МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ МОДЕЛЮВАННЯ ТОРГІВЕЛЬНИХ ПЛОЩ»**

Виконав: студент 2 курсу, групи 8.1218-з

Спеціальності 121 інженерія програмного забезпечення

(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення

А.В. Пасічний

(ініціали та прізвище)

Керівник Декан математичного факультету, професор кафедри програмної інженерії, професор, д.т.н. Гоменюк С. І.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної математики доцент, д.т.н. Гребенюк С. М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Математичний
Кафедра програмної інженерії
Рівень вищої освіти магістр
Спеціальність 121 інженерія програмного забезпечення
Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

" ____ " _____ 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Пасічному Андрію Вікторовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка інформаційної системи підтримки моделювання торгівельних площ

керівник роботи Гоменюк Сергій Іванович, д.т.н., професор
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від "29" травня 2019 р. № 812-С

2. Строк подання студентом роботи _____

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Огляд технологічних підходів для розробки інформаційної системи.
3. Етапи розробки інформаційної системи підтримки моделювання торгівельних площ.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 29.05.2019

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	30.08.2019	Виконано
2.	Збір вихідних даних.	25.09.2019	Виконано
3.	Обробка методичних та теоретичних	15.10.2019	Виконано
	джерел.		
4.	Розробка першого і другого розділу.	24.11.2019	Виконано
5.	Розробка третього розділу.	14.12.2019	Виконано
6.	Оформлення і нормоконтроль	27.12.2019	Виконано
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи	10.01.2020	Виконано

Студент _____ А .В. Пасічний
(підпис) (ініціали та прізвище)

Керівник роботи _____ С. І. Гоменюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О. В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційної системи підтримки моделювання торгівельних площ»: 63 с., 20 рис., 15 джерел.

АВТОМАТИЗОВАНА ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, МОДЕЛЬ ДАНИХ, ШАБЛОН, МОДЕЛЬ, КОНТРОЛЕР, ФРЕЙМВОРК.

Об’єкт дослідження – інформаційна система.

Мета роботи – розробка автоматизованої інформаційної системи підтримки моделювання торгівельних площ.

Метод дослідження – порівняння, аналіз, описовий, структурний.

У кваліфікаційній роботі розглянуто автоматизовані інформаційні системи, які використовуються для планування торгівельних площ. Зростання високого попиту на планування торгівельних площ робить важливим розв'язок задач ефективного планування та моніторингу. Тому розробка автоматизованої інформаційної системи є актуальною задачею.

SUMMARY

Master’s Qualification Thesis "Development of an Information System for Support of Retail Space Modeling": 63 pages, 20 figures, 15 references.

AUTOMATED INFORMATION SYSTEM, DATABASE, DATA MODEL, TEMPLATE, MODEL, CONTROLLER, FRAMEWORK.

The object – the information system.

The aim of the study is to develop an automated information system for support of retail space modeling.

Methods of research are analytical, comparative, analysis and searching of information.

The qualification work deals with the automated information systems used to plan retail space. Increasing demand for retail space planning makes it important to address the effective planning and monitoring challenges. Therefore, the development of an automated information system is a main task.

ЗМІСТ

ВСТУП

Проектування – початкова стадія створення нового торговельного об’єкта. Основна мета проектування полягає у плануванні процесу створення торговельного об’єкта, організації його зовнішньої території та внутрішнього простору, забезпеченні функціональних взаємозв’язків між окремими приміщеннями, прийнятті рішень щодо організації торговельно технологічного процесу і робочих місць відповідно до чинних норм та правил із забезпеченням психофізіологічного комфорту працівникам та споживачам, дизайнерської виразності та цілісного будівельно-архітектурного представлення об’єкта.

Основні завдання, що виникають під час проектування магазинів наступні: здійснювати планування площі торгівельної зали магазину; проектувати неторгівельну площу магазину; обґрунтувати вибір типу торгівельного центру для конкретної території; здійснювати зонування площі торгівельного центру та розподіляти потоки покупців; визначити особливості створення концепціїї дизайн-проектів магазинів; здійснювати розрахунки ефективності використання площі для розміщення торговельно-технологічного обладнання; обґрунтовувати вибір типу майбутнього магазину; розробляти об'ємно-планувальні рішення щодо приміщень магазинів; проводити техніко-економічні розрахунки площі основних приміщень магазинів; обґрунтовувати розміщення груп товарів на торговельно-технологічному обладнання; розробляти планограми розміщення торговельно-технологічного обладнання; визначати вибір стелажного обладнання для модернізації складу.

Нинішні умови торгівлі в супермаркетах складають чималу конкуренцію один одному. Серед магазинів, які просто переповнені однотипними товарами боротьбу виграють ті торговельні мережі, які роблять акцент на високу якість позиціювання товарів. Крім того, важливо

підвищувати привабливість торговельного залу для потенційних покупців, стимулювати збут шляхом регулярного оновлення інтер'єру, планування і т.д.

Подібні маркетингові заходи, однак, вимагають чимало інвестицій в торгове обладнання для торгових точок, але, завдяки доведеної високої ефективності та раціонального управління, їх окупність набагато вище витрачених коштів. Вклавши гроші правильно, а саме, придбавши якісне професійне обладнання для магазину, супермаркету, Ви створите атмосферу, яка, так чи інакше, приваблюватиме покупців на здійснення покупок.

У супермаркетах, а точніше в торгових залах магазинів будь-якого виду обслуговування, широко використовується різноманітні торгові меблі (камери для зберігання речей, шафи, бокси) і торгові стелажі (стелаж прямий, стелаж кондитерський, стелаж овочевий, стелаж хлібний, стелаж настінний, стелаж з сітчастими кошиками, стелаж прямий на стяжках, стелаж книжковий і т.д.).

Стелаж – це універсальна конструкція, яка дозволяє представити товар покупцеві. Торговельні та складські стелажі відіграють одну з головних ролей в оснащенні супермаркетів і магазинів торговим обладнанням і для здійснення складської логістики, тому вибираючи їх потрібно ретельно спланувати розстановку для раціонального використання всієї торговельної площі або складу. Крім основного призначення - розміщення і розташування товарів, стелажі зможуть послужити доповненням в загальній атмосфері магазину.

Всі конструкції в торговому залі (стелаж прямий, стелаж кутовий внутрішній, стелаж прямий перфорований, стелаж прямий двосторонній) повинні бути компактними, таким чином, вони забезпечать простір торгової площі, при цьому задіють найменшу територію для розміщення всього товару.

1 ОГЛЯД ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ ПЛАНУВАННЯ ТОРГОВИХ ПЛОЩ

1.1 Огляд стану проблеми моделювання торгівельних площ

У зв'язку зі стрімким зростанням кількості супермаркетів і магазинів меншого формату актуальним стає питання правильного і грамотного розташування торгового обладнання для забезпечення зручності і комфорту покупців, що безпосередньо відбивається на продажах.

Мета грамотного моделювання торгових площ – максимізувати продажу на кожен квадратний метр виділеної торгової площі.

Для здійснення моделювання можливе використання систем автоматизованого проектування таких як Autocad, Archicad і ін. Проте істотним мінусом цих систем є ліцензійні відрахування, що тягне за собою додаткові витрати при відкритті магазину, а також необхідна певна кваліфікація і досвід як користувачів даних програм.

У даній роботі метою є реалізація системи планування торгових площ на етапі розробки концепції магазину і орієнтовного розташування торгового обладнання, з певними спрощеннями для зручності користувача інформаційної системи. Це так званий «ескізний проект» для створення якого не потрібні знання програм проектування.

1.2 Зонування магазину

Попереднє зонування площі торгового об'єкта є необхідною складовою частиною концепції магазину. Подальше планування розміщення торгового обладнання в торговому залі є, мабуть, однією з найбільш важливих і складних складових технологічного проектування, оскільки від цього

залежить збільшення обороту і прибутку магазину, і навпаки – помилки тут здатні привести до втраченого прибутку або навіть до збитків в процесі експлуатації магазину.

Процес технологічного проектування супермаркету, безумовно, вимагає спеціальних знань.

При плануванні торгового залу необхідно враховувати всі торгово-технологічні процеси, що відбуваються в магазині. Тому процес технологічної планування торгового залу нерозривно пов'язаний з проектуванням складських, допоміжних і виробничих приміщень торгового об'єкта.

Необхідно визначити концепцію магазину. У ній відображено повний асортиментний перелік, групи товарів, перелік відділів, який може бути, наприклад таким: Овочі-фрукти; М'ясо і м'ясопродукти; Молочні продукти; Гастрономія, кулінарія; риба; Заморожені продукти; Бакалія; Хлібобулочні і кондитерські вироби; Соки-води; Спиртні напої; Супутні товари.

Крім відділів при зонуванні враховуються такі об'єкти і параметри, як зона входу-виходу, касова зона, шляхи подачі товару в торговий зал, напрямки купівельних потоків, переваги покупців і ін.

Вважається, що різні зони в торговому приміщенні не рівноцінні по віддачі. По ходу свого руху покупець в першій чверті залу здійснює близько 40% покупок, у другій – близько 30%, в третій – 20% і залишилася – 10%.

Так само є статистика, що 80% покупців обходить всі точки продажів, розташовані по периметру, і тільки 50% покупців обходять внутрішні ряди. І хоча ці цифри досить умовні і застосовні до торговельних об'єктів середньої площі, їх необхідно враховувати.

Для грамотного зонування усі товари, відповідно до переваг потенційних покупців умовно ділять на: Товари повсякденного попиту; Товари періодичного попиту; Товари імпульсного попиту.

Товари повсякденного попиту – це товари, покупка яких планується під час кожного візиту в магазин. До них можна віднести: молочні і м'ясні продукти, хліб, овочі.

Товари періодичного попиту – це товари, покупка яких планується один раз в декілька візитів. До них відносяться: товари бакалійної групи, делікатеси, кондитерські вироби, побутова хімія.

Товари імпульсного попиту – це товари, покупка яких в даний візит не планувалася покупцем. Наприклад, господарські товари, друкована продукція, батарейки, іграшки, косметика, компакт диски і т.п.

Як правило, в зоні інтенсивної торгівлі (першої частини торгового залу) розміщують найбільш прибуткові товари. Для вирівнювання віддачі з іншою торговельною площею товари повсякденного попиту в основному розміщують у другій і третій частині залу переважно по периметру. На шляху до них мають у своєму розпорядженні супутні товари, стимулюючи імпульсні покупки.

Але слід враховувати що в супермаркеті з великою площею концентрація відділів з товарами повсякденного попиту в кінці маршруту руху покупця може негативно позначитися на продажах товарів, що знаходяться на початку і середині шляху руху.

1.3 Планування торгового залу і розміщення торгового обладнання

Після проведення зонування визначаються з методом торгівлі у відділах: самообслуговування; торгівля через прилавок; змішаний метод.

Після цього визначається, який вид торгового обладнання буде застосовуватися для різних груп товарів всередині відділу. На підставі даних попереднього аналізу визначається необхідна площа експозиції того чи іншого торгового і холодильного устаткування. Потім відбувається розстановка обладнання у відділах згідно з попередньо проведеним

зонування. При розміщенні обладнання в кожному відділі крім основних також передбачаються додаткові місця продажу для товарів імпульсного попиту. Також передбачаються місця для установки мобільних стендів та проведення промоакцій.

