Скорость доступу до даних | 0,06 | 4 | 0,24 | 3 | 0,18 | |
| 7.Гнучкість налаштування | 0,13 | 3 | 0,39 | 3 | 0,39 | |
| 8. Здатність до навчання персоналу | 0,09 | 4 | 0,36 | 4 | 0,36 | |
| Узагальнений показник якості 𝐽ЕТР | | 3,54 | | 3,2 | |  |

Відношення індексів для двох об’єктів називають коефіцієнтом технічного рівня (𝐴𝑘 ) першого об’єкту по відношенню до другого:

𝐴𝑘 = 𝐽ЕТР1𝐽ЕТР2.

Для зрівнюваних ПЗ та програмного продукту як його аналога коефіцієнт технічного рівня буде дорівнювати

𝐴𝑘 =3,54/3,2=1,106.

Оскільки коефіцієнт більше 1, то розробка ПЗ з технічної точки зору виправдана.

**6.3 Планування комплексу робіт з розробки програмного забезпечення і оцінка трудомісткості робіт**

Для розробки ПЗ задіяно дві людини: керівник проекту і виконавець (інженер-програміст). Керівник проекту формує завдання проекту (комплекс робіт), план виконання цих завдань, здійснює управління проектом розробки ПЗ, проводить необхідні консультації. Виконавець здійснює проєктування архітектури ПЗ, програмну реалізацію алгоритмів, розробку бази даних, розробку інтерфейсу, тестування програми та інші дії згідно з планом проекту. Вибір комплексу робіт щодо розробці проекту проводиться відповідно до стандарту ISO/IEC12207: 2008 «System and software engineering - Software life cycle processes», який встановлює стадії розробки програмних продуктів.

Приклад комплексу робіт наведено у таблиці 6.2.

На основі даних таблиці 6.2 розроблено календарний графік виконання робіт за проектом, який наведено в таблиці 6.3 і на рисунку 6.1, що показує послідовність і взаємозв'язок виконання комплексу робіт.

Таблиця 6.2 – Комплекс робіт проекту по розробці ПЗ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Зміст робіт | Виконавець | Тривалість, день | Навантаження викон. | | |
| Трив,день | % | людино-дні |
| 1. Підготовка процесу розробки і аналіз вимог | | | | | |
| 1.1 Дослідження та обґрунтування розробки | | | | | |
| 1.1.1 Постановка задачі | Керівник, Програміст | 3 | 1  3 | 25  100 | 0,25  3 |
| 1.1.2 Збір вихідних даних | Керівник, Програміст | 14 | 5  14 | 50  100 | 2,5  14 |
| 1.2 Пошук аналогів і прототипів | | | | | |
| 1.2.1 Аналіз існуючих методів розв’язання задачі | Програміст | 6 | 6 | 100 | 6 |
| 1.2.2 Обґрунтування необхідності розробки ПЗ | Керівник, Програміст | 2 | 1  2 | 50  100 | 0,5  2 |
| 1.3 Аналіз вимог | | | | | |
| 1.3.1 Визначення і аналіз вимог до ПЗ | Керівник, Програміст | 3 | 1  3 | 25  100 | 0,25  3 |
| 1.3.2 Визначення структури даних | Керівник, Програміст | 5 | 1  5 | 0,5  100 | 0,5  5 |
| 1.3.3 Вибір технічних і програмних засобів реалізації ПЗ | Керівник, Програміст | 3 | 1  3 | 0,5  100 | 0,5  3 |
| 1.3.4 Узгодження і затвердження технічного завдання розробки ПЗ | Керівник, Програміст | 3 | 1  3 | 0,5  100 | 0,5  3 |
| **Разом за етапом 1** | **Керівник, Програміст** | **39** | **11**  **39** | **28**  **100** | **5**  **39** |
| Закінчення таблиці 6.2. | | | | | |
|  | | | | | |
| 2. Проектування | | | | | |
| 2.1 Проектування програмної архітектури | Програміст | 3 | 3 | 100 | 3 |
| 2.2 Технічне проектування компонентів ПЗ | Програміст | 7 | 7 | 100 | 7 |
| **Разом за етапом 2** | **Програміст** | **10** | **10** | **100** | **10** |
| 3. Програмування та тестування програмних модулів | | | | | |
| 3.1 Програмування модулів ПЗ | Програміст | 13 | 13 | 100 | 13 |
| 3.2 Тестування програмних модулів | Програміст | 21 | 21 | 100 | 21 |
| 3.3 Збірка і випробування ПЗ | Керівник, Програміст | 5 | 2  5 | 40  100 | 0,8  5 |
| 3.4 Аналіз результатів випробувань | Керівник, Програміст | 5 | 1  5 | 20  100 | 0,2  5 |
| **Разом за етапом 3** | **Керівник, Програміст** | **44** | **3**  **44** | **7**  **100** | **1**  **44** |
| 4. Оформлення робочої документації | | | | | |
| 4.1 Проведення розрахунків показників безпеки життєдіяльності | Програміст | 3 | 3 | 100 | 3 |
| 4.2 Проведення економічних розрахунків | Програміст | 4 | 4 | 100 | 4 |
| 4.3 Оформлення пояснювальної записки | Керівник, Програміст | 15 | 6  15 | 0,5  100 | 3  15 |
| **Разом за етапом 4** | **Керівник, Програміст** | **22** | **6**  **22** | **27**  **100** | **3**  **22** |
| **Разом за проектом** | **Керівник, Програміст** | **115** | **20**  **115** |  | **9**  **115** |

Таблиця 6.3 – Календарний графік виконання робіт проекту

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Зміст роботи | Виконавець | Тривалість, день | Графік роботи | |
| Початок | Кінець |
| 1 | Постановка задачі | Керівник | 1 | 10.02.2020 | 10.02.2020 |
| Програміст | 3 | 10.02.2020 | 12.02.2020 |
| 2 | Збір вихідних даних | Керівник | 5 | 13.02.2020 | 16.02.2020 |
| Програміст | 14 | 13.02.2020 | 25.02.2020 |
| 3 | Аналіз існуючих методів розв’язання задачі | Програміст | 6 | 26.02.2020 | 01.03.2020 |
| 4 | Обґрунтування необхідності розробки ПЗ | Керівник | 1 | 02.03.2020 | 02.03.2020 |
| Програміст | 2 | 02.03.2020 | 03.03.2020 |
| 5 | Визначення і аналіз вимог до ПЗ | Керівник | 1 | 04.03.2020 | 04.03.2020 |
| Програміст | 3 | 04.03.2020 | 06.03.2020 |
| 6 | Визначення структури даних | Керівник | 1 | 07.03.2020 | 07.03.2020 |
| Програміст | 5 | 07.03.2020 | 10.03.2020 |
| 7 | Вибір технічних і програмних засобів реалізації ПЗ | Керівник | 1 | 11.03.2020 | 11.03.2020 |
| Програміст | 3 | 11.03.2020 | 15.03.2020 |
| 8 | Узгодження і затвердження технічного завдання розробки ПЗ | Керівник | 1 | 16.03.2020 | 16.03.2020 |
| Програміст | 3 | 16.03.2020 | 18.03.2020 |
| 9 | Проектування програмної архітектури | Програміст | 3 | 19.03.2020 | 21.03.2020 |

Закінчення таблиці 6.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | Технічне проектування компонентів ПЗ | Програміст | 7 | 22.03.2020 | 27.03.2020 |
| 11 | Програмування модулів ПЗ | Програміст | 13 | 28.03.2020 | 12.04.2020 |
| 12 | Тестування програмних модулів | Програміст | 21 | 13.04.2020 | 03.05.2020 |
| 13 | Збірка і випробування ПЗ | Керівник | 2  5 | 04.05.2020  04.05.2020 | 05.05.2020  08.05.2020 |
| Програміст |
| 14 | Аналіз результатів випробувань | Керівник | 1  5 | 09.05.2020  09.05.2020 | 09.05.2020  13.05.2020 |
| Програміст |
| 15 | Виконання робіт щодо охорони праці та навколишнього середовища | Програміст | 3 | 14.05.2020 | 16.05.2020 |
| 16 | Проведення економічного обґрунтування | Керівник  Програміст | 4 | 17.05.2020 | 19.05.2020 |
| 17 | Оформлення пояснювальної записки | Керівник | 5  15 | 20.05.2020  20.05.2020 | 24.05.2020  03.06.2020 |
| Програміст |



Рисунок 6.1 – Календарний графік проекту

**6.4 Розрахунок проектних витрат на розробку ПЗ**

Капітальні вкладення в проекти, пов’язані з розробкою і впровадженням програмних продуктів, розраховуються за формулою

𝐾= 𝐾𝑃+𝐾𝑅,

де *Кр* - капітальні вкладення на проектування (передвиробничі витрати), грн.;

*KR* - капітальні вкладення на реалізацію проекту, грн.

