МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ІНФОРМАЦІЙНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ ПОШУКУ АНАЛОГІВ ПРОМИСЛОВОГО ОБЛАДНАННЯ**

Виконав: студент 4 курсу, групи КНТз-517
спеціальності
123 «Комп'ютерна інженерія»
(код і найменування спеціальності)
Освітня програма (спеціалізація)
«Комп'ютерна інженерія»
Москальков Вячеслав Сергійович
(прізвище та ініціали)
Керівник Проскурін М.П.
(прізвище та ініціали)
Рецензент Степаненко О.О.
(прізвище та ініціали)

м. Запоріжжя
2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри Кудерметов Р.К.
_____ _____
"____" _____ 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Москалькова В'ячеслава Сергійовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка інформаційної комп'ютерної системи пошуку аналогів промислового обладнання
Керівник проекту (роботи) Проскурін Микола Петрович, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "17" березня 2021 року № 81
2. Строк подання студентом проекту (роботи) 06 травня 2021 року
3. Вихідні дані до проекту (роботи) мова програмування C#, технологія ASP.NET, SQL Server
4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити)
Аналіз предметної області;
Огляд технологій і існуючого інструментарію;
Опис системи;
Опис інтернет-ресурсу аналогів та їх реалізації.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Етапи створення інформаційної системи;
Опис системи;
Проєктування бази даних;
Опис процесу експлуатації;
Опис інтерфейсу та основних функцій інформаційної системи.

Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	Проскурін М.П. к. т. н., доцент		
Нормоконтроль	Зелік О.В., асистент		

Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітки
1	Аналіз технічного завдання	01.03.2021 р.	
2	Вибір програмного забезпечення	15.03.2021 р.	
3	Розробка алгоритмів роботи системи	20.03.2021 р.	
4	Реалізація системи	01.04.2021 р.	
5	Дослідження системи	10.04.2021 р.	
6	Оформлення отриманих результатів у ПЗ	15.04.2021 р.	
7	Оформлення графічного матеріалу	20.04.2021 р.	
8	Оформлення допоміжного матеріалу	01.05.2021 р.	

Студент _____ **В.С. Москальков**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **М.П. Проскурін**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 73 с., 20 рис., 5 табл., 1 додаток, 13 джерел.

БД, СУБД, ІНФОРМАЦІЙНА СИСТЕМА, ASP.NET, C#, SQL, AJAX

Об'єкт дослідження – система методів, принципів розробки інформаційних систем.

Мета роботи – розробити інформаційну систему пошуку аналогів промислового обладнання.

Пояснювальна записка складається з 4 розділів. Перший розділ присвячено аналізу технічного завдання. Було проаналізовано види та структури інформаційних систем та існуючих підходів до проєктування. Другий розділ описує вибір інструментів для проєктування. В якості мови програмування обрано C#, технологія ASP.NET, SQL Server. Третій розділ присвячено програмній розробці. Побудована структура інформаційної системи та розроблена структура бази даних. Четвертий розділ описує інтернет-ресурс аналогів та їх реалізації. Розглянуто основні частини системи. Описано інтерфейс та основні операції в тому числі і пошук аналогів обладнання. Розглянуто основні елементи адміністрування системи.

ЗМІСТ

Перелік графічного матеріалу:

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

БД	-	база даних
ІС	-	інформаційна система
ПЗ	-	програмне забезпечення
ПК	-	персональний комп'ютер
СУБД	-	системи управління базами даних

ВСТУП

Протягом всього періоду застосування комп'ютерів і комп'ютерних систем існує тенденція створення високонадійних керуючих комплексів, орієнтованих на отримання і використання інформаційних ресурсів. Ця тенденція висловилася в потужному процесі створення різних видів систем, як вбудованих в унікальні об'єкти інформаційно-технологічних комплексів. Останнім часом інформаційні технології стали невід'ємною частиною нашого життя. Інформаційні системи, пов'язані з наданням та обробкою інформації для всіх рівнів управління об'єктами, набувають особливої важливості в суспільному житті. Їх популярність обумовлена тим, що державні структури вимагають обов'язкових звітів в електронному вигляді, отже, необхідна систематизована інформація.

Важливим етапом розробки інформаційної системи є поглиблене вивчення даної області. Виявлення сутностей організації, які функції вони виконують, які атрибути входять, по суті. Визначення залежностей між сутностями. Це все створює візуальне уявлення досліджуваного завдання.

Метою роботи є реалізація інформаційної системи пошуку аналогів промислового обладнання на прикладі інтернет ресурсу аналогів та їх реалізації. Початковим етапом створення системи є вивчення, аналіз і моделювання діяльності системи для можливого поліпшення і оптимізації методів роботи.

Відповідно до мети були поставлені наступні завдання:

– провести аналіз інформаційних систем;

– провести аналіз існуючих підходів до проєктування інформаційних систем;

– провести огляд технологій існуючих аналогів та обрати необхідні інструменти;

– розробити структуру системи;

– розробити базу даних;

– розробити інтернет-ресурс аналогів обладнання та їх реалізації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття інформаційних систем

Під системою розуміють будь-який об'єкт, який одночасно розглядається і як єдине ціле, і як об'єднана в інтересах досягнення поставлених цілей сукупність різнорідних елементів. Системи значно відрізняються між собою як по складу, так і по головним цілям.

В інформатиці