Найчастіше при організації торгового простору використовується лінійне планування. Ряди торгового обладнання в центрі залу розташовані паралельно напрямку входу в торговий зал і касовим терміналам. При такій організації торгова площа використовується найбільш раціонально, а покупець легко орієнтується в торговому залі і відчуває себе комфортно. З точки зору організації системи внутрішньої безпеки торговельного об'єкта та захисту від крадіжок, це планування так само має незаперечні переваги.

При розміщенні обладнання в торговому залі зазвичай використовують принцип «подібне - подібному». Для чого це робиться.

По-перше, покупцеві легше орієнтуватися в торговому залі і вибирати товари.

По-друге, при компактно розташованому холодильному обладнанні менше втрати «холоду», який іде у торговий зал. По-третє, торговий зал виглядає більш естетично. Крім відділів супермаркету дуже важливо правильно спроектувати такі зони: зона входу-виходу; зона входу в торговий зал; розрахунково-касова зона.

Послідовність прийняття рішень щодо організації мерчандайзингу в торговому залі магазину починається з визначення асортиментних позицій, рівня цін і необхідного торговельного запасу, які пов'язуються з наявністю торгової площі, іміджем магазину, його цільовою аудиторією.

Потім вирішується питання планування торгового залу магазину.

Планування торговельного залу магазину визначає поділ торгової площі на секції і маршрут руху покупців по території магазину. Стелажі і товари на них розташовуються виходячи з правила: чим краще видно товар, тим більше шансів, що його куплять. Розміщення товарів в місцях найбільш

інтенсивного потоку покупців підвищує ймовірність покупки, у тому числі і незапланованою.

З маркетингової точки зору планування торгового залу повинна бути такою, щоб покупець міг побачити товар і у нього виникло бажання пройти по всьому торговому залу, а підприємець міг створити в магазині приємну атмосферу. При плануванні торгового залу магазину, розстановці устаткування важливо враховувати поведінку покупця всередині магазину. Підкоряючись природним рефлексам, що покупець буде рухатися справа наліво.

Не кожен покупець обходить весь магазин. Товар, розташований в далеких кутах, може не потрапити в поле його зору.

При організації мерчандайзингу в магазині намагаються розташувати товар таким чином, щоб він був повніше охоплений купівельним потоком. Для цього необхідно впливати на напрямок потоку і вміти розгорнути його назад. Щоб здійснити це завдання, іноді слід:

а) скоротити проходи;

б) зменшити освітлення в правій частині, а в лівій підсилити;

в) в дальній частині торгового залу розмістити товари-"зірки" або "зазивні" товари.

Для маленьких магазинів важливо, в який бік відчиняються двері. Якщо вона відкривається від себе направо, то люди підуть наліво, якщо відкривається наліво, то люди підуть направо. Даний фактор необхідно враховувати, так як більша площа залу може виявитися неохопленою.

Залежно від системи розстановки обладнання використовують різні види технологічної планування торгового залу:

а) лінійну (решітка);

б) боксову ("трек", "петля");

в) змішану;

г) виставкову;

д) вільне (довільну).

Лінійне планування торговельного залу передбачає схеми розміщення товарів і проходів для покупців у вигляді паралельних ліній. Відповідно розподіляються і лінії торгового обладнання. При цьому лінія вузла розрахунку розташовується перпендикулярно. Таке планування використовується у магазинах самообслуговування.

По відношенню до розміщення ліній обладнання в торговому залі схеми можуть бути поздовжніми, поперечними і змішаними, варіанти яких представлені нижче.

"Грати" з лінійним поздовжнім розміщенням торгового обладнання: стелажі з товарами розташовуються переважно перпендикулярно входить в зал купівельному потоку (рис. 1.1).

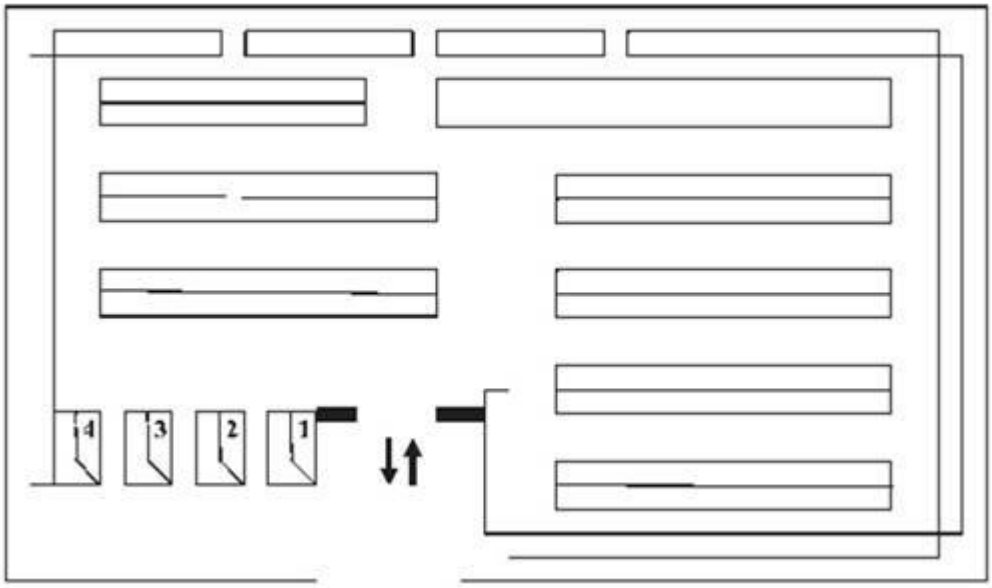

Рисунок 1.1 – "Грати" з лінійним поздовжнім розміщенням торгівельного обладнання

"Грати" з лінійним поперечним розміщенням обладнання. При такому плануванні стелажі з товарами розташовуються переважно паралельно входить купівельному потоку (рис. 1.2).

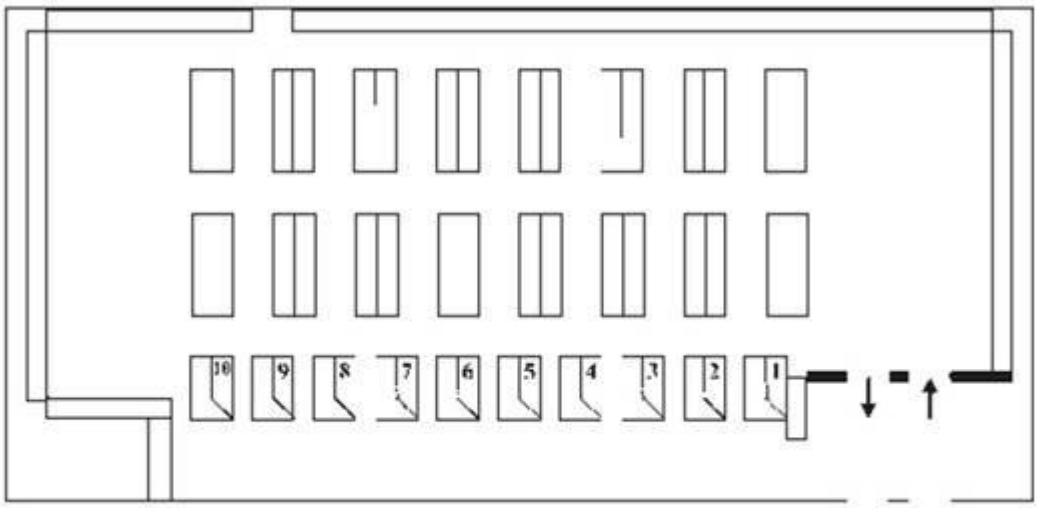

Рисунок 1.2 – "Грати" з лінійним поперечним розміщенням торгівельного обладнання

"Решітка" зі змішаним лінійним розміщенням обладнання. Дана планування передбачає поєднання в торговому залі поздовжнього і поперечного розміщення (рис. 1.3).

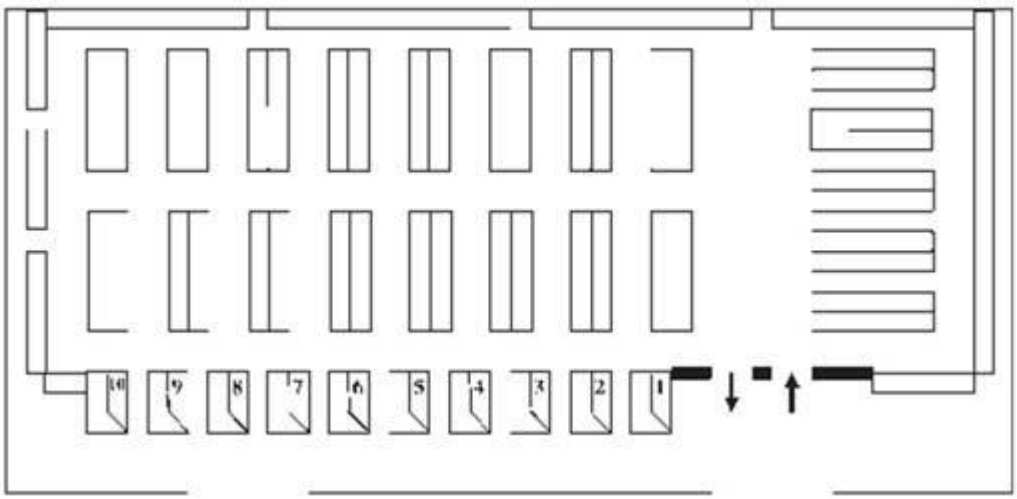

Рисунок 1.3 – "Решітка" зі змішаним лінійним розміщенням торгівельного обладнання

Змішане планування передбачає різноманітні комбінації лінійної і боксової планувань. Розстановка устаткування в магазині оптимізується в залежності від геометрії торгового простору і структури торгових секцій (рис. 1.4).

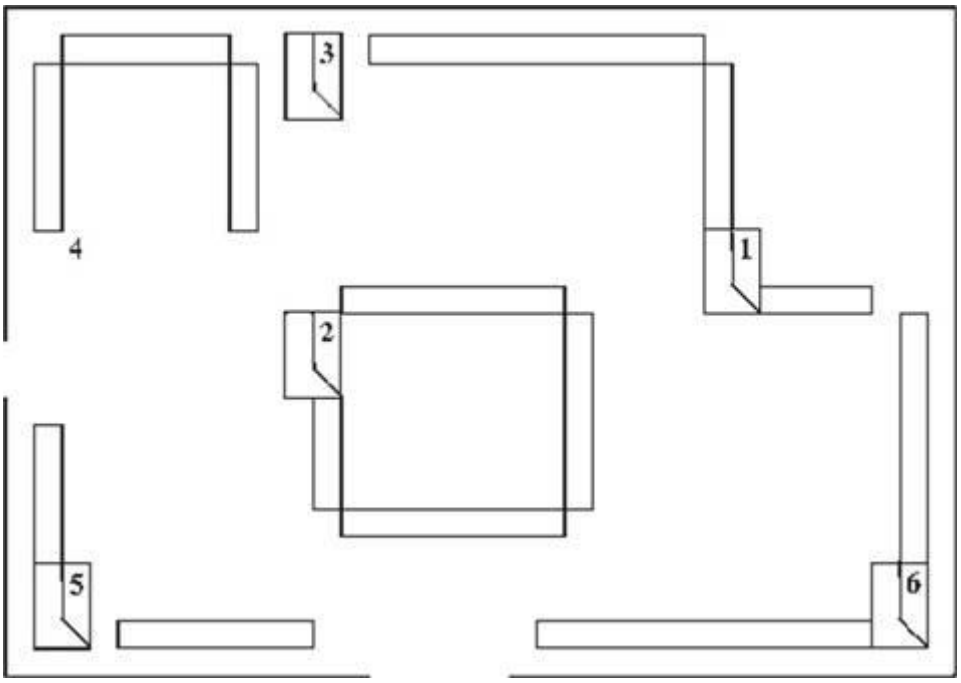

Рисунок 1.4 – Змішане планування

Виставкове планування торгового залу застосовується при продажу товарів за зразками. Великогабаритні товари розміщуються на різному, як правило, нестандартному обладнанні і утворюють демонстраційні композиції.