Сумарні витрати на проектування ПЗ, його розробку, тестування та налагодження визначаються за формулою:

𝐾𝑃=((1+𝑊𝑑)(1+𝑊𝑐)+𝑊𝐻)Σ𝐶𝐻𝑖𝑚𝑖=1+𝐶𝑀+𝑀𝑇,

де *m -*  кількість виконавців проекту, які беруть участь у розробці ПЗ;

*Chi* - витрати на основну заробітну плату виконавця *i*-ї категорії, грн.;

*Wd* - коефіцієнт, що враховує додаткову заробітну плату в частках до основної заробітної плати (встановлюється кожною організацією окремо, наприклад, це може бути кваліфікаційні надбавки за стаж, наукове звання, науковий ступінь тощо);

*Wc* - коефіцієнт, що враховує єдиний соціальний внесок (податок з фізичної особи, та інші податки та збори), в частках до суми основної та додаткової заробітної плати виконавця проекту (*Wc* = 22%);

*WH* - коефіцієнт, що враховує накладні витрати організації, у частках до основної заробітної плати виконавця проекту (приймається за фактичними даними, *WH* = 0,6);

*CM* - витрати на матеріали;

*MT* - витрати на використання машинного часу.

Витрати на заробітну плату працівника *i*-ї категорії наведено у таблиці 6.4.

Таблиця 6.4 –Основна заробітна плата виконавців проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада | Посадовий оклад, грн | Середня денна ставка 𝐶𝑇𝑖, грн. | Витрати часу на розробку *ti*,  людино-дні | Витрати на основну заробітну плату *CHi*, грн. |
| Керівник | 8000 | 8000/21=381 | 9 | 3429 |
| Програміст | 1300 | 1300/21= 61,9 | 115 | 7118,5 |
| Усього, *CH* | | | | 10547,5 |

Витрати на використання машинного часу *MT*. З огляду на те, що ПЗ повинне бути запрограмоване і налагоджене за допомогою комп’ютерів, до сумарних витрат на розробку додаються витрати на використання машинного часу, що обчислюються як:

𝑀𝑇=𝑡𝑀𝑇∙𝑆𝑀𝑇∙𝐾𝑀,

де *tMT* – машинний час комп'ютера, необхідне для розробки програмного продукту; *tMT* = 115\*8=920 годину;

*SMT*– вартість 1 год машинного часу; = 5 грн. / год;

*KM* - коефіцієнт мультипрограмності, який показує частку машинного часу, яке надається для реалізації проекту; у випадку виконання ДР *KM* = 1.

*MT* = 920\*5\*1 = 4600 грн

Витрати на матеріали *CM* .Матеріали, придбані у процесі виконання проекту, та їх вартість наведені в таблиці 6.5.

Таблиця 6.5 – Витрати на матеріали

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Матеріали | Од. вимірювання | Необхідна кількість | Ціна за од, грн. | Сума, грн. |
| Зошит загальний | шт. | 1 | 40 | 40 |
| Тонер для принтера | шт. | 1 | 110 | 110 |
| Папір офісний | Пачка | 1 | 100 | 100 |
| Усього | | | | 250 | |

Отже, капітальні вкладення проекту:

Кр = ((1+0,5)×(1+0,22)+0,6) 10547,5+274 + 4600 = 27704,74 грн.

Кошторис на розробку ПЗ наведено у таблиці 6.6.

Таблиця 6.6 – Кошторис на розробку ПЗ

|  |  |  |
| --- | --- | --- |
| Статті витрат | Сума, грн. | *Коментар* |
| Основна заробітна плата | 10547,5 | *Згідно табл. 2.4* |
| Додаткова зарплата | 5273,75 | 𝐶𝐻∙ 𝑊𝑑 |
| Відрахування | (3429 + 7118,5 \* 1 ) \* 0,22 = 2320,45 | (𝐶𝐻∙+𝐶𝐻∙ 𝑊𝑑)∙𝑊𝑐 |
| Витрати на матеріали | 250 | *Згідно табл. 2.5* |
| Витрати на машинний час | 4600 | *Згідно (2.7)* |
| Накладні витрати організації | 6 328,5 | 𝐶𝐻∙ 𝑊𝐻 |
| ИТОГО | | 29320,2 |

Отже, на розробку ПЗ витрати *Кр* склали 29320,2грн.

**6.5 Розрахунок витрат на впровадження ПЗ**

Капітальні вкладення проекту на впровадження ПЗ розраховують за формулою:

𝐾𝑅=𝐾𝐴𝐸+𝐾𝑅𝐷+ 𝐾𝑃𝑃+𝐾𝑂𝑆+𝐾𝐵𝐷+𝐾𝐻𝑅,

де *КАЕ* - витрати на основне і допоміжне обладнання, грн.;

*KRD* - витрати на будівництво, реконструкцію будівлі та приміщень, грн.;

*Kpp* - витрати на придбання типових розробок, пакетів, грн.;

*Kos* - витрати на прокладку ліній зв'язку, грн.;

*KBD* - витрати на створення інформаційної бази, грн.;

*KHR* - витрати на підготовку і перепідготовку кадрів, грн.

Оскільки у даному прикладі вважається, що для впровадження ПЗ не потрібні кошти на створення інформаційної бази організації, ліній зв’язку, на реконструкцію будівель, то відповідні витрати у дорівнюють нулю. Для впровадження ПЗ не потрібно великих часових та інших ресурсів для підготовки кадрів, отже, ці витрати також будуть дорівнювати нулю.

Вважається, що на підприємстві, де буде впроваджуватися ПЗ, можна здійснити додатково закупівлю ПК та програмне забезпечення. Отже, необхідні кошти5 на:

– закупівлю основного обладнання, до якого входить ПК вартістю 15000 грн.

Отже, витрати на закупівлю додаткового обладнання складуть 15000 грн.

Оскільки це обладнання буде використовуватися і для інших програм, то визначення витрат *КАЕ* здійснюється на основі формули: 𝐾𝐴𝐸=Σ𝐶𝑏𝑗∙𝑄𝑗𝑁𝑗=1∙𝑌𝑗,

де *N* – кількість видів основного та допоміжного обладнання, необхідного для реалізації проекту (для даного проєкту *N* = 1);

*Cbj* - балансова вартість *j*-го виду обладнання, грн. (*Cbj* = 15000 грн.);

Qj - кількість одиниць *j*-гo обладнання, шт. *Qj* = 1 шт.);

*Yj* - коефіцієнт завантаження *j*-го виду обладнання при обробці інформації за рішенням завдань предметної області:

𝑌𝑗=𝑇𝑗𝐹𝑇𝑗⁄,

де *FTj* - ефективний річний фонд часу роботи технічного засобу *j*-го виду, год./рік (це може бути 2008годин (251 *днів* х 8*год* ), у 2020 році загальна кількість днів складе 366, робочих днів – 251);

*Tj* – час роботи технічного засобу *j*-го виду для розв’язку *S* задач, год./рік:

𝑇𝑗=Σ𝑡𝑘𝑗𝑆𝑘=1∙𝑈𝑘

де *tkj* – трудомісткість одноразової обробки інформації *к*-ї задачі на *j*-му виді технічних засобів, годин машинного часу ( у даному проекті вважається, що виконується 1 задача з трудомісткістю *tkj*= 6 );

*Uk* - частота (періодичність) рішення *к*-ї задачі, днів/рік (*Uк* = 251).

Для даного проєкту витрати на його реалізацію *KAE* складуть:

𝐾𝑅=𝐾𝐴𝐸+𝐾𝑃𝑃=15000∙1∙6∙251/2008=11250 грн.

Отже, сумарні витрати на реалізацію проекту щодо розробки ПЗ складуть: 𝐾1= 𝐾𝑃+𝐾𝑅=29320,2+11250=40570,2 грн.

**6.6 Розрахунок витрат на закупівлю та впровадження програмного продукту – аналога ПЗ, яке розроблено**

Розрахуємо сумарні витрати, які пов’язані з впровадженням аналога (zoo-baza.com.ua), з яким порівнюється ПЗ, яке розроблено у ДР. Такі витрати складаються з:

– витрат на основне та допоміжне обладнання. До такого обладнання входить ПК, серверне, мережеве обладнання тощо (знадобиться такий же ПК, що і для ПЗ вартістю 15000 грн., а також додаткове ПЗ вартістю 4000 грн);

𝐾𝑅=𝐾𝐴𝐸+𝐾𝑃𝑃=15000∙1∙6∙251/2008 + 4000 =15250 грн.

Отже, сумарні витрати на впровадження Інтернет магазину zoo-baza.com.ua – аналога ПЗ, що розроблено, складуть

𝐾2=0 + 18250= 15250 грн.

**6.7 Розрахунок поточних експлуатаційних витрат ПЗ та його аналога**

До експлуатаційних витрат відносяться витрати, які пов’язані з забезпеченням нормального функціонування ПЗ. Такі витрати називають також поточними витратами. Це можуть бути витрати на ведення інформаційної бази, експлуатацію комплексу технічних засобів, експлуатацію систем програмно-математичного забезпечення, реалізацію технологічного процесу обробки інформації згідно з задачами, експлуатація системи у цілому. Поточні витрати розраховуються за формулою:

𝐶С=𝐶𝑆+𝐶𝐷+𝐶𝐸+𝐶𝑅𝐸+𝐶𝑀+𝐶𝑂𝐻 ,

де Cs – витрати на зарплату основну та додаткову з відрахуваннями до соціальних фондів, грн.;

CD – амортизаційні відрахування від вартості обладнання і пристроїв системи, грн.;

CE – витрати на електроенергію, грн.;

CRE – витрати на поточний ремонт обладнання та пристроїв системи, грн.;

CM – витрати на матеріали і носії інформації, грн.;

COH – накладні витрати інформаційного відділу, грн.