Вільне планування передбачає розміщення встаткування без певної геометричної системи відповідно до поставленої замовником завданням і можливостями, які надає конкретний торговий зал і продемонструє дизайнер-проектувальник.

Лінійне планування торговельного залу дозволяє чітко формувати потоки руху покупців, створює кращі умови для групування і розміщення товарів, забезпечує кращий перегляд торгового залу. При лінійному плануванні покупці найбільш повно сприймають інформацію про товари. Зміною довжини ліній регулюється зосередження покупців на різних ділянках торгового залу. До переваг лінійного планування також слід віднести найбільш ефективне використання площі торгового залу. У торгових залах прямокутної конфігурації шириною від 7 до 12 м доцільно

застосовувати лінійну розстановку з поздовжнім розміщенням обладнання. У магазинах, ширина торгового залу яких перевищує 24 м, більш ефективною виявляється змішане планування обладнання. Комбінація лінійного, поздовжнього та лінійно-поперечного розміщення потребує врахування багатьох чинників і певного мистецтва.

Рекомендується використовувати довжину ліній острівних гірок не більше 20 м, оскільки велика їх довжина призведе до надмірного збільшення потоків покупців у магазині, що ускладнить їх рух в торговому залі. За розміщення торгового обладнання слід враховувати розташування наявних в приміщенні колон – вони повинні перебувати в межах ліній і не заважати руху покупців.

При вільному плануванні торгове обладнання та інвентар розміщуються в довільному порядку. Напрямку руху покупців нічим не обмежені, люди можуть вільно переходити з однієї ділянки залу на інший, підходити до стелажів, прилавків, вітрин, оглядати товар в будь-якій послідовності. Більшість покупців віддають перевагу саме вільне планування, так як їм подобається відчувати себе невимушено.

1.4 Загальний огляд фреймворків для створення веб-додатків

WEB фреймворк – це каркас, призначений для створення динамічних веб-сайтів, мережевих додатків, сервісів або ресурсів. Він спрощує розробку і позбавляє від необхідності написання рутинного коду. Багато фреймворків спрощують доступ до баз даних, полегшують розробку інтерфейсу, а також зменшують дублювання коду.

Виділяють п’ять типів веб-фреймворків: Requestbased, Component-based, Hybrid, Meta and RIA-based. Request-based – фреймворки, які безпосередньо обробляють вхідні запити. Збереження стану відбувається за рахунок серверних сесій. Приклади: Django, Ruby on Rails, Struts, Grails.

Component-based – фреймворки, які абстрагують обробку запитів всередині стандартних компонентів і самостійно стежать за станом. Дані каркаси мають багато спільного зі стандартними програмними графічними інтерфейсами. Приклади: JSF, Tapestry, Wicket.

Hybrid-based – фреймворки, які комбінують Request-based та Component-based фреймворки, беручи під свій контроль всі дані і логічний потік в заснованої на запиті моделі. Розробники мають повний контроль над URL, формами, параметрами, cookies і pathinfos. Однак замість того, щоб відобразити дії і контролери безпосередньо до запиту, гібридні фреймворки забезпечують об'єктну модель компонентів, яка поводиться тотожно в багатьох різних ситуаціях, таких як окремі сторінки, перервані запити, подібні порталу фрагменти сторінок та інтегровані віджети. Компоненти можуть розподілятися окремо і ефективно інтегруватися в інші проекти. Приклади: RIFE.

Meta-based – фреймворки, що мають ряд базових інтерфейсів для загального обслуговування і основу, яка легко розширюється з метою інтегрування компонентів і служб. Приклад: Keel. RIA-based (фреймворки для розробки Rich Internet Applications (RIA) – фреймворки, що служать для розробки повноцінних додатків, які запускаються всередині браузера. Приклад: Flex.

Найбільш поширеними є Request-based і Component-based веб-фреймворки. Зібравши і проаналізувавши інформацію, було виділено такі характерні компоненти web фреймворків:

а) шаблонизатор. Відповідає за незалежність верстки від програмного коду;

б) роутер. Розпізнає URL, за яким відбулося звернення до сервера;

в) модуль доступу до бази даних;

г) модуль кешування. Прискорює завантаження сторінок;

д) модуль безпеки. Аутентифікація і авторизація користувачів;

е) файли конфігурації.

WEB фреймворки також можуть керувати сесіями, вести логи, спрощувати використання технології Ajax та ін.

1.5 Переваги та недоліки використання фреймворків для розробки Web-додатків

Під час вибору фреймворку можуть виникнути певні труднощі, пов’язані з визначенням завдань, які він може виконувати, та його призначення. Якщо для створення сайту потрібно знайти зручний і простий в освоєнні фреймворк, то необхідно ретельно підійти до питання його вибору та зважити всі «за» і «проти».

Розглянемо загальні переваги використання фреймворку:

а) гнучкість і масштабування – завжди має гнучке рішення нестандартних завдань і можливість далі розширення функціоналу шляхом підключення сторонніх бібліотек або окремих класів; ефективне використання ресурсів сервера;

б) використання підходу модель-вид-контролер (MVC) суттєво розширює функціонування та гнучкість проекту, так як використовується під час проектування та розробки програмного забезпечення; фреймворки написані розробниками для розробників, що дозволяє мати на виході прекрасно написаний код і своєчасне виправлення помилок;

в) наявність детальної документації з використання фреймворку;

г) безпека – забираються всі проломи в безпеці, практично немає вузьких місць для SQL-ін’єкцій; фреймворк дозволяє сконцентруватися на вирішенні архітектурних завдань, а не базових, як прирозробці без його застосування;

д) якість матеріалу на виході.

Головна переваги фреймворків, те що вони якнайкраще підходять для створення масштабованих і унікальних сайтів. Жоден масштабний проект не

розроблений на готовій CMS – вони для цього не призначені. Майже всі унікальні web-додатки розробляються з використанням фреймворків. Web-проект, розроблений за допомогою фреймворку, розвивається динамічно. При зміні вимог змінюється і сайт, для створення нового розділу або внесення новизни в дизайні, достатньо змінити окремий модуль. Замінити окремий блок (модуль), створити новий розділ або внести новизну в дизайні.

Недоліки застосування фреймворку досить умовні і незначні порівняно з перевагами:

а) важко обслуговувати – якщо проект створював один розробник, а потім з якихось причин він пропадає або просто відмовляється супроводжувати створений ним проект, то його подальший розвиток і обслуговування ставає більш складним питанням і часто не вигідним заняттям;

б) ціна розробки – вартість стандартного сайту зробленого на фреймворку з нуля буде дорожче, ніж на готової CMS, тому що часу на розробку витрачатися в кілька разів більше.багато коду не використовується і лежить мертвим вантажем в проекті;

в) складність в освоєнні;

г) відсутність готових модулів і компонентів, які міг би встановити клієнт, в мережі Інтернет немає ані безкоштовних, ані платних. Всі доробки необхідно замовляти у розробників.

1.6 Огляд Back-end фреймворків

Yii – це універсальний фреймворк, який може бути задіяний у всіх типах веб додатків. Завдяки його компонентній структурі та відмінній підтримці кешування, фреймворк особливо підходить для розробки таких великих проектів як портали, форуми, CMS, магазини чи RESTful-додатки.

Можливості Yii :

а) низький поріг входження;

б) парадигма Модель-вид-контролер;

в) інтерфейси DAO і ActiveRecord для роботи з базами даних (PDO);

г) підтримка інтернаціоналізації;

д) кешування сторінок і окремих фрагментів;

е) перехоплення і обробка помилок;

ж) введення і валідація форм;

з) аутентифікація і авторизація;

и) використання AJAX і інтеграція з jQuery;

к) генерація базового PHP-коду для CRUD-операцій (скаффолдинг);

л) підтримка тем оформлення для їх легкої зміни;

м) можливість підключення сторонніх бібліотек;

н) міграції бази даних;

о) автоматичне тестування;

п) підтримка REST.

Недоліки:

а) AR не підтримує AR запити;

б) не дуже хороший зв'язок в БД багато-до-багатьох(плагін CAdvancedArBehavior випраляє цей недолік).

Symfony2 – PHP фреймворк, який має велику бібліотеку класів, написаний на PHP 5. Архітектура має корисні компоненти та інструменти, призначені для створення складних веб-додатків. Symfony — це вільний каркас, написаний на PHP5, який використовує патерн Model-View-Controller (MVC). Symfony пропонує швидку розробку і керування веб-додатками, що дозволяє легко вирішувати рутинні завдання веб-програміста. Symfony безкоштовний і доступний під ліцензією MIT.

Перваги :

а) підтримує безліч баз даних (MySQL, PostgreSQL, SQLite, або будь-яка інша PDO-сумісна СУБД);

б) вбудовані класи для роботи з email;

в) гнучка система шаблонів у поданні;

г) вбудований кодогенератор;

д) підтримка французького спонсора Sensio;

е) дуже гнучкий;

ж) висока продуктивність;

Недоліки:

а) складний в освоєнні;

б) підходить тільки для великих проектів;

в) відсутність російської документації;

г) остання версія 5.3 вимагає PHP;

д) немає вбудованої ORM;

е) немає російськомовного співтовариства.

Zend framework – це PHP– фреймворк, що створений і підтримується компанією Zend, співробітники якої є безпосередніми авторами мови PHP. Тому він є послідовником традицій і цінностей PHP – базується на простоті, об'єктноорієнтованих принципах, дружній ліцензії і ретельно тестованому коді із застосуванням Agile методів.

Можливості:

а) всі компоненти повністю орієнтовані на PHP 5 та E_STRICT – сумісні;

б) вбудований генератор коду;

в) архітектура «використовуй тільки те, що необхідно» з мінімальними залежностями компонентів;

г) використовує легко розширюваний шаблон проектування MVC, підтримує макети і PHP-скрипти подання за замовчуванням;

д) підтримує безліч різних баз даних, включаючи MariaDB, MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite, and Informix Dynamic Server;

е) спеціальні класи для створення, відправлення, отримання email з допомогою mbox, Maildir, POP3 та IMAP4;

ж) гнучка система кешування з підтримкою безлічі сховищ.

Недоліки:

а) досить великий;

б) дуже повільний без кешування;

в) занадто складна архітектура, без глибокого розуміння шаблонів проектування складний в освоєнні;

г) слабке російськомовне співтовариство;

д) досить повільна швидкість розвитку;

е) остання версія вимагає PHP 5.2;

ж) відсутній ORM.

CakePHP є фреймворком для PHP, який надає розширену архітектуру для розробки, обслуговування і розгортання web-додатків. Він використовує відомий шаблон проектування MVC, як і в об'єктно-реляційних фреймворкаху.

Основною парадигмою CakePHP є збільшення продуктивності розробки, і, як наслідок, допомога програмісту у вигляді зменшенні обсягу написання коду. Спочатку створювався як клон популярного Ruby on Rails, тому багато ідей були запозичені саме звідти.