Проведемо розрахунки цих витрат на 1 рік експлуатації програмного продукту (ПЗ та його аналога).

1. Витрати () на заробітну плату фахівців – користувачів програмного продукту розраховують так:

𝐶𝑆= Σ𝑀𝑖=1𝑡𝑖∙𝐶𝑆𝑖(1+𝑊𝑑)(1+𝑊𝑐) ,

де *ti* – час експлуатації програми *i*-м працівником, у днях;

*M* – кількість працівників (фахівців), які будуть користуватися ПЗ;

Csi– середньоденна заробітна плата *i*-го працівника, грн./день.

Значення надбавки до заробітної плати Wd встановлюється організацією самостійно, може дорівнювати нулю; значення Wc залежить від типу підприємства, де буде впроваджено програмний продукт (від схеми оподаткування бізнесу).

Дані розрахунку заробітної плати фахівців– користувачів ПЗ та його аналога наведені відповідно у таблиці 6.7 і 6.8.

Таблиця 6.7 – Дані по заробітній платі користувачів ПЗ, що розроблено

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада | Посадовий оклад, грн. | Середня денна ставка, грн./день | Витрати часу на роботу з ПЗ, людино-дні | Фонд з/п, грн. |
| Програміст, (системний адміністратор) | 18000 | 857,14 | 2 день \*12 міс = 24 | 20 571,36 |
| Разом | | | | 20571,36 |

𝐶𝑆1=(857,14∙24)(1+0,4)(1+0,22)=35135,88 грн. (за рік)

Таблиця 6.8 – Дані по заробітній платі фахівців (аналог)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада | Посадовий оклад, грн. | Середня денна ставка, грн./день | Витрати часу на роботу з програмою, людино-дні | Фонд з/п, грн. |
| Програміст, (системний адміністратор) | 21000 | 1000 | 3 дні\*12 міс = 36 | 36000 |
| Разом | | | | 36000 |

𝐶𝑆2=(1000 \* 36)(1+0,4)(1+0,22)=61488 грн. (за рік).

2. Сума амортизаційних відрахувань (𝐶𝐷) для обладнання розраховується так:

𝐶𝐷=Σ𝐶𝑏𝑗∙𝑎𝑗∙𝑔𝑗∙𝑡𝑗𝐹𝑇𝑗𝑁𝑗=1 ,

де C*bj*– балансова вартість *j*-гo виду обладнання, грн.;

t*j*– час роботи *j*-гo виду обладнання, години;

a*j*– норма річних амортизаційних відрахувань для *j*-гo виду обладнання;

F*Tj*– ефективний фонд часу роботи обладнання в рік, години;

g*j*– кількість одиниць обладнання *j*-гo виду.

Дані для ПЗ та його аналога:

a*j*= 0,2 (використовується прискорена амортизація – 20-30% оскільки мова йде про комп’ютерну техніку, яка має моральний знос);

g*j* = 1 (використовується один ПК);

t*j*(для проекту) = 24× 8 = 192 год ;

t*j* (для аналога) = 36 × 8 = 288 год.;

C*bj* = 15000грн .; C*bj* = 15000грн.

Сума амортизаційних відрахувань для ПЗ складе:

𝐶𝐷1=15000∙0,2∙1∙192/2008=286,85 грн.

Сума амортизаційних відрахувань для аналога складе:

𝐶𝐷2=15000∙0,2∙1∙288/2008=430,27 грн.

3. Витрати на електроенергію 𝐶 розраховуються за формулою:

𝐶𝐸*=Σ𝑉𝑗 ∙𝑡𝑗 ∙𝑞𝑗 ∙𝑇𝐸 𝑁𝑗=1,*

деV*j*– встановлена потужність j-го виду технічних засобів, кВт;

q*j* - коефіцієнт використання встановленої потужності обладнання;

T*E* - тариф на електроенергію, грн./кВт год.

На даний час тариф на електроенергію складає 1,68 гривен/кВт\*ч., встановлена потужність для комп'ютера дорівнює 0,4 кВт, таким чином витрати на силову енергію для ПЗ складуть:

𝐶𝐸1=0,4∙1∙192∙1,68=129,02 грн.

Для аналога:

𝐶𝐸2=0,4∙1∙960∙1,68=193,53 грн.

4. Витрати C*RE* на поточний ремонт обладнання розраховуються за формулою:

𝐶𝑅𝐸=Σ𝑁𝑅𝑗∙𝐶𝑏𝑗∙𝑡𝑗𝐹𝑇𝑗𝑁𝑗=1,

де N*Rj*– норматив витрат на ремонт (*Cpi* = 0,05).

Витрати на поточний ремонт обладнання складуть:

- для ПЗ C*RE1* = (0,05\*192\*15000)/2008 = 71,71 грн.

-для аналогу C*RE2* = (0,05\*288\*15000)/2008 = 107,57 грн.

5. Витрати 𝐶𝑀 на матеріали, які споживаються протягом року, складають 1% від балансової вартості основного обладнання.

Отже, для ПЗ, що розроблене, та для його аналога ці витрати будуть однакові, оскільки використовується обладнання – ПК однієї вартості.

𝐶𝑀1=𝐶𝑀2=15000∙0,01=150 грн.

6. Накладні витрати включають витрати на утримання адміністративного та управлінського персоналу, на утримання приміщення та інші витрати (див. додаток А). Розподіл накладних витрат це один з найважливіших процесів на підприємстві, що надає змогу не занижувати результати діяльності підприємства або до їх завищення. Існують різні бази Розподіл накладних витрат, найчастіше, здійснюють пропорційно:

– зарплаті виробничих робітників;

– до кількості виготовленої продукції (наданих послуг),

– прямим матеріальним витратам,

– часу обробки виробів на основі середніх витрат на одну машино-годину та ін.

Вибір бази розподілу визначається з урахуванням специфіки підприємства, його галузевих особливостей, співвідношень окремих статей витрат у структурі загальних витрат підприємства. Отже, в залежності від роду діяльності виділяють і застосовують різні бази розподілу накладних витрат.

У даному випадку будемо вважати, що норматив накладних витрат встановлюється як 20% від прямих витрат, що включають перші п’ять статей витрат з таблиці 6.9.

Таблиця 6.9 – Річні експлуатаційні витрати програмного продукту

|  |  |  |  |
| --- | --- | --- | --- |
| № | Статі витрат | Витрати на ПЗ, грн. | Витрати на аналог, грн. |
| 1 | Основна і додаткова зарплата з відрахуванням, Cs | 35135,88 | 61488 |
| 2 | Амортизовані відрахування, CD | 286,85 | 430,27 |
| 3 | Витрати на електроенергію, CE | 129,02 | 193,53 |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Витрати на поточний ремонт, CRE | 71,71 | 107,57 |
| 5 | Витрати на матеріали CM | 150 | 150 |
| 6 | Накладні витрати | 7027,17 | 12297,6 |
| Усього витрат, Cc | | 42800,63 | 74473,44 |

Накладні витрати для ПЗ складуть:

𝐶𝑂𝐻1=(35135,88+268,85+129,02+71,71+150)∙0,2=7027,176 грн.

Накладні витрати для аналогу:

𝐶𝑂𝐻2=(61488+430,27+193,53+107,57+150)∙0,2= 12297,6 грн.

Отже, поточні експлуатаційні витрати 𝐶С ПЗ, яке розроблене, складуть 42800,63грн., а його аналога – 74473,44 грн.

**6.8 Розрахунок показника економічного ефекту від розробки ПЗ**

Оцінка економічної ефективності варіантів програмних продуктів ґрунтується на розрахунку показників порівняльної економічної ефективності капітальних вкладень у їх розробку. Річний економічний ефект від використання розробленого програмного продукту визначається по різниці приведених витрат на розробку та експлуатацію за рік ПЗ та витрат на закупку та експлуатацію його аналога (у розрахунку на річний обсяг випуску продукції/послуг, робіт):

𝐸=(С2∙𝐴𝑘 −С1)𝑁Σ ,

де C1, C2 – приведені витрати на одиницю робіт, що виконуються за допомогою ПЗ, що розроблено, та його аналогу, грн.;

𝑁Σ – обсяг робіт, які виконуються за допомогою програмного продукту

Наведені витрати на одиницю робіт, що виконуються за допомогою ПЗ, що розроблено, та його аналогу, розраховуються за формулою:

C = Cc + En \* K ,

де Cc– собівартість (як поточні експлуатаційні витрати одиниці роботи/задачі), грн.;

En– нормативний коефіцієнт економічної ефективності (приймаємо у даному випадку En = 0,33)15;

K – сумарні витрати, які пов’язані з розробкою (закупкою) та впровадженням програмного продукту.

Витрати на виконання одиниці роботи (задачі) за допомогою ПЗ, який розроблено, складають:

С1=42800,63 +0,33∙11250=46513,13 грн.

Приведені витрати на одиницю виконання роботи за допомогою аналогу складуть:

С2=74473,44+0,33∙15250=79505,5 грн.

Тоді економічний ефект від використання розробленого ПЗ складе:

*Е =* (79505,5\*1,106 – 46513,13)\*1 = 41419,95

Зведені дані за розрахунками економічного ефекту наведені у таблиці 6.10.

Таблиця 6.10 – Економічний ефект розробленого ПЗ

|  |  |  |
| --- | --- | --- |
| Характеристика | Значення для | |
| ПЗ | аналогу |
| Собівартість робіт (поточні експлуатаційні витрати програмного продукту), грн. | 42800,63 | 74473,44 |
| Сумарні витрати, які пов’язані з впровадженням ПЗ, грн. | 11250 | 15250 |
| Приведені витрати на одиницю робіт, грн. | 46513,13 | 79505,5 |
| Економічний ефект від використання програмного продукту, грн. | 41419,95 | |

Після визначення річного економічного ефекту необхідно розрахувати

термін окупності витрат на розробку програмного продукту за формулою:

𝑃𝑝=𝐾/𝐸 (2.20)

Термін окупності ПЗ складе:

𝑃𝑝=11250 /79505,5 =0,14 року

Розрахуємо фактичний коефіцієнт економічної ефективності розробки (*Efact*) і порівняємо його з нормативним значенням коефіцієнта ефективності капітальних вкладень *En*= 0,33:

𝐸𝑓𝑎𝑐𝑡=1/𝑃𝑝=1/0,14=7,14

Фактичний коефіцієнт економічної ефективності розробки ПЗ більше, ніж нормативний (7,14 >0,33), тому розробка та впровадження ПЗ є ефективно доцільною.

Отже, знайдені усі дані, які доводять визначити технічну доцільність і економічну ефективність розробленого ПЗ.

7 ОХОРОНА ПРАЦІ ТА НАВКОЛИШНЬГО СЕРЕДОВИЩА

**7.1 Загальні питання охорони праці**

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності. Роботодавець – власник підприємства, установи, організації або уповноважений ним орган, незалежно від форм власності, виду діяльності, господарювання, і фізична особа, яка використовує найману працю. Працівник - особа, яка працює на підприємстві, в організації, установі та виконує обов'язки або функції згідно з трудовим договором (контрактом) [9].

Актуальність проблеми охорони здоров’я в нашій державі ґрунтується на ряді чинників, які призводять незадовільної ситуації в цій сфері:

* Недофінансування заходів охорони праці зі сторони роботодавців;
* Велика кількість випадків не розслідуються або не становляться на облік;
* Частині випадків дають іншу кваліфікацію (як не виробничих);

Як висновок відсутність реальної картини про стан справ в цій сфері заважає приймати необхідні заходи для поліпшення ситуації і може здатися, що ситуація стабільна і не потребує особливих змін.

Розділ виконано для етапу розрахунку на ЕОМ веб-сервісу з продажу товарів для домашніх тварин.

При написанні розділу була використана наступна науково - технічна документація:

* НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями.
* ДСанПіН 3.3.2.007-98. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.
* ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень.
* ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціонування.
* ДБН В.2.5-28-2018. Природне та штучне освітлення.
* ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку.
* ДСТУ ГОСТ 12.1.012:2008. Вібраційна безпека. Загальні вимоги.
* ДСН 3.3.6.039-99. Санітарні норми виробничої загальної та локальної вібрації.
* ДБН В.2.5-28-2016. Державні будівельні норми України. Інженерне обладнання будівель i споруд. Природне i штучне освітлення.
* ДСТУ ГОСТ 12.1.012:2008. Вібраційна безпека. Загальні вимоги.
* ПУЕ. Правила улаштування електроустановок.
* ДСТУ ГОСТ 12.1.038:2008. Електробезпека. Гранично допустимі значення напруги дотику і струму.
* ДСТУ БВ.1.1-36:2016. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежноюнебезпекою.
* ДБН В 1.1-7-2016. Пожежна безпека об’єктів будівництва. Загальні вимоги.
* НАПБ А.01.001-2014. Правила пожежної безпеки в Україні.

Робота виконувалась на робочому місці на кафедрі, яке розташована на 3 поверсі 3-поверхової будівлі.

На користувача ЕОМ під час роботи на комп’ютері з монітором на рідинних кристалах діють небезпечні і шкідливі виробничі фактори, які приведені у таблиці 7.1.

Таблиця 7.1 – Потенційні шкідливі і небезпечні фактори, що впливають на користувача під час роботи на комп’ютері

|  |  |  |
| --- | --- | --- |
| Шкідливі і  небезпечні виробничі фактори | Джерела їх  утворення | Характер дії |
| Несприятливий мікроклімат у приміщенні | Стан систем опалення та вентиляції | Шкідливий |
| Недостатня освітленість робочого місця | Стан систем природного та штучного освітлення | Шкідливий |
| Підвищений шум, вібрація | Принтери, система охолодження процесора, кондиціонер тощо | Шкідливий |
| Небезпека ураження електричним струмом | Електрична мережа живлення устаткування | Небезпечний |
| Пожежа небезпека приміщення | Присутні джерела можливого виникнення пожежі (електромережа) та матеріали, що здатні горіти | Шкідливий,  небезпечний |

Розміри приміщення з ЕОМ повинні відповідати вимогам ДСанПіН 3.3.2.007-98: на одне робоче місце з ЕОМ повинно приходитися не менше 6 м2 площі та 20 м3 об’єму приміщення. Площа приміщення, в якому виконувалася робота, 6х3 м2, висота 2,7 м; кількість робочих місць - 2. У даному випадку на одне робоче місце фактично приходиться 9 м2 площі і 24,3 м3 об’єму приміщення. Таким чином розміри приміщення відповідають нормативним вимогам ДСанПіН 3.3.2.007-98.

Робота користувача під час роботи за комп’ютером – це легка робота категорії Iа, тому що виконується сидячи. Для користувачів повинні підтримуватися оптимальні параметри мікроклімату у приміщенні, які відповідають вимогам ДСН 3.3.6.042-99 і приведені в таблиці 2.3.2

Таблиця 7.2 – Оптимальні параметри мікроклімату виробничого приміщення

|  |  |  |  |
| --- | --- | --- | --- |
| Час року | Температура, | Відносна вологість, % | Швидкість руху повітря, м/с  (не більш) |
| Холодний | 22-24 | 40-60 | 0,1 |
| Теплий | 23-25 | 40-60 | 0,1 |

Оптимальні параметри мікроклімату на робочому місці підтримують відповідно до ДБН В.2.5-67:2013 в теплий період року за допомогою кондиціонеру, а в холодний – опаленням (загальним паровим) від центральної тепломережі. Треба передбачити можливість природного провітрювання.

Освітлення на робочому місці забезпечується за рахунок природного і штучного освітлення. Спостереження за екраном комп’ютеру відповідає ІІІ розряду зорової роботи (розмір об’єктів, що спостерігаються, складає від 0,3 до 0,5 мм − роботи високої точності) за ДБН В.2.5-28-2018. Природнє освітлення нормується показниками КПО, %. Нормовані значення КПО для сумісного освітлення: Д*нсуммін*=1,2 %. Штучне освітлення у приміщенні – загальне і нормується мінімальним освітленням . Для зорових робіт ІІІ розряду, підрозряду «в» (контраст середній, фон середній) нормована освітленість при загальному освітленні дорівнює 300 лк.

Природне освітлення підтримується за рахунок площі вікон, яке розраховується під час проектування будівель, а штучне − за рахунок визначеної кількості світильників і вибору потужності ламп в них.

Принтери, кондиціонер, система охолодження процесорного блоку є джерелом шуму та вібрації. Відповідно до ДСН 3.3.6.037-99 допустимий рівень звуку на робочому місці користувача не повинний перевищувати 50 дБА. Нормативні параметри вібрації регламентуються ДСТУ ГОСТ 12.1.012:2008 та ДСН 3.3.6.039-99. Для загальної технологічної вібрації категорії 3 типу «в» умов «комфорт» рівень віброшвидкості не повинен перевищувати 75 дБ.

При роботі за комп’ютером важливим фактором є небезпека поразки людини електричним струмом. Обладнання живиться від трифазної чотирьох провідної мережі напругою 380/220 В, частотою 50 Гц з глухо заземленою нейтраллю. За небезпекою поразки людини електричним струмом приміщення відноситься до приміщень «з підвищеною небезпекою», так як є можливість одночасного дотику до металевих корпусів обладнання та металоконструкцій будівлі, що пов’язані із землею (ПУЕ). За ДСТУ ГОСТ 12.1.038:2008 найбільша допустима напруга дотику В, струм через людину  мА.

Пожежна безпека об’єкта, це стан, коли виключається можливість пожежі, а у випадку її виникнення запобігається вплив на людей небезпечних та шкідливих факторів пожежі і забезпечується захист матеріальних цінностей.

Причиною пожежі можуть бути аварії і неполадки в роботі електроустановок:

* пошкодження ізоляції;
* коротке замикання кола струму;
* перегрів проводів;
* поганий контакт в місцях з’єднання тощо;
* Категорія пожежної безпеки приміщення В, так як присутні тверді матеріали, що здатні горіти: столи, стільці, ізоляція та інше (ДСТУ БВ.1.1-36:2016).