Можливості:

а) сумісність з PHP4 (до версії 1.3 включно) і PHP5;

б) диспетчер URL із застосуванням регулярних виразів;

в) генерація всього коду за схемою бази даних (потрібно дотримуватися стандарту іменування стовпців);

г) перевірка форм;

д) компоненти для авторизації, обмеження доступу (ACL), управління сесіями, cookies, подання деревоподібної інформації (у вигляді Nested Sets);

е) хелпери(компоненти) для генерації та заповнення форм, поділу на сторінки (paginate), управління кешем, JavaScript(в тому числі і AJAX);

ж) механізм інтернаціоналізації;

з) генерація SQL-запитів, в тому числі для таблиць з відносинами один до багатьох і багато до багатьох, ORM;

и) Scaffolding і генерація CRUD-сторінок для сутностей, Router:mapResources з Put Delete Get Post;

к) автогенератор коду Bake;

л) міграції;

м) консольна інтеграція, клас Shell і завдання Task;

н) плагіни (як окремі програми), компоненти і поведінки;

о) підтримка Simple Test.

Недоліки:

а) низька продуктивність;

б) слабка документація;

в) нестійкість до CSRF-атак;

г) немає російськомовної спільноти;

д) складний в розумінні ACL;

е) строгі угоди з іменування;

ж) низька швидкість розвитку.

Ruby on Rails – об'єктно-орієнтований програмний каркас (фреймворк) для створення веб-додатків, написаний на мові програмування Ruby. Ruby on Rails надає каркас модель-вид-контролер (Model-View-Controller) для веб-додатків, а також забезпечує їхню інтеграцію з веб-сервером і сервером бази даних.

Можливості:

а) лаконічний і простий синтаксис, часто зустрічається вплив Ада, Eiffel і Python;

б) може обробляти винятки в стилі Java і Python;

в) дозволяє переозначувати оператори, які насправді є методами;

г) простий та послідовний синтаксис;

д) можливість прямих системних викликів;

е) миттєва поява змін під час розробки;

ж) відсутність стадії компіляції;

з) повністю об'єктно-орієнтована мова програмування. Всі дані в Ruby є об'єктами в розумінні Smalltalk. Єдиний виняток — керівні конструкції, які в Ruby на відміну від Smalltalk не є об'єктами;

и) містить автоматичний прибиральник сміття. Він працює для всіх об'єктів Ruby, в тому числі і для зовнішніх бібліотек;

к) створення розширень для Ruby на C дуже просте завдяки збору сміття, та нескладного і зручного API;

л) підтримує цикли з повною прив'язкою до змінних;

м) підтримує блок коду (код взятий в { … } або do … end). Блоки можуть використовуватись в методах або перетворюватись в цикли;

н) змінні динамічно типізовані;

о) безпосередньо в мові Ruby реалізовано багато шаблонів програмування. Так, наприклад, «одинак» (англ. singleton) може бути реалізований додаванням потрібних методів одному конкретному об'єктові;

п) може динамічно завантажувати розширення, якщо це дозволяє операційна система;

р) перенесена на багато платформ. Мова розроблялася на GNU/Linux, але працює на багатьох версіях Unix, DOS, Microsoft Windows (частково, Win32), Mac OS, BeOS,OS/2 і т.д.

Недоліки:

а) низька швидкість роботи;

б) проекти на ньому досить громіздкі;

в) складний debug коду;

г) філософія розвитку багато в чому залежить від "магії" і "припущень";

д) погана російськомовна документація.

ASP.Net MVC – фреймворк для створення веб-додатків, який реалізує шаблон Model-view-controller. Даний фреймворк розроблений компанією Microsoft.

Фреймворк ASP.NET MVC надає наступні можливості.

Поділ завдань програми (логіка введення, бізнес-логіка і логіка інтерфейсу), широкі можливості тестування і розробки на основі тестування. Всі основні контракти платформи MVC засновані на інтерфейсі і підлягають тестуванню за допомогою макетів об'єкта, які імітують поведінку реальних об'єктів програми. Додаток можна піддавати модульному тестуванню без запуску контролерів у процесі ASP.NET, що прискорює тестування і робить його більш гнучким. Для тестування можливе використання будь-якої платформи, сумісної з .NET Framework.

Платформа має можливість розширення і доповнення . Компоненти платформи ASP.NET MVC можна легко замінити або налаштувати. Розробник може підключати власний механізм уявлень, політику маршрутизації URL-адрес, серіалізацію параметрів методів, дій та інші компоненти. Платформа ASP.NET MVC також підтримує використання моделей контейнера впровадження залежності (DI) та інверсії елемента керування (IOC). Модель впровадження залежності дозволяє впроваджувати об'єкти в клас, а не очікувати створення об'єкта самим класом. Модель інверсії елемента управління вказує на те, що якщо один об'єкт вимагає інший об'єкт, то перші об'єкти мають отримати другий об'єкт із зовнішнього джерела (наприклад, з файлу конфігурації), що значно полегшує тестування.

Розширена підтримка маршрутизації ASP.NET. Цей потужний компонент зіставлення URL-адрес дозволяє створювати додатки із зрозумілими URL-адресами, які можна використовувати в пошуку. URL-адреси не повинні містити розширення імен файлів і призначені для підтримки шаблонів іменування URL-адрес, що забезпечують адресацію, оптимізовану для пошукових систем (SEO) і для передачі репрезентативного стану (REST).

Підтримка використання розмітки в існуючих файлах сторінок ASP.NET (ASPX), елементів управління (ASCX) і головних сторінок (MASTER) як шаблонів представлень. Разом з платформою ASP.NET MVC

можна використовувати існуючі функції ASP.NET, наприклад, вкладені головні сторінки, вбудовані вирази , декларативні серверні елементи управління, шаблони, прив'язку даних, локалізацію і т. д.

Недоліки:

а) високий поріг входження;

б) відсутність механізму зберігання стану;

в) складність створення бібліотек компонентів.

Кожен з вищерозглянутих фреймворків має свої переваги та недоліки, тому їх вибір необхідно здійснювати з урахуванням проекту, який буде створюватися. Це є головним критерієм вибору, адже жоден із фреймворків не є універсальним рішенням для будь-якого проекту.

Висновки до розділу 1

Отже, вибір конкретного виду планування залежить від величини торговельного залу магазину, його конфігурації, методів продажу й асортименту поданих у магазині товарів. Особливу увагу звертають на ефективність використання торгової площі – кожен метр корисної площі повинен приносити магазину максимальний прибуток.

2 ПРОЕКТУВАННЯ МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура Модель–Вид–Контролер

Для як найшвидшого розуміння суті архітектури MVC, необхідно розглянути фундаментальну концепцію web-додатків. Архітектура Модель-Вид-Контролер дуже важлива в контексті фреймворків, оскільки вона є типовою для більшості web-додатків.

Model View Controller – схема використання декількох шаблонів проектування, за допомогою яких модель, інтерфейс і взаємодія з користувачем розділені на три окремі компоненти таким чином, щоб модифікація одного з компонентів мала мінімальний вплив на інші . Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області.

Архітектурний шаблон Модель-Вид-Контролер (MVC) поділяє програму на три частини. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них; вигляд (View) відповідальний за відображення цих даних користувачеві; контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача і повідомляє про зміни компонента Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних.

Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, в яких відображається

інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле). Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об'єктам, відповідальним за відображення даних.

Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

Рисунок 2.1 – Архітектура MVC

Також існує архітектура Модель-Вид-Представник (Model-View-Presenter) (MVP) та архітектура N–Tier. Відмінності між MVC і MVP полягають в тому, як дані надходять з моделі у вид. Так, у MVC вони надходять з моделі у вид, а в MVP проходять через Вид. Очевидною перевагою при використанні концепції MVC є чіткий поділ логіки подання (інтерфейсу користувача) і логіки програми.

В даний час набирає популярності застосування об'єктно-реляційного відображення (Object Relational Mapping – ORM) для реалізації моделі webдодатку. ORM у широкому сенсі – це правило або метод, що зв'язує об'єкти мови програмування з відносинами реляційної бази даних і трансформує операції над об'єктами у відповідні операції над відносинами. У цьому сенсі в будь-якому проекті, розробка якого ведеться за допомогою об'єктно-орієнтованої мови, а дані зберігаються в реляційній СУБД, ця трансформація якимось чином вже здійснюється .

2.2 Концептуальна та логічна модель даних

Концептуальне проектування – збір, аналіз і редагування вимог до даних. Для цього здійснюються наступні заходи:

а) обстеження предметної області, вивчення її інформаційної структури;

б) виявлення всіх фрагментів, кожний з яких характеризується призначеним для користувача представленням, інформаційними об'єктами і зв'язками між ними, процесами над інформаційними об'єктами;

в) моделювання і інтеграція всіх представлень.

Логічне проектування – перетворення вимог до даних в структури даних.

На виході отримуємо СУБД-орієнтовану структуру бази даних і специфікації прикладних програм. На цьому етапі часто моделюють бази даних стосовно різних СУБД і проводять порівняльний аналіз моделей.

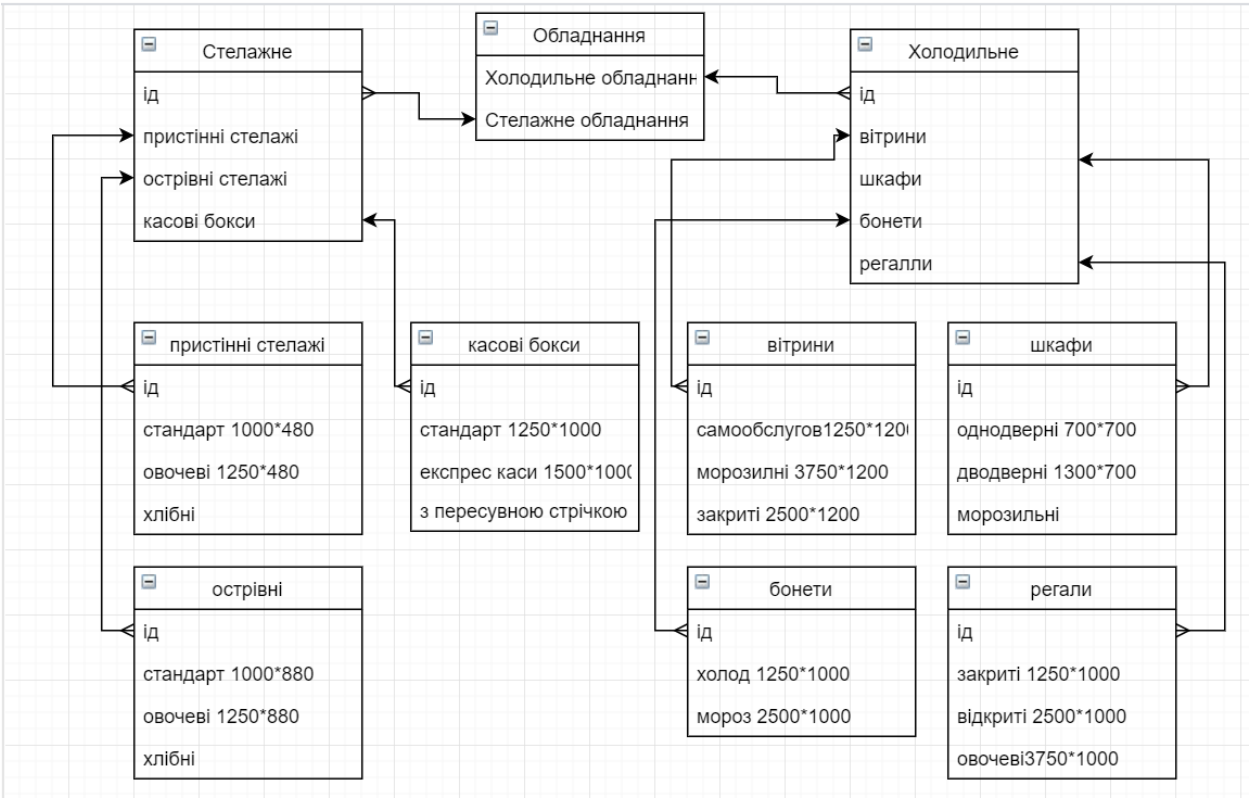

Рисунок 2.2 – Концептуальна модель даних

Для досягнення наочності в представлені концептуальної і зовнішніх схем бази даних були розроблені і зараз активно використовуються графічні моделі. Графічні семантичні моделі надають можливість формального і разом з тим наочного опису предметної області. Частини предметної області, що відповідають об’єктам, властивостям і зв’язкам зображуються у вигляді діаграм.