Ступінь вогнестійкості будівлі – ІІІ за ДБНВ 1.1-7-2016. У приміщенні забезпечується протипожежна сигналізація.

**7.2 Охорона навколишнього природного середовища**

Актуальність проблеми захисту навколишнього середовища полягає насамперед в тому, що з розвитком виробництва ми вилучаємо все більше ресурсів з природи це призводить до зростання їх вартості, а також до збільшення кількості відходів. Але цю проблему не доцільно вирішувати шляхом скорочення або взагалі припинення виробництвом. Бо такі заходи ідуть врозріз з сутністю людини і на практиці є нездійсненими.

Тому для вирішення цього питання країни приймають закони, які регулюють використання природних ресурсів і обсяг шкідливих відходів.

В Україні це закон «Про охорону навколишнього природного середовища» передбачає заходи, які спрямовані на організацію раціонального природокористування та захист навколишнього середовища [10].

При масовому використанні моніторів та комп’ютерів не можна не враховувати їхній вплив на навколишнє середовище на всіх стадіях – при виготовленні, експлуатації та після закінчення терміну служби.

Міжнародні екологічні стандарти, що діють на сьогоднішній день в усьому світі, визначають набір обмежень до технологій виробництва та матеріалів, які можуть використовуватися в конструкціях пристроїв. Так, за стандартом ТСО-95, вони не повинні містити фреонів ( турбота про озоновий шар), полівінілхлоридів, бромідів (як засобів захисту від загоряння).

У стандарті ТСО-99 закладене обмеження за кадмієм у світлочутливому шарі екрана дисплея та ртуті в батарейках; э чіткі вказівки відносно пластмас, лаків та покриттів, що використовуються. Поверхня кнопок не повинна містити хром, нікель та інші матеріали, які визивають алергічну реакцію. ГДК пилу дорівнює 0,15 мг/м3, рекомендовано 0,075 мг/м3; ГДК озону під час роботи лазерного принтеру − 0,02 мг/м3. Особливо жорсткі вимоги до повторно використовуваних матеріалів.

Апарати, тара і документація повинні допускати нетоксичну вторинну переробку після закінчення терміну експлуатації.

Міжнародні стандарти, починаючи з ТСО-92, включають вимоги зниженого енергоспоживання та обмеження припустимих рівнів потужності, що споживаються у неактивних режимах.

ВИСНОВКИ

Результатом дипломної роботи є розроблений веб-сервіс з продажу товарі для домашніх тварин.

В ході розробки був реалізований зрозумілий і приємний для користувача інтерфейс, реєстрація і авторизація користувача, можливість пошуку товарів з будь-якої сторінки, можливість додання і видалення товару з кошику, можливість замовити товар, можливість змінити мову інтерфейсу, можливість перегляду особистих даних і можливість вибору товару.

У майбутньому слід реалізувати зв’язок з різними платіжними системами для проведення платежів. Для цього необхідно розгорнути веб-сервіс у мережі Інтернет та зареєструватися на Liqpay, який є веб-інтерфейсом Банку АТ КБ “ПриватБанк”.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1.IDEF. [Електронний ресурс]//Режим доступу: https://en.wikipedia.org/wiki/IDEF 10.06.2020.

2. Scientific modelling. [Електронний ресурс]//Режим доступу: <https://en.wikipedia.org/wiki/Scientific_modelling> 08.06.2020.

3. Підручник з Umbrello UML Modeller. [Електронний ресурс]//Режим доступу: [https://docs.kde.org/stable5/uk/kdesdk/umbrello/index.html 10.06.2020](https://docs.kde.org/stable5/uk/kdesdk/umbrello/index.html%2010.06.2020).

4. Олищук А. В. Разработка Web-приложений на PHP 5. Профессиональная работа. / Олищук А. В. – М.: «Вильямс», 2006. – С. 352.

5. Еккель Б. Философия Java. 4-е полное изд. – СПБ.: Питер, 2015 – 1168с.: ил. – (Серия “Классика computer science”).

6. Системы управления базы данных. Курс лекций [Електронний ресурс]//Режим доступу: [http://www.bseu.by/it/tohod/lekcii5.htm 10.06.2020](http://www.bseu.by/it/tohod/lekcii5.htm%2010.06.2020).

7. Сейед Тахагхогхи. Руководство по MySQL / Пер. с англ./ Сейед Тахагхогхи, Хью Е. Вильямс — М. : Издательство «Русская редакция» ; 2007. — 544 стр. : ил.

8. Машнин Т. С. “Eclipse: разработка RCP-, Web-, Ajax- и Android-приложений на Java”. - БХВ-Петербург, 2013 год - 384 с.

9. Apache Tomcat. [Електронний ресурс]//Режим доступу: <http://tomcat.apache.org/> 03.06.2020.

10. HTML Tutorial. [Електроний ресурс]//Режим доступу - [https://www.w3schools.com/html/ 04.06.2020](http://htmlbook.ru/html%2004.06.2020).

11. Блинов И.Н. Java. Методы програмирования : уч-мет. пособие / И.Н. Блинов, В.С. Романчик. – Минск: издательство “Четыре четверти”, 2013. – 896 с.

12. Герберт Шилдт «Java. Полное руководство, 10-е издание» - Диалектика, 2018 год. - 1488 с.

13. Maven Getting Started Guide. [Електронний ресурс]//Режим доступу - [http://maven.apache.org/guides/getting-started/index.html 05.06.2020](http://maven.apache.org/guides/getting-started/index.html%2005.06.2020).

14. Закон України «Про охорону праці» № 2694-12 від 27.12.2019 року.

15. Закон України «Про охорону навколишнього середовища» №1264-12 від 07.06.2020 року.

ДОДАТОК А

Тексти програм

PetShopEmbeddedDBManager.java

package maksym.db;

import com.wix.mysql.EmbeddedMysql;

import com.wix.mysql.Sources;

import com.wix.mysql.SqlScriptSource;

import com.wix.mysql.config.MysqldConfig;

import com.wix.mysql.distribution.Version;

import java.io.File;

public class PetShopEmbeddedDBManager {

private static final String SCHEMA\_NAME = "PetShop";

private static final String SCRIPT\_FILE\_NAME = "PetShop.sql";

private static final String USER\_NAME = "Maksym";

private static final String PASSWORD = "pwEzhF258YPjZUG8";

private static PetShopEmbeddedDBManager INSTANCE;

public static PetShopEmbeddedDBManager getInstance() {

if (INSTANCE == null) {

INSTANCE = new PetShopEmbeddedDBManager();

}

return INSTANCE;

}

private EmbeddedMysql mysqld;

private PetShopEmbeddedDBManager() {

}

public void start() {

MysqldConfig config = MysqldConfig.aMysqldConfig(Version.v5\_7\_10)

.withPort(3306)

.withTempDir("../temp")

.withUser(USER\_NAME, PASSWORD)

.build();

mysqld = EmbeddedMysql.anEmbeddedMysql(config)

.addSchema(SCHEMA\_NAME)

.start();

}

public void init() {

if (mysqld == null) {

return;

}

File scriptFile = new File(getClass().getClassLoader().getResource(SCRIPT\_FILE\_NAME).getFile());

SqlScriptSource startScript = Sources.fromFile(scriptFile);

mysqld.executeScripts(SCHEMA\_NAME, startScript);

}

public void stop() {

if (mysqld == null) {

return;

}

mysqld.stop();

}

}

EmbeddedDbContextListener.java

package maksym.db;

import javax.servlet.ServletContextEvent;

import javax.servlet.ServletContextListener;

import javax.servlet.annotation.WebListener;

@WebListener

public class EmbeddedDbContextListener implements ServletContextListener {

private PetShopEmbeddedDBManager embeddedDBManager = PetShopEmbeddedDBManager.getInstance();

public void contextInitialized(ServletContextEvent event) {

embeddedDBManager.start();

embeddedDBManager.init();

}

public void contextDestroyed(ServletContextEvent event) {

embeddedDBManager.stop();

}

}

AddProductServlet.java

package maksym.web;

import maksym.db.ProductDAO;

import maksym.db.UserDAO;

import maksym.db.entity.Product;

import maksym.db.entity.User;

import org.apache.commons.lang3.StringUtils;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.regex.Pattern;

@WebServlet("/AddProduct")

public class AddProductServlet extends HttpServlet {

private static final String NUMBER\_REG\_EXP = "/d[0-9]+";

private static final String ONE\_NUM\_REG\_EXP = "/d[0-9]";

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

RequestDispatcher dispatcher = request.getRequestDispatcher("AddProduct.jsp");

dispatcher.forward(request,response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String name = request.getParameter("name");

String price = request.getParameter("price");

String description = request.getParameter("description");

String amount = request.getParameter("amount");

String weight = request.getParameter("weight");

String producer = request.getParameter("producer");

String type\_id = request.getParameter("type\_id");

String age = request.getParameter("age");

String breed = request.getParameter("breed");

String pet\_id = request.getParameter("pet\_id");

String photo\_link = request.getParameter("photo\_link");

Map<String, String> errors = new HashMap<>();

List<Product> allProd = ProductDAO.getInstance().getAllProducts();

for (Product p:allProd) {

if(p.getName().equals(name)&& p.getWeight().equals(weight)) {

errors.put("prodOrigin","Such a product already exists");

}

}

if(StringUtils.isBlank(name)){

errors.put("name", "Provide name product");

}

if(StringUtils.isBlank(price)){

errors.put("price", "Provide price product");

}

if(StringUtils.isBlank(description)){

errors.put("description", "Provide description product");

}

if(StringUtils.isBlank(amount)){

errors.put("amount", "Provide amount product");

}

if(StringUtils.isBlank(weight)){

errors.put("weight", "Provide weight product");

}

if(StringUtils.isBlank(producer)){

errors.put("producer", "Provide producer product");

}

if(StringUtils.isBlank(type\_id)){

errors.put("type\_id", "Provide type product");

}

if(StringUtils.isBlank(age)){

errors.put("age", "Provide age ");

}

if(StringUtils.isBlank(breed)){

errors.put("breed", "Provide breed ");

}

if(StringUtils.isBlank(pet\_id)){

errors.put("pet\_id", "Provide pet\_id product");

}

if(StringUtils.isBlank(photo\_link)){

errors.put("photo\_link", "Provide photo link product");

}

boolean foundPrice = Pattern.matches(NUMBER\_REG\_EXP, price);

if(foundPrice){

errors.put("pricePattern", "The entered price is incorrect");

}

boolean foundAmount = Pattern.matches(NUMBER\_REG\_EXP, amount);

if(foundAmount){

errors.put("amountPattern", "The entered amount is incorrect");

}

boolean foundWeight = Pattern.matches(NUMBER\_REG\_EXP, weight);

if(foundWeight){

errors.put("weightPattern", "The entered weight is incorrect");

}

boolean foundPet\_id = Pattern.matches(ONE\_NUM\_REG\_EXP, pet\_id);

if(foundPet\_id){

errors.put("pet\_idPattern", "The entered pet\_id is incorrect");

}

boolean foundType\_id = Pattern.matches(ONE\_NUM\_REG\_EXP, type\_id);

if(foundType\_id){

errors.put("type\_idPattern", "The entered type\_id is incorrect");

}

if (!errors.isEmpty()) {

RequestDispatcher dispatcher = request.getRequestDispatcher("AddProduct.jsp");

request.setAttribute("errors", errors);

dispatcher.forward(request, response);

return;

}

Product prod = Product.createProduct(name, price, description, amount, weight, producer,

Integer.parseInt(type\_id), age, breed, Integer.parseInt(pet\_id), photo\_link);

ProductDAO.getInstance().insertProduct(prod);

String direct = "Catalog?pet="+ type\_id;

response.sendRedirect(direct);

}

}

AboutProductServlet.java

package maksym.web;

import java.io.IOException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import maksym.db.ProductDAO;

import maksym.db.entity.Product;

/\*\*

\* Servlet implementation class AboutProduct

\*/

@WebServlet("/AboutProduct")

public class AboutProductServlet extends HttpServlet {

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse

\* response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int prodID = Integer.parseInt(request.getParameter("id"));

Product prod = null;

prod = ProductDAO.getInstance().getProductForId(prodID);

String breed;

if (prod.getBreed() != null) {

breed = prod.getBreed();

} else {

breed = "None";

}

request.setAttribute("prod", prod);

request.setAttribute("breed", breed);

RequestDispatcher dispatcher = request.getRequestDispatcher("AboutProduct.jsp");

dispatcher.forward(request, response);

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

\* response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

BasketServlet.java

package maksym.web;

import maksym.db.entity.Basket;

import maksym.db.entity.Product;

import org.apache.commons.lang3.StringUtils;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.util.List;

@WebServlet(value = "/basket", name = "BasketServlet")

public class BasketServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html; charset=UTF-8");

HttpSession session = request.getSession();

Basket basket = (Basket) session.getAttribute("basket");

if (basket == null ) {

RequestDispatcher dispatcher = request.getRequestDispatcher("BasketEmpty.jsp");

dispatcher.forward(request, response);

return;

}

List<Product> products = basket.getProducts();

String idString = request.getParameter("id");

if(StringUtils.isNotBlank(idString)){

int id = Integer.parseInt(idString);

for(Product p : products) {

if (p.getId() == id){

products.remove(p);

break;

}

}

}

if (basket == null || products.isEmpty()) {

RequestDispatcher dispatcher = request.getRequestDispatcher("BasketEmpty.jsp");

dispatcher.forward(request, response);

return;

}

RequestDispatcher dispatcher = request.getRequestDispatcher("Basket.jsp");

dispatcher.forward(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}

BackToProdutServlet.java

package maksym.web;

import maksym.db.ProductDAO;

import maksym.db.entity.Basket;

import maksym.db.entity.Product;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.util.List;

/\*\*

\* Servlet implementation class back

\*/

@WebServlet(value = "/back", name= "Basket")

public class BackToProductsServlet extends HttpServlet {

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse

\* response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html; charset=UTF-8");

int id = Integer.parseInt(request.getParameter("id"));

Product prod = null;

prod = ProductDAO.getInstance().getProductForId(id);

HttpSession session = request.getSession(true);

Basket basket = (Basket) session.getAttribute("basket");

if (basket == null) {

basket = new Basket();

session.setAttribute("basket", basket);

}

List<Product> products = basket.getProducts();

products.add(prod);

response.sendRedirect("Catalog?pet=" + prod.getPet\_id());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

\* response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

SearchResultServlet.java

package maksym.web;

import maksym.db.ProductDAO;

import maksym.db.entity.Product;

import org.apache.commons.lang3.StringUtils;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.List;

@WebServlet("/SearchResult")

public class SearchResultServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String requestString = request.getParameter("search");

if (StringUtils.isBlank(requestString)) {

RequestDispatcher dispatcher = request.getRequestDispatcher("ErrorSearchRequest.jsp");

dispatcher.forward(request, response);

} else {

List<Product> prod = null;

prod = ProductDAO.getInstance().getProductsForSearchRequest(requestString);

if (prod != null) {

request.setAttribute("products", prod);

RequestDispatcher dispatcher = request.getRequestDispatcher("Catalog.jsp");

dispatcher.forward(request, response);

} else {

RequestDispatcher dispatcher = request.getRequestDispatcher("ErrorSearchRequest.jsp");

dispatcher.forward(request, response);

}

}

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}

SuccessBuyServlet.java

package maksym.web;

import maksym.db.OrderDAO;

import maksym.db.entity.\*;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.sql.SQLException;

@WebServlet("/SuccessBuy")

public class SuccessBuyServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession();

Order order = new Order();

User user = (User)session.getAttribute("user");

order.setUser\_id(user.getId());

Basket basket = (Basket)session.getAttribute("basket");

/\* ListOrders listOrders = (ListOrders) session.getAttribute("order");

if (listOrders == null) {

listOrders = new ListOrders();

session.setAttribute("order", listOrders);

}

List<Order> orders = listOrders.getOrders();\*/

for (Product prod: basket.getProducts()) {

//manager.updateProductAmount(prod, 1);

order.setProdut\_id(prod.getId());

order.setCount\_product(1);

OrderDAO.getInstance().insertOrder(order);

// orders.add(order);

}

session.removeAttribute("basket");

response.sendRedirect("Gratitude.jsp");

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}

BuyOneProductServlet.java

package maksym.web;

import maksym.db.OrderDAO;

import maksym.db.entity.Basket;

import maksym.db.entity.Order;

import maksym.db.entity.Product;

import maksym.db.entity.User;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.sql.SQLException;

import java.util.List;

@WebServlet("/BuyOneProduct")

public class BuyOneProductServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int count = Integer.parseInt(request.getParameter("quantity"));

HttpSession session = request.getSession();

Order order = new Order();

User user = (User)session.getAttribute("user");

int prodID = Integer.parseInt(request.getParameter("id"));

order.setUser\_id(user.getId());

order.setProdut\_id(prodID);

order.setCount\_product(count);

OrderDAO.getInstance().insertOrder(order);

Basket basket = (Basket)session.getAttribute("basket");

List<Product> products = basket.getProducts();

for(Product p : products) {

if (p.getId() == prodID){

products.remove(p);

break;

}

}

if (basket == null || products.isEmpty()) {

response.sendRedirect("Gratitude.jsp");

return;

}

RequestDispatcher dispatcher = request.getRequestDispatcher("Basket.jsp");

dispatcher.forward(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}

CatalogServlet.java

package maksym.web;

import java.io.IOException;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import maksym.db.ProductDAO;

import maksym.db.entity.Product;

import org.apache.commons.lang3.StringUtils;

/\*\*

\* Servlet implementation class Catalog

\*/

@WebServlet("/Catalog")

public class CatalogServlet extends HttpServlet {

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String petString = request.getParameter("pet");

String typeString = request.getParameter("type\_product");

/\*if (StringUtils.isBlank(petString)) {

// TODO: 16.05.2020 redirect to error page

return;

}\*/

int pet = Integer.parseInt(petString);

List<Product> prod = null;

if (StringUtils.isNotBlank(typeString)) {

int type = Integer.parseInt(typeString);

prod = ProductDAO.getInstance().getProductsForTypeAndPetId(pet, type);

request.setAttribute("products", prod);