Фізичне проектування – визначення особливостей зберігання даних, методів доступу, фізичне представлення бази даних в комп'ютері і т.д. На цьому рівні описується, як інформація зберігається в базі даних.

2.3 Проектування структури сайту

Розроблена структура Web-сайту впливає на його навігаційну схему, від цього залежить як будь-який користувач зможе по ньому пересуватися і

отримувати доступ до запропонованої інформації. Одним з найважливіших факторів є простота і зручність навігації. Від них залежить відвідуваність Web-сайту. Відвідувачі сайту повинні мати можливість швидко і просто перейти на будь-яку сторінку Web-сайту, в тому числі на головну. Загальне уявлення про сайт, його структуру та принцип роботи починають формуватися вже на цьому етапі. Відразу ж потрібно чітко продумати назви майбутніх розділів сайту, заголовки до сторінок, продумати та визначити переходи між ними, щоб розміщення інформації на сайті було точним і логічним. Погано структуровані сайти можуть погано впливати на відвідуваність сайту, а користувачі можуть не знайти потрібну інформацію під час їх перегляду.

Деякі особливості проектування структури Web-сайту:

а) необхідно дотримуватись одноманітності елементів. При розробці структури слід визначити ієрархію об'єктів. Наприклад, кожен розділ включає в себе певні підрозділи і т.д.;

б) слід уникати створення подібних сторінок. Якщо однотипна інформація може бути розміщена на одній сторінці, не слід розробляти для неї окрему;

в) не потрібно створювати розділи, які дублюють один одного;

г) потрібно передбачити простий і швидкий доступ до всіх найбільш важливих розділів з усіх сторінок сайту;

д) проектування структури сайту має включати складання карти основних сторінок. Обов'язково треба продумати які розділи найбільш необхідні для користувача і виключити ті, які будуть абсолютно марними;

е) потрібно придумати свій ідеальний варіант структури, повністю відповідний до тематики ресурсу.

Для того, щоб розробка структури веб-сайту була максимально комфортною, її можна відобразити за допомогою комп'ютерних програм, таких як MS Visio, Power Point або звичайний Paint. Таким чином, можна в графічному вигляді скласти набір необхідних сторінок, а також продумати їх

зв'язки один з одним. Також можна скористатися звичайною ручкою і папером, щоб намалювати структуру ресурсу. Варто зазначити, що для подібних цілей існує спеціалізоване ПО, але його використання не доцільно при створенні невеликих сайтів.

2.4 Загальний огляд фреймворка Yii2

Yii – це високопродуктивний компонентний PHP фреймворк, призначений для швидкої розробки сучасних веб-додатків.

Завдяки його компонентній структурі і відмінною підтримкою кешування, фреймворк особливо підходить для розробки таких великих проектів, як портали, форуми, CMS, магазини або RESTful-додатки.

Як і багато інших PHP фреймворки, для організації коду Yii використовує архітектурний патерн MVC (Model-View-Controller).

Yii дотримується філософії простого і елегантного коду, не намагаючись ускладнювати дизайн тільки заради проходження будь-яким шаблонами проектування.

Yii включає в себе перевірені і добре зарекомендували себе можливості, такі як ActiveRecord для реляційних і NoSQL баз даних, підтримку REST API, багаторівневе кешування і інші.

Yii відмінно розширюється. Можна налаштувати або замінити практично будь-яку частину основного коду. Використовуючи архітектуру розширень, легко ділитися кодом або використовувати код спільноти.

Одна з головних цілей Yii – продуктивність.

Yii підтримується і розвивається сильною командою і великим співтовариством розробників, які їй допомагають. Автори фреймворка стежать за тенденціями веб-розробки і розвитком інших проектів. Найбільш підходящі можливості і кращі практики регулярно впроваджуються в фреймворк у вигляді простих і елегантних інтерфейсів.

2.5 Послідовність роботи фреймворка Yii. Модель-представлення-контролер (MVC)

Yii використовує шаблон проектування Модель-Представлення-Контролер (MVC, Model-View-Controller), який широко застосовується в веб-програмуванні.

MVC призначений для поділу бізнес-логіки і призначеного для користувача інтерфейсу, щоб розробники могли легко змінювати окремі частини програми, не зачіпаючи інші. В архітектурі MVC модель надає дані і правила бізнес-логіки, уявлення відповідає за користувальницький інтерфейс (наприклад, текст, поля введення), а контролер забезпечує взаємодію між моделлю і представленням.

Крім цього, Yii використовує фронт-контролер, званий додатком (application), який інкапсулює контекст обробки запиту. Додаток збирає інформацію про запит і передає її для подальшої обробки відповідного контролера.

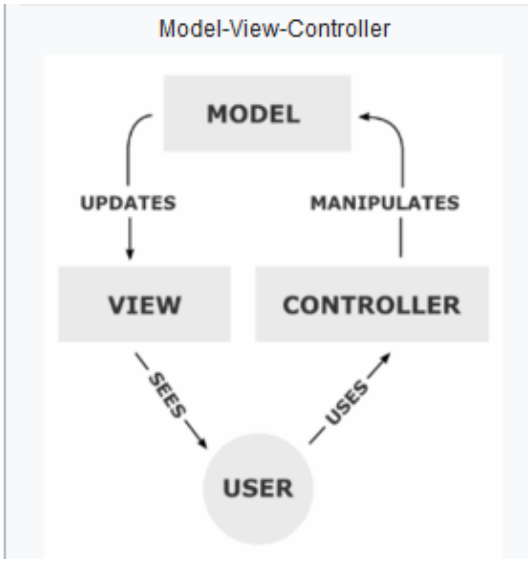

Рисунок 2.3 – Схематичне зображення MVC

Цей шаблон передбачає поділ системи на три взаємопов'язані частини:

а) модель даних – надає дані і реагує на команди контролера, змінюючи свій стан;

б) вигляд (інтерфейс користувача) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі;

в) модуль керування інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача, схематично зображено на рисунку 2.3.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу підвищується можливість повторного використання коду. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах або з різних точок зору. Зокрема, виконуються наступні завдання:

а) до однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми;

б) не торкаючись реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних) – для цього досить використовувати інший контролер;

в) ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому

можливо добитися того, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть обізнані про те, яке представлення буде використовуватися.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель зберігає ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер вносить зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.6 Додатки фреймворка Yii2

Yii додатки організовані згідно шаблону проектування модель-уявлення-контролер (MVC). Моделі представляють собою дані, бізнес логіку і бізнес правила; уявлення відповідають за відображення інформації, в тому числі і на основі даних, отриманих з моделей; контролери приймають вхідні дані від користувача і перетворюють їх в зрозумілий для моделей формат і команди, а також відповідають за відображення потрібного уявлення.

Крім MVC, Yii додатки також мають наступні сутності:

а) вхідні скрипти: це PHP скрипти, які доступні безпосередньо кінцевому користувачеві програми. Вони відповідальні за запуск і обробку вхідного запиту;

б) додатки – це глобально доступні об'єкти, які здійснюють коректну роботу різних компонентів програми та їх координацію для обробки запиту;

в) компоненти програми – це об'єкти, зареєстровані в додатку і надають різні можливості для обробки поточного запиту;

г) модулі – це самодостатні пакети, які включають в себе повністю всі кошти для MVC. Додаток може бути організовано за допомогою декількох модулів;

д) фільтри – це код, який повинен бути виконаний до і після обробки запиту контролерами;

е) віджети – це об'єкти, які можуть включати в себе уявлення. Вони можуть містити різну логіку і бути використані в різних уявленнях.

Нижче на діаграмі представлена структурна схема програми (рисунок 2.4).

Загальний процес створення веб-додатків з використанням фреймворку. Має такі етапи:

а) створення структури директорій. Утиліта yiic може бути використана для того, щоб прискорити цей процес;

б) конфігурація додатка шляхом модифікації файлу конфігурації програми. Цей етап також може запропонувати написання деяких компонентів програми (наприклад, компонента управління користувачами);

в) створення класу моделі для кожного використовуваного типу даних. Для автоматичної генерації всіх необхідних моделей Active Record можна скористатися інструментом Gii;

г) створення класу контролера для кожного типу користувальницького запиту. Класифікація призначених для користувача запитів залежить від поточних вимог. У загальному випадку, якщо клас моделі використовується користувачем, повинен існувати відповідний клас контролера. Утиліта Gii також може автоматизувати цей процес;

д) створення дій і уявлень. Саме тут і відбувається основна робота;

е) конфігурація необхідних фільтрів для дій в класах контролерів;

ж) створення тем оформлення при необхідності;

з) переклад повідомлень в разі, коли потрібна локалізація додатків;

и) виявлення даних і уявлень, які можуть бути закешовані, і застосування відповідних технік кешування;

к) налаштування продуктивності і розгортання.

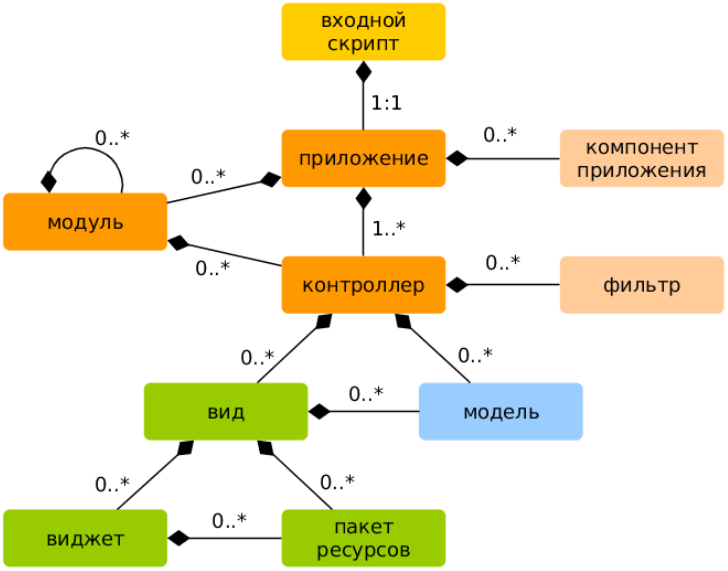

Рисунок 2.4 – Структурна схема MVC

Висновки до розділу 2

Таким чином, проаналізувавши всі існуючі види структури інформаційних систем, можна зробити висновок, що деревовидна структура підходить якнайбільше.

Yii – це універсальний фреймворк і може бути задіяний у всіх типах веб-додатків. Завдяки його компонентній структурі і відмінною підтримкою кешування, фреймворк особливо підходить для розробки таких великих проектів, як портали, форуми, CMS, магазини або RESTful-додатки.