RequestDispatcher dispatcher = request.getRequestDispatcher("Catalog.jsp");

dispatcher.forward(request, response);

} else {

prod = ProductDAO.getInstance().getProductsForPetId(pet);

request.setAttribute("products", prod);

RequestDispatcher dispatcher = request.getRequestDispatcher("Catalog.jsp");

dispatcher.forward(request, response);

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

ContactServlet.java

package maksym.web;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@WebServlet("/Contacts")

public class ContactsServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

RequestDispatcher dispatcher = request.getRequestDispatcher("Contacts.jsp");

dispatcher.forward(request,response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}

EditProductServlet.java

package maksym.web;

import maksym.db.ProductDAO;

import maksym.db.entity.Product;

import org.apache.commons.lang3.StringUtils;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.regex.Pattern;

@WebServlet("/EditProduct")

public class EditProductServlet extends HttpServlet {

private static final String NUMBER\_REG\_EXP = "/d[0-9]+";

private static final String ONE\_NUM\_REG\_EXP = "/d[0-9]";

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

String prodId = request.getParameter("id");

Product prod = null;

prod = ProductDAO.getInstance().getProductForId(Integer.parseInt(prodId));

request.setAttribute("prod", prod);

RequestDispatcher dispatcher = request.getRequestDispatcher("EditProduct.jsp");

dispatcher.forward(request,response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

int id = Integer.parseInt(request.getParameter("id"));

String name = request.getParameter("name1");

String price = request.getParameter("price");

String description = request.getParameter("description");

String amount = request.getParameter("amount");

String weight = request.getParameter("weight");

String producer = request.getParameter("producer");

String type\_id = request.getParameter("type\_id");

String age = request.getParameter("age");

String breed = request.getParameter("breed");

String pet\_id = request.getParameter("pet\_id");

String photo\_link = request.getParameter("photo\_link");

Map<String, String> errors = new HashMap<>();

List<Product> allProd = ProductDAO.getInstance().getAllProducts();

Product prod = ProductDAO.getInstance().getProductForId(id);

for (Product p:allProd) {

if(p.getName().equals(name) && !prod.getName().equals(name) && p.getWeight().equals(weight) && !prod.getWeight().equals(weight)) {

errors.put("prodOrigin","Such a product already exists");

}

}

if(StringUtils.isBlank(name)){

errors.put("name", "Provide name product");

}

if(StringUtils.isBlank(price)){

errors.put("price", "Provide price product");

}

if(StringUtils.isBlank(description)){

errors.put("description", "Provide description product");

}

if(StringUtils.isBlank(amount)){

errors.put("amount", "Provide amount product");

}

if(StringUtils.isBlank(weight)){

errors.put("weight", "Provide weight product");

}

if(StringUtils.isBlank(producer)){

errors.put("producer", "Provide producer product");

}

if(StringUtils.isBlank(type\_id)){

errors.put("type\_id", "Provide type product");

}

if(StringUtils.isBlank(age)){

errors.put("age", "Provide age ");

}

if(StringUtils.isBlank(breed)){

errors.put("breed", "Provide breed ");

}

if(StringUtils.isBlank(pet\_id)){

errors.put("pet\_id", "Provide pet\_id product");

}

if(StringUtils.isBlank(photo\_link)){

errors.put("photo\_link", "Provide photo link product");

}

boolean foundPrice = Pattern.matches(NUMBER\_REG\_EXP, price);

if(foundPrice){

errors.put("pricePattern", "The entered price is incorrect");

}

boolean foundAmount = Pattern.matches(NUMBER\_REG\_EXP, amount);

if(foundAmount){

errors.put("amountPattern", "The entered amount is incorrect");

}

boolean foundWeight = Pattern.matches(NUMBER\_REG\_EXP, weight);

if(foundWeight){

errors.put("weightPattern", "The entered weight is incorrect");

}

boolean foundPet\_id = Pattern.matches(ONE\_NUM\_REG\_EXP, pet\_id);

if(foundPet\_id){

errors.put("pet\_idPattern", "The entered pet\_id is incorrect");

}

boolean foundType\_id = Pattern.matches(ONE\_NUM\_REG\_EXP, type\_id);

if(foundType\_id){

errors.put("type\_idPattern", "The entered type\_id is incorrect");

}

if (!errors.isEmpty()) {

RequestDispatcher dispatcher = request.getRequestDispatcher("AddProduct.jsp");

request.setAttribute("errors", errors);

dispatcher.forward(request, response);

return;

}

try {

ProductDAO.getInstance().updateProduct(name, price, description, amount, weight, producer,

type\_id, age, breed, pet\_id, photo\_link, id);

} catch (Exception e) {

response.sendRedirect("SomeWrong.html");

}

String direct = "AboutProduct?id="+id;

response.sendRedirect(direct);

}

}

LanguageServlet.java

package maksym.web;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

@WebServlet("/language")

public class LanguageServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String lang = request.getParameter("lang");

String path = request.getHeader("Referer");

HttpSession session = request.getSession();

session.setAttribute("lang", lang);

response.sendRedirect(path);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

}

ProfileServlet.java

package maksym.web;

import maksym.db.UserDAO;

import maksym.db.entity.User;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.\*;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.Objects;

@WebServlet("/Profile")

public class ProfileServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

HttpSession session = request.getSession();

String forwardPage;

String logout = request.getParameter("logout");

if("logout".equals(logout)) {

session.invalidate();

forwardPage = "Profile.jsp";

} else if (session == null || session.getAttribute("user") == null) {

forwardPage = "Profile.jsp";

} else {

User user = (User) session.getAttribute("user");

session.setAttribute("role", user.getRole());

forwardPage = "UserProfile.jsp";

}

RequestDispatcher dispatcher = request.getRequestDispatcher(forwardPage);

dispatcher.forward(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String email = request.getParameter("email");

String pass = request.getParameter("pass");

User user = UserDAO.getInstance().getUserForEmail(email);

Map<String, String> errors = new HashMap<>();

if (user == null) {

errors.put("error", "Entered Login or Password is incorrectly");

RequestDispatcher dispatcher = request.getRequestDispatcher("Profile.jsp");

request.setAttribute("errors", errors);

dispatcher.forward(request, response);

return;

}

if (!Objects.equals(user.getPass(), pass)) {

errors.put("error", "Entered Login or Password is incorrectly");

RequestDispatcher dispatcher = request.getRequestDispatcher("Profile.jsp");

request.setAttribute("errors", errors);

dispatcher.forward(request, response);

return;

}

HttpSession session = request.getSession(true);

session.setAttribute("user", user);

session.setAttribute("role", user.getRole());

RequestDispatcher dispatcher = request.getRequestDispatcher("UserProfile.jsp");

dispatcher.forward(request, response);

}

}

FilterPermission.java

package maksym.filter;

import javax.servlet.\*;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

public class FilterPermission implements Filter {

private boolean active = false;

@Override

public void init(FilterConfig config) {

String act = config.getInitParameter("active");

if (act != null) {

active = (act.equalsIgnoreCase("TRUE"));

}

}

@Override

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException {

if (active) {

if (!(request instanceof HttpServletRequest &&

response instanceof HttpServletResponse)) {

throw new ServletException("non-HTTP request or response");

}

HttpServletRequest httpServletRequest = (HttpServletRequest) request;

HttpServletResponse httpServletResponse = (HttpServletResponse) response;

HttpSession session = httpServletRequest.getSession();

String forwardPage;

if (session == null || session.getAttribute("user") == null) {

httpServletResponse.sendRedirect("Profile");

return;

}

chain.doFilter(request, response);

}

}

@Override

public void destroy() {

}

}

header.jsp

<%@ page contentType="text/html;charset=utf-8" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<c:if test="${not empty param.language}">

<c:set var="language" value="${param.language}" scope="session"/>

</c:if>

<fmt:setLocale value="${lang}" />

<fmt:setBundle basename="Bundles" />

<div id = "left-head" >

&nbsp

<form action="language" id="lang-flag" >

<input type="hidden" name="lang" value="ua">

<input type="image" name="image" src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/Flag\_UA.jpg" height="20" width="25">&nbsp

</form>

<form action="language" id="lang-flag">

<input type="hidden" name="lang" value="ru">

<input type="image" src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/Flag\_RU.jpg" height="20" width="25">&nbsp

</form>

<form action="language" id="lang-flag">

<input type="hidden" name="lang" value="en">

<input type="image" src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/Flag\_UK.jpg" height="20" width="25">

</form>

<br>

<br>

<a href="/pet\_shop-web/"><img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/%D0%97%D0%BD%D0%B0%D1%87%D0%BE%D0%BA.PNG" width="130%"></a>

</div>

<div id = "right-head" class = "shadow">

<img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D1%8C.png" height="20" width="5%"><a href="Profile"> <fmt:message key="header.profile"/></a>&nbsp

<img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/%D0%9A%D0%BE%D0%BD%D1%82%D0%B0%D0%BA%D1%82%D1%8B.png" height="20" width="5%"><a href="Contacts"><fmt:message key="header.contact"/></a>&nbsp

<img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/%D0%9A%D0%BE%D1%80%D0%B7%D0%B8%D0%BD%D0%B0.png" height="20" width="5%"><a href="basket"><fmt:message key="header.basket"/></a>