Також для створення дизайну сайту найкраще підійде мінімалістичний дизайн, у якому оформлення сторінок буде зроблено у приємній колірній гаммі. На сторінках буде розроблена зручна навігація, досить великі і контрастні шрифти. Призначення кожного елементу сторінки сайту має бути таким, щоб відвідувачі не витрачали свій час на спроби зрозуміти, як виконати ту чи іншу дію.

3 ЕТАПИ РОЗРОБКИ WEB – РЕСУРСУ

3.1 Робота фреймворка Yii2 з базою даних

Побудовані поверх PDO, Yii DAO (об'єкти доступу до даних) забезпечують об'єктно-орієнтована API для доступу до реляційних баз даних. Це основа для інших, більш просунутих, методів доступу до баз даних, включаючи будівник запитів і active record.

При використанні Yii DAO в основному використовується чистий SQL і масиви PHP. Як результат, це найефективніший спосіб доступу до баз даних. Проте, так як синтаксис SQL може відрізнятися для різних баз даних, використовуючи Yii DAO потрібно докласти додаткових зусиль, щоб зробити додаток не залежних від конкретної бази даних.

Yii DAO з коробки підтримує наступні бази даних:

а) MySQL;

б) MariaDB;

в) SQLite;

г) PostgreSQL;

д) CUBRID;

е) Oracle;

ж) MSSQL.

Для створення проекту Yii2 з використанням composer, а саме: composer create-project – prefer-dist yiisoft/yii2-app-basic relax.loc.

Для створення міграції для проекту з існуючої бази даних необхідно використати наступне: php yii migrate / create init_tables. Перед цим необхідно налаштувати зв'язок з БД у файлі config / db.php:

```
<?php
return [
    'class' => 'yii\db\Connection',
```

```
    'dsn' => 'mysql:host=localhost;dbname=relax',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
];
```

Можна налаштувати декілька компонентів підключення, якщо в додатку використовується кілька баз даних.

Під час налаштування підключення, обов'язково потрібно вказувати ім'я джерела даних через параметр dsn. Формат DSN відрізняється для різних баз даних.

3.2 Створення головної сторінки

Головна сторінка сайту (доменне ім'я, index.html) повинна бути оптимізована за словосполученням, які найбільшою мірою важливі для всього змісту сайту. Саме головна сторінка – перший кандидат на потрапляння в топ пошукових систем.

Для посторінкового відображення матеріалів на домашній сторінці сайту в класі-контролері веб-додатки SiteController потрібно змінити метод actionIndex:

```
public function actionIndex()
{
    $query = Stellage::find()->orderBy('name');
    $pages = new Pagination(['totalCount' => $query->count(), 'pageSize' => 2]);
    $stellage =$query->offset($pages->offset)->limit($pages->limit)->all();
    return $this->render('index', compact('stellage', 'pages'));
}
```

В змінну $query витягуються все з моделі Stellages (стелажі) з сортуванням по назві. Потім в змінній $pages створюється компонент посторінкового виведення із зазначенням кількості записів і кількості виведених на сторінку. Далі в змінній $stellages задається підмножина записів із зсувом і лімітом з компонента сторінок. Вкінці, при рендері сторінки index за допомогою методу compact в уявлення домашньої сторінки передаються змінні stellages і pages.

У представленні домашньої сторінки (файл view / site / index.php) здійснюється виведення коротких відомостей про стелажі. Заголовки є посиланнями на детальне уявлення.

Нижче наведено програмний код для створення головної сторінки:

```
<?php
use yii\helpers\Html;
use yii\widgets\LinkPager;
$this->title = 'Стелажі';
?>
<div class="site-index">
    <div class="jumbotron">
        <h1>Стелажі</h1>
    </div>
    <div class="body-content">
        <div class="row">
            <div class="col-lg-12">
                <?php if (!empty($stellages)): ?>
                    <?php foreach ($stellages as $rr): ?>
                        <div class="panel panel-default">
                            <div class="panel-heading">
                                <h2class="panel-title"><a
href="<?=\yii\helpers\Url::to(['stellage/view', 'id' => $rr->id]); ?>"><?=$rr->name;
?></a></h2><?=$rr->city->nameWithParent; ?>
```

```
                    </div>
                    <div class="panel-body"><?=$rr->description; ?></div>
                    <div class="panel-footer">
                        Типи стелажів:
                        <?php
                        $types = $rr->types;
                        foreach ($types as $type): ?>
                            <?=$type['name']; ?>;
                        <?php endforeach; ?></div>
                </div>
            <?php endforeach; ?>
        <?php endif; ?>
    </div>
</div>
<?=LinkPager::widget(['pagination' => $pages]) ?>
</div>
</div>
```

В результаті отримаємо головну сторінку інформаційної системи, як зображено на рисунку 3.1

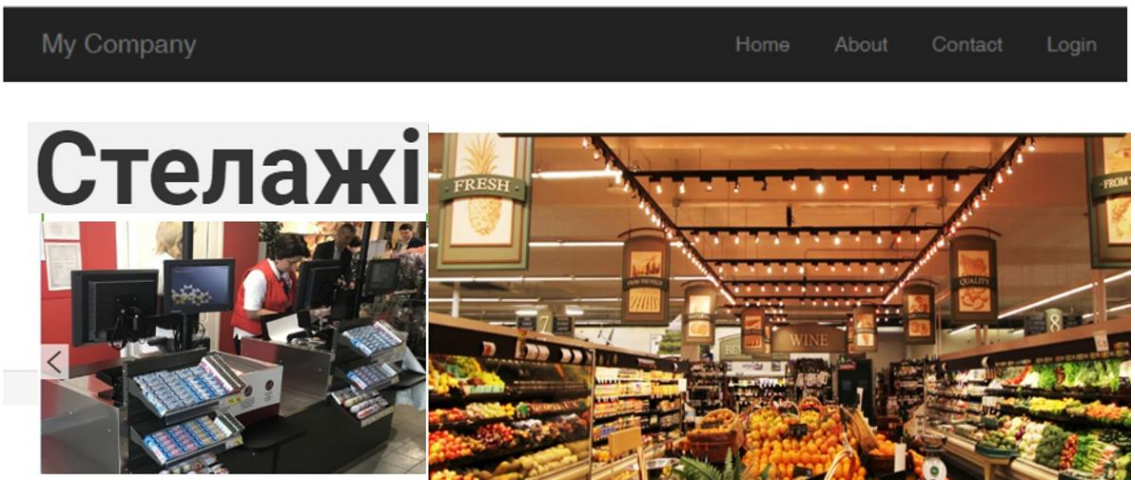

Рисунок 3.1 – Головна сторінка інформаційної системи

3.3 Створення підрозділів інформаційної системи

Для відображення підрозділу «Тип стелажів» необхідно створити для нього модель та представлення. Для створення моделі використаємо наступний програмний код:

```
class StellType extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'stell_type';
    }
    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['name'], 'required'],
            [['description'], 'string'],
            [['name'], 'string', 'max' => 50],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'Ідентифікатор',
            'name' => 'Назва',
            'description' => 'Опис',
        ];
    }
```

```
    public function getStellage()
    {
        return $this->hasMany(Stellage::className(), ['type_id' => 'id']);
    }
}
```

Далі необхідно створити представлення за допомогою такого коду:

```
$this->title = 'Типи стелажів';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="stell-type-index">
    <h1><?= Html::encode($this->title) ?></h1>
    <?php Pjax::begin(); ?>
    <?php // echo $this->render('_search', ['model' => $searchModel]); ?>
    <p>
        <?= Html::a('Додати новий тип стелажів', ['create'], ['class' => 'btn btn-success']) ?>
    </p>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'filterModel' => $searchModel,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],
            'id',
            'name',
            'description:ntext',
            ['class' => 'yii\grid\ActionColumn'],
        ],
    ]); ?>
    <?php Pjax::end(); ?>
</div>
```

В результаті отримаємо сторінку на нашому сайті, як зображено на рисунку 3.2, з різними типами стелажів, наприклад, холодильні, для овочів, для непродовольчих товарів, стелажі для хліба. Кожен тип має свій ідентифікатор, у створеній базі даних, має свою назву та загальний опис. Також кожен тип можна як просто переглядати, так редагувати і видаляти.

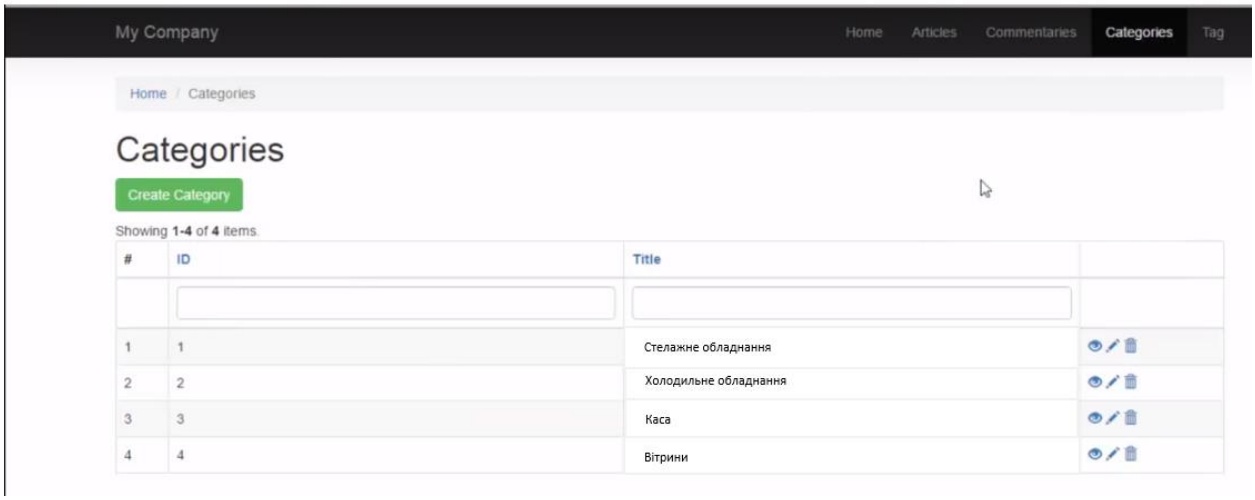

Рисунок 3.2 – Створення підрозділу «Категорії»

Для перегляду необхідно використати наступний код:

```
$this->title = $model->name;
$this->params['breadcrumbs'][] = ['label' => 'Типи стелажів', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="stell-type-view">
    <h1><?= Html::encode($this->title) ?></h1>
    <p>
        <?= Html::a('Редагувати', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>
        <?= Html::a('Видалити', ['delete', 'id' => $model->id], [
            'class' => 'btn btn-danger',
            'data' => [
```

```
                'confirm' => 'Ви підтверджуєте видалення запису?',
                'method' => 'post',
            ],
        ]) ?>
    </p>
    <?= DetailView::widget([
        'model' => $model,
        'attributes' => [
            'id',
            'name',
            'description:ntext',
        ],
    ]) ?>
</div>
```

В результаті отримаємо вікно для редагування та видалення записів (рис. 3.3).

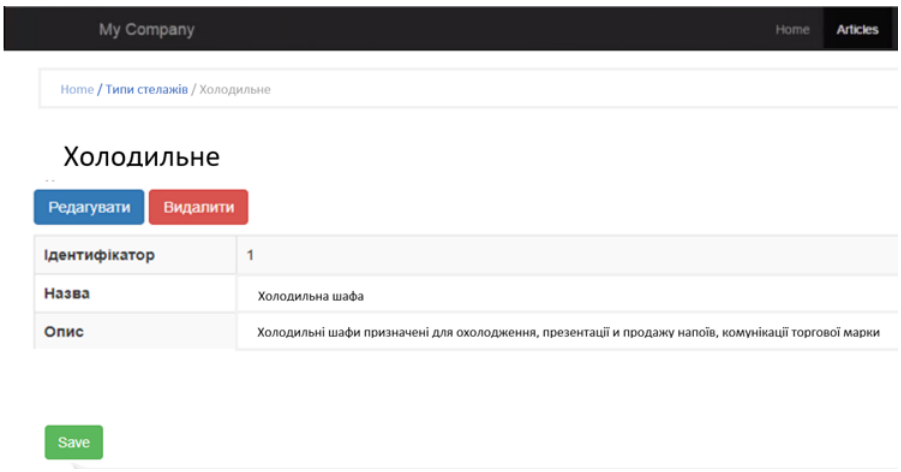

Рисунок 3.3 – Редагування та видалення записів

Для створення форми оновлення типів стелаів використаємо такий код:

```
$this->title = 'Форма оновлення типу стелажів: ' . $model->name;
$this->params['breadcrumbs'][] = ['label' => 'Типи стелажів', 'url' => ['index']];
```

```
$this->params['breadcrumbs'][] = ['label' => $model->name, 'url' => ['view', 'id' => $model->id]];
$this->params['breadcrumbs'][] = 'Update';
?>
<div class="stell-type-update">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
</div>
```

В результаті отримаємо форму, як зображено на рисунку 3.4.

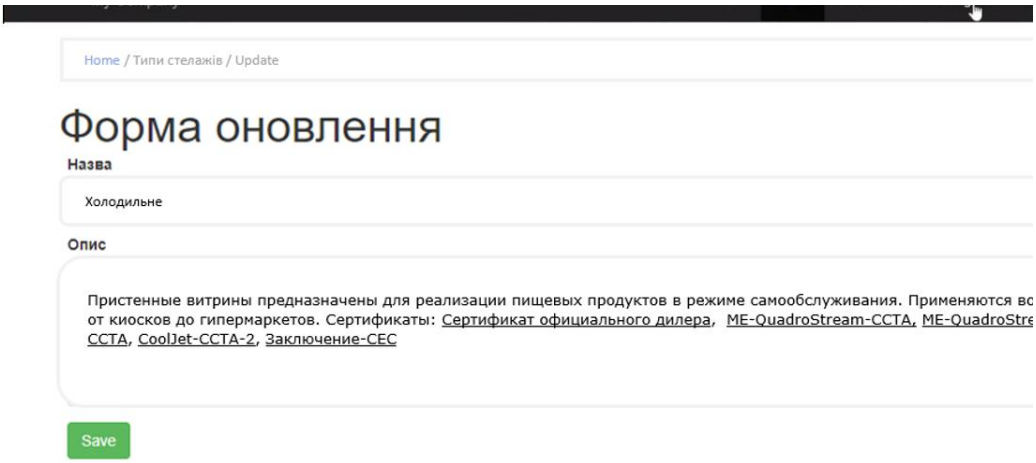

Рисунок 3.4 – Форма оновлення типів стелажів

Для створення форми додавання нового типу стелажів використаємо наступний програмний код:

```
$this->title = 'Форма додавання нового типу стелажів';
$this->params['breadcrumbs'][] = ['label' => 'Типи стелажів', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="stell-type-create">
    <h1><?= Html::encode($this->title) ?></h1>
```

```
    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
</div>
```

В результаті отримаємо форму, як зображено на рисунку 3.5:

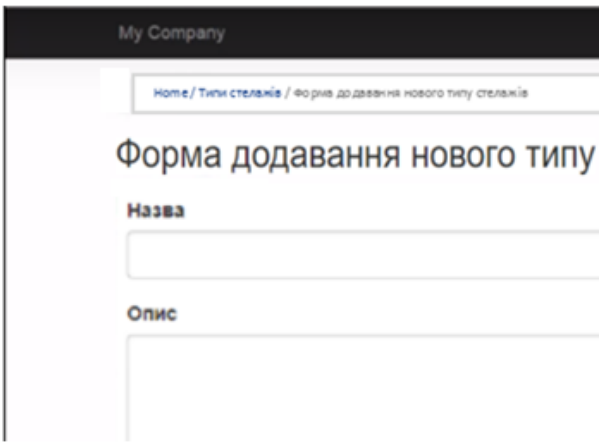

Рисунок 3.5 – Форма додавання нового типу стелажів

3.4 Створення довідників стелажів

Для створення довідників стелажів спочатку необхідно створити модель:

```
class Stell extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'stell';
    }
    /**
    public function rules()
    {
        return [
```

```
            [['name', 'type_id'], 'required'],
            [['type_id', 'parent_id'], 'integer'],
            [['description'], 'string'],
            [['name'], 'string', 'max' => 50],
            [['type_id'], 'exist', 'skipOnError' => true, 'targetClass' => StellType::className(), 'targetAttribute' => ['type_id' => 'id']],
            [['parent_id'], 'exist', 'skipOnError' => true, 'targetClass' => Stell::className(), 'targetAttribute' => ['parent_id' => 'id']],
        ];
    }
    public function attributeLabels()
    {
        return [
            'id' => 'Ідентифікатор стелажів',
            'name' => 'Назва стелажів',
            'type_id' => 'Тип стелажів',
            'description' => 'Опис стелажів',
            'parent_id' => 'Каталог',
            'fullName' => 'Назва каталогу',
        ];
    }
    public function getType()
    {
        return $this->hasOne(StellType::className(), ['id' => 'type_id']);
    }
    public function getParent()
    {
        return $this->hasOne(Stell::className(), ['id' => 'parent_id']);
    }
    public function getStells()
```

```
        {
            return $this->hasMany(Stell::className(), ['parent_id' => 'id']);
        }
        public function getStallages()
        {
            return $this->hasMany(Stellages::className(), ['city_id' => 'id']);
        }
        public function getFullName()
        {
            return $this->name . ', ' . mb_strtolower($this->type->name, 'UTF-8');
        }
    }
```

Для того, щоб записи були розміщені у вигляді дерева необхідно використати наступне:

```
    <?php
    use yii\helpers\Html;
    use yii\grid\GridView;
    use yii\widgets\Pjax;
    use leandrogehlen\treegrid\TreeGrid;
    use yii\helpers\ArrayHelper;
    $this->title = 'Каталог';
    $this->params['breadcrumbs'][] = $this->title;
    ?>
    <div class="stell-index">
        <h1><?= Html::encode($this->title) ?></h1>
        <?php Pjax::begin(); ?>
        <?php // echo $this->render('_search', ['model' => $searchModel]); ?>
        <p>
            <?= Html::a('Додати новий запис', ['create'], ['class' => 'btn btn-
success']) ?>
```

```
</p>
<?= TreeGrid::widget([
    'dataProvider' => $dataProvider,
    'keyColumnName' => 'id',
    'parentColumnName' => 'parent_id',
    'columns' => [
        'fullName',
        ['class' => 'yii\grid\ActionColumn']
    ]
]); ?>
<?php Pjax::end(); ?>
```

В результаті отримаємо сторінку інформаційної системи з повним переліком стелажів, розподілених по категоріях у формі дерева, як зображено на рисунку 3.6.

Рисунок 3.6 – Перегляд каталогу

Для додавання нової одиниці каталогу використаємо наступний програмний код:

```
$this->title = 'Форма додавання нового стелажу';
$this->params['breadcrumbs'][] = ['label' => 'Каталог', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
```

```
?>
<div class="stell-create">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
</div>
```

В результаті отримаємо форму додавання нової одиниці стелажу, як зображено на рисунку 3.7.

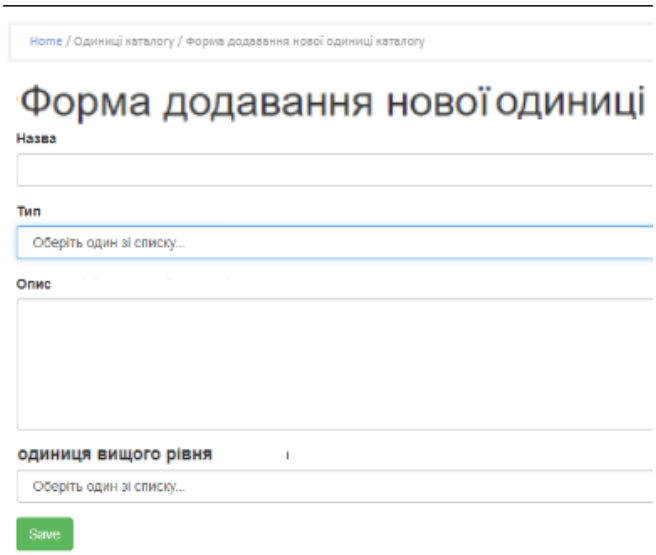

Рисунок 3.7 – Форма додавання одиниці каталогу

Для перегляду однієї одиниці з каталогу використаємо наступне:

```
$this->title = $model->getFullName();
$this->params['breadcrumbs'][] = ['label' => 'Каталог', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="stell-view">
    <h1><?= Html::encode($this->title) ?></h1>
    <p>
```

```
            <?= Html::a('Редагувати', ['update', 'id' => $model->id], ['class' => 'btn
btn-primary']) ?>
            <?= Html::a('Видалити', ['delete', 'id' => $model->id], [
                'class' => 'btn btn-danger',
                'data' => [
                    'confirm' => 'Ви підтверджуєте видалення запису?',
                    'method' => 'post',
                ],
            ]) ?>
        </p>
        <?= DetailView::widget([
            'model' => $model,
            'attributes' => [
                'id',
                'name',
                'type.name',
                'description:ntext',
                [
                        'label' => Каталог вищого рівня',
                        'value' => ($model->parent) ? $model->parent->getFullName() :
"],
            ],
```

В результаті отримаємо сторінку, на якій можна редагувати, наприклад змінити назву каталогу або її опис, або видалити дані з каталогу.

3.5 Створення сторінки авторизації та реєстрації користувача

Наступним етапом є створення сторінки авторизації користувача.

Для того щоб створити такий елемент, потрібна тільки форма авторизації і скрипт, скрипт який повинен обробляти дані, після того коли дані вводяться в полях форми, оскільки тільки після введення логіну і паролю, можна буде зайти на таку сторінку сайту.

Для того щоб на сторінку сайту можна було відвідати тільки через, таке спеціальне правило, це зайти після введення логіну і паролю, для цього потрібно створити саму форму авторизації для входу (рис. 3.8, 3.9).

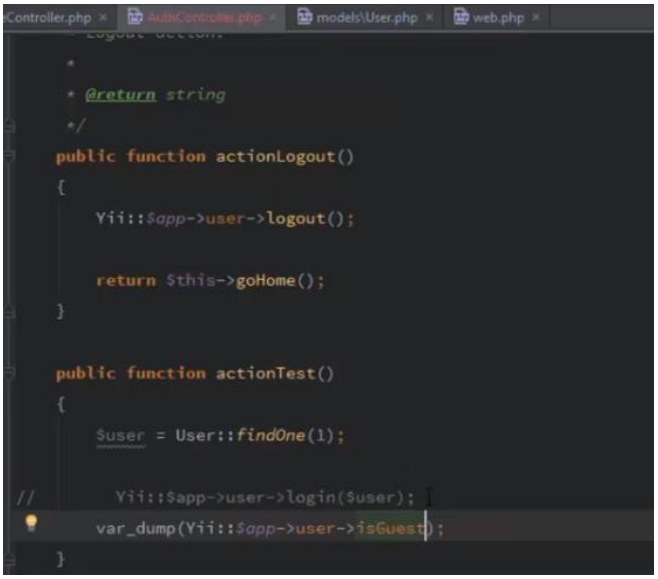

Рисунок 3.8 – Створення реєстрації користувача

Форма авторизації на сайті створюється для того щоб сайт, або окрема сторінка сайту була доступна, і передивлялась тільки окремим колом відвідувачів, а не кожним, будь яким відвідувачем.