<br><br>

<a href="https://www.purina-felix.com.ua/"><img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/%D0%A0%D0%B5%D0%BA%D0%BB%D0%B0%D0%BC%D0%B0.jpg" height="150" width="100%"></a>

</div>

<div id = "middle-head">

<br><br><br><br><br><br>

<div class="wrapper">

<div class="container">

<form action= "SearchResult">

<input type="text" class="input" name= "search" placeholder="<fmt:message key="header.label.search"/> ">

<input type="submit" value="<fmt:message key="header.search"/>" class="close-btn">

</form>

</div>

</div>

</div>

<img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/OrangeLine.PNG" height="15" width="100%">

<nav>

<ul>

<li><a href="Catalog?pet=0"><fmt:message key="header.dog"/></a>

<ul>

<li><a href="Catalog?pet=0&type\_product=0"><fmt:message key="header.mainproduct"/></a></li>

<li><a href="Catalog?pet=0&type\_product=1"><fmt:message key="header.healthproducts"/></a></li>

<li><a href="Catalog?pet=0&type\_product=2"><fmt:message key="header.goodies"/></a></li>

<li><a href="Catalog?pet=0&type\_product=3"><fmt:message key="header.cannedfood"/></a></li>

</ul>

</li>

<li><a href="Catalog?pet=1"><fmt:message key="header.cat"/></a>

<ul>

<li><a href="Catalog?pet=1&type\_product=0"><fmt:message key="header.mainproduct"/></a></li>

<li><a href="Catalog?pet=1&type\_product=1"><fmt:message key="header.healthproducts"/></a></li>

<li><a href="Catalog?pet=1&type\_product=2"><fmt:message key="header.goodies"/></a></li>

<li><a href="Catalog?pet=1&type\_product=3"><fmt:message key="header.cannedfood"/></a></li>

</ul>

</li>

<li><a href="Catalog?pet=2"><fmt:message key="header.rodent"/></a>

<ul>

<li><a href="Catalog?pet=2&type\_product=0" ><fmt:message key="header.mainproduct"/></a></li>

<li><a href="Catalog?pet=2&type\_product=1"><fmt:message key="header.healthproducts"/></a></li>

<li><a href="Catalog?pet=2&type\_product=2"><fmt:message key="header.goodies"/></a></li>

<li><a href="Catalog?pet=2&type\_product=3"><fmt:message key="header.cannedfood"/></a></li>

</ul>

</li>

<li><a href="Catalog?pet=3"><fmt:message key="header.fish"/></a>

<ul>

<li><a href="Catalog?pet=3&type\_product=0"><fmt:message key="header.mainproduct"/></a></li>

<li><a href="Catalog?pet=3&type\_product=1"><fmt:message key="header.healthproducts"/></a></li>

<li><a href="Catalog?pet=3&type\_product=2"><fmt:message key="header.goodies"/></a></li>

<li><a href="Catalog?pet=3&type\_product=3"><fmt:message key="header.cannedfood"/></a></li>

</ul>

</li>

<li> <a href="Catalog?pet=4"><fmt:message key="header.bird"/></a>

<ul>

<li><a href="Catalog?pet=4&type\_product=0"><fmt:message key="header.mainproduct"/></a></li>

<li><a href="Catalog?pet=4&type\_product=1"><fmt:message key="header.healthproducts"/></a></li>

<li><a href="Catalog?pet=4&type\_product=2"><fmt:message key="header.goodies"/></a></li>

<li><a href="Catalog?pet=4&type\_product=3"><fmt:message key="header.cannedfood"/></a></li>

</ul>

</li>

</ul>

</nav>

<img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/OrangeLine.PNG" height="15" width="100%">

MainPage.jsp

<%@ page contentType="text/html;charset=utf-8" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<fmt:setLocale value="${lang}" />

<fmt:setBundle basename="Bundles" />

<html>

<head>

<title>Sweety Pet</title>

<META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-8">

<META HTTP-EQUIV="CONTENT-LANGUAGE" CONTENT="uk-UK">

<META NAME="DESCRIPTION" CONTENT="Онлайн-магазин для домашніх тварин">

<META NAME="KEYWORDS" CONTENT="Онлайн-магазин, товар, домашние, животные, собака, кот, кошка, рыба, кролик, попугай, хомяк, крыса, шиншила">

<LINK rel="stylesheet" href="styles.css">

<LINK rel="stylesheet" href="reset.css">

</head>

<body>

<jsp:include page="header.jsp" />

<img src="https://raw.githubusercontent.com/Maksym1996/Diplom/master/%D0%9D%D0%B0%20%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D1%83%D1%8E.PNG" width="100%">

</body>

</html>

Basket.java

package maksym.db.entity;

import java.util.ArrayList;

import java.util.List;

public class Basket {

private List<Product> products = new ArrayList<>();

public List<Product> getProducts() {

return products;

}

public void setProducts(List<Product> products) {

this.products = products;

}

}

UserRole.java

package maksym.db.entity;

public enum UserRole {

USER(0),

ADMIN(1);

private final int roleId;

UserRole(int roleId) {

this.roleId = roleId;

}

public int getRoleId() {

return roleId;

}

public static UserRole fromId(int id) {

for (UserRole role : values()) {

if (role.roleId == id) {

return role;

}

}

return null;

}

}

User.java

package maksym.db.entity;

/\*\*

\* User entity

\*

\* @author Maks

\*/

public class User {

private int id;

private String name;

private String surname;

private String pass;

private int role;

private String phone;

private String email;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getPass() {

return pass;

}

public void setPass(String pass) {

this.pass = pass;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public int getRole() {

return role;

}

public void setRole(int role) {

this.role = role;

}

@Override

public String toString() {

return "User{" +

"id=" + id +

", name='" + name + '\'' +

", surname='" + surname + '\'' +

", pass='" + pass + '\'' +

", role=" + role +

", phone='" + phone + '\'' +

", email='" + email + '\'' +

'}';

}

public static User createUser(String n, String sur, String phone, String mail, String pass, int role) {

User user = new User();

user.setName(n);

user.setSurname(sur);

user.setPhone(phone);

user.setEmail(mail);

user.setPass(pass);

user.setRole(role);

return user;

}

}

Product.java

package maksym.db.entity;

/\*\*

\* Product entity

\*

\* @author Maks

\*/

public class Product {

private int id;

private String name;

private String price;

private String description;

private String amount;

private String weight;

private String producer;

private int type\_id;

private String age;

private String breed;

private int pet\_id;

private String photo\_link;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPrice() {

return price;

}

public void setPrice(String price) {

this.price = price;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public String getAmount() {

return amount;

}

public void setAmount(String amount) {

this.amount = amount;

}

public String getWeight() {

return weight;

}

public void setWeight(String weight) {

this.weight = weight;

}

public String getProducer() {

return producer;

}

public void setProducer(String producer) {

this.producer = producer;

}

public int getType\_id() {

return type\_id;

}

public void setType\_id(int type\_id) {

this.type\_id = type\_id;

}

public String getAge() {

return age;

}

public void setAge(String age) {

this.age = age;

}

public String getBreed() {

return breed;

}

public void setBreed(String breed) {

this.breed = breed;

}

public int getPet\_id() {

return pet\_id;

}

public void setPet\_id(int pet\_id) {

this.pet\_id = pet\_id;

}

public String getPhoto\_link() {

return photo\_link;

}

public void setPhoto\_link(String photo\_link) {

this.photo\_link = photo\_link;

}

@Override

public String toString() {

return "Product [id=" + id + ", name=" + name + ", price=" + price + ", description=" + description

+ ", amount=" + amount + ", weight=" + weight + ", producer=" + producer + ", type\_id=" + type\_id

+ ", age=" + age + ", breed=" + breed + ", pet\_id=" + pet\_id + ", photo\_link=" + photo\_link + "]";

}

public static Product createProduct(String name, String price, String description, String amount, String weight, String producer,

int type\_id, String age, String breed, int pet\_id, String photo\_link) {

Product prod = new Product();

prod.setName(name);

prod.setPrice(price);

prod.setDescription(description);

prod.setAmount(amount);

prod.setWeight(weight);

prod.setProducer(producer);

prod.setType\_id(type\_id);

prod.setAge(age);

prod.setBreed(breed);

prod.setPet\_id(pet\_id);

prod.setPhoto\_link(photo\_link);

return prod;

}

}

Order.java

package maksym.db.entity;

/\*\*

\* Order entity

\*

\* @author Maks

\*/

public class Order {

private int id;

private int user\_id;

private int produt\_id;

private int count\_product;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getUser\_id() {

return user\_id;

}

public void setUser\_id(int user\_id) {

this.user\_id = user\_id;

}

public int getProdut\_id() {

return produt\_id;

}

public void setProdut\_id(int produt\_id) {

this.produt\_id = produt\_id;

}

public int getCount\_product() {

return count\_product;

}

public void setCount\_product(int count\_product) {

this.count\_product = count\_product;

}

@Override

public String toString() {

return "Order [id=" + id + ", user\_id=" + user\_id + ", produt\_id=" + produt\_id + ", count\_product="

+ count\_product + "]";

}

}