Сам логін і пароль має бути прописаний в скрипті, прописаний в ручну, в одному з рядків коду, і коли на сторінці сайту в полі форми авторизації набирати цей логін і пароль, який ви прописали заздалегідь в кодах скрипта, тоді відбувається перенаправлення на сторінку сайту, в іншому випадку вхід буде закритий.

Така сторінка може бути потрібна, щоб розмістити таку інформацію, яку можуть бачити тільки ті користувачі, яких ви вважаєте за потрібних.

Рисунок 3.9 – Створення авторизації користувача

Після авторизації ви отримаєте доступ до розділів сайту, що недоступні незареєстрованим користувачам.

Приклад сторінки, яку отримаємо після внесення усіх змін можна побачити на рисунку 3.10. Де користувач повинен ввести своє ім’я, електронну адресу та пароль, щоб авторизуватись або зареєструватися.

Рисунок 3.10 – Сторінка реєстрації користувача

Також всі зареєстровані користувачі мають змогу залишати коментарі у розділі «Commentaries» як зображено на рисунку 3.11. адміністратор має

право вирішувати залишати певний коментар на сторінці інформаційної системи або видалити його.

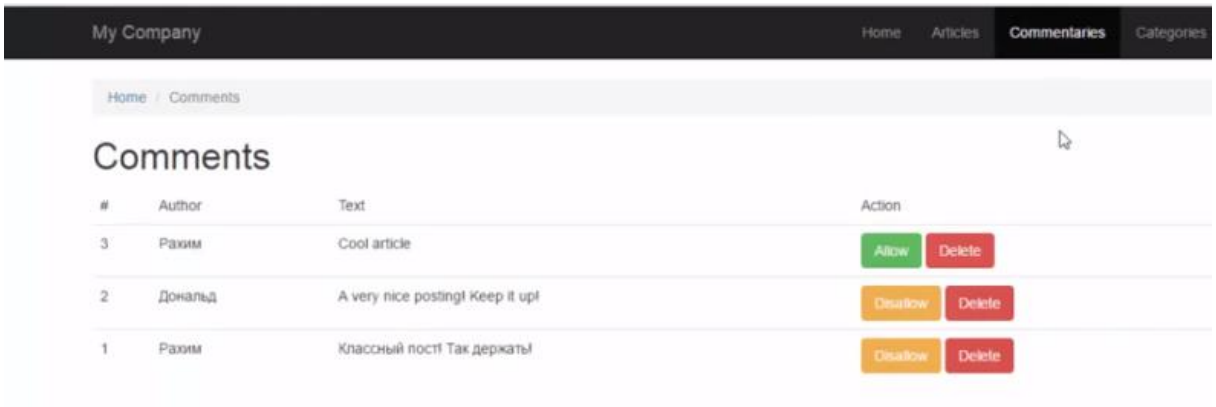

Рисунок 3.11 – Створення коментарів

3.6 Створення та перегляд сторінок сайту

Для перегляд всіх даних використаємо наступний програмний код:

```
$this->title = 'Стелажі';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="stell-index">
    <h1><?= Html::encode($this->title) ?></h1>
    <?php Pjax::begin(); ?>
    <?php // echo $this->render('_search', ['model' => $searchModel]); ?>
    <p>
        <?= Html::a('Створити новий запис', ['create'], ['class' => 'btn btn-success']) ?>
    </p>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'filterModel' => $searchModel,
```

```
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],
        'id',
        'name',
        [
            'attribute' => 'stell',
            'label' => 'Description',
            'value' => 'stell.nameWithParent'
        ],
        'address',
        ['class' => 'yii\grid\ActionColumn'],
    ],
]); ?>
<?php Pjax::end(); ?>
```

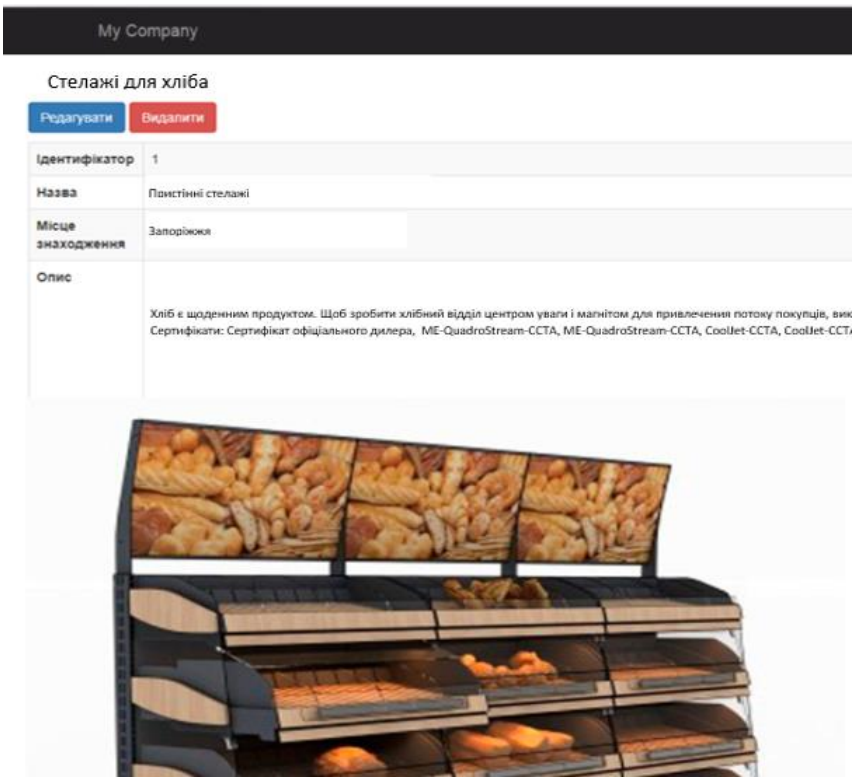

Рисунок 3.12 – Приклад представлення інформації про стелажі

В результаті роботи отримаємо сторінку інформаційної системи з точним повним описом кожного стелажу, які мають свій ідентифікатор, назву, опис, дату виготовлення, фотографію, розмірами, до якого типу даний стелаж відноситься. Приклад представлення стелажів зображено на рисунку 3.12.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є модель і автоматизована інформаційна система ведення обліку та моніторингу стелажів.

Підсумком ретельного вивчення предметної області, аналізу вже існуючих інформаційних систем, виділення їх недоліків та переваг, стало концептуальне проектування предметної області, побудова ER-діаграми з виділенням сутностей, їх типів та зв’язків та розробка єдиної уніфікованої моделі реляційної бази даних.

Створена база даних, до якої розроблено дружній інтерфейс користувача, котра є основою автоматизованої інформаційної системи «Каталог стелажів».

Для розробки інформаційної системи був використаний фреймворк Yii2. Yii – це високопродуктивний компонентний PHP фреймворк, призначений для швидкої розробки сучасних веб-додатків. Завдяки його компонентній структурі і відмінною підтримкою кешування, фреймворк особливо підходить для розробки таких проектів, як портали, форуми, CMS, магазини або RESTful-додатки. Для організації коду Yii використовує архітектурний патерн MVC (Model-View-Controller). MVC призначений для поділу бізнес-логіки і призначеного для користувача інтерфейсу, щоб розробники могли легко змінювати окремі частини програми, не зачіпаючи інші. В архітектурі MVC модель надає дані і правила, уявлення відповідає за користувальницький інтерфейс (наприклад, текст, поля введення), а контролер забезпечує взаємодію між моделлю і представленням.

Розроблена інформаційна система може бути використана для проектування інтер’єрів магазинів, супермаркетів, також нею можуть користуватися для ознайомлення з типами стелажів, їх характеристиками та розмірами. Спеціалізацією розробленої моделі є надання розгорнутої

інформації про різноманітні стелажі, їх категорії, характеристики та загальний огляд.

Інформаційна система є доступною для користувачів базового рівня підготовки та обізнаності у предметній області.

Отже, мета, поставлена на початку роботи, досягнута, всі завдання виконані.

Можливими напрямками розвитку може бути наповнення сайту більш детальною інформацією про стелажі, вдосконалення інтерфейсу й функціональних можливостей.

ПЕРЕЛІК ПОСИЛАНЬ

1. Петров П.В. Введение в системы баз данных. Спб.: ИТМО, 2010. 128 с.

2. Райордан Р. Основы реляционных баз данных. Москва : Русская Редакция, 2001. 384 с.

3. Коннолли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика. Москва : Вильямс, 2000. 1120 с.

4. Дейт К.Дж. Введение в системы баз данных. Москва : Вильямс, 2006. 1328 с.

5. Проектирование баз данных. *Общие рекомендации*. URL: http://www.mstu.edu.ru/education/materials/zelenkov/ch_5_1.html (дата звернення 16.10.2019).

6. Економіка. Управління. Інновації. *Інноваційні технології в туризмі*. URL: http://tourlib.net/statti_ukr/glebova2.htm (дата звернення 15.10.2019)

7. Розробка макету сайту. *Принципи та рекомендації*. URL: http://dist.org.ua/pluginfile.php/7033/mod_resource/content/1/2.%20%D0%A0%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0%20%D0%BC%D0%B0%D0%BA%D0%B5%D1%82%D1%83%20%D1%81%D0%B0%D0%B9%D1%82%D1%83.pdf (дата звернення 03.11.2019)

8. Створюємо свій сайт. *Навчально-методичний ресурс циклової комісії обліково-економічних дисциплін.* URL:http://aek-oed.pl.ua/stvoryuyemo-sviy-sayt/ (дата звернення 13.11.2019)

9. Фреймворк Yii. *Вікіпедія – вільна енциклопедія.* URL: https://uk.wikipedia.org/wiki/Yii (дата звернення 13.12.2019)

10. Фреймворк Yii2 с Нуля до Профи. *Yii2: Обучение.* URL: https://coursehunters.net/course/webformyself-yii2 (дата звернення 12.12.2019)

11. Створення Web-сторінок. *Інструментарій для створення Web-сторінок*. URL: http://gymlit.in.ua/re_%D0%A1%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F_Web-%D1%81%D1%82%D0%BE%D1%80% D1%96% D0%BD%D0 %BE%D0%BA (дата звернення 16.10.2019)

12. Працюємо над створенням якісного web-сайту. *Веб студія*. URL: http://www.web-master.lviv.ua/blog/pratsyujemo-nad-stvorennyam-yakisnoho-web-sajtu.html (дата звернення 16.10.2019)

13. Що таке внутрішня структура сайту. *Веб розробка*. URL: http://korusno-znatu.in.ua/internet/web-rozrobka/vnytrinnya-stryktyra-sajty.php (дата звернення 19.10.2019)

14. Дизайн сайтів: інформаційні сайти. *Веб студія*. URL: http://webstudio2u.net/ua/design-site/707-dizain-informatsionnykh-saitov.html (дата звернення 21.11.2019)

15. Плескач В. Л. Інформаційні системи і технології на підприємствах. *Засоби створення Web-сайтів*. URL: http://westu-dents.com.ua/glavy/27290-zasobi-stvorennya-Web-saytv.html (дата звернення 29.11.2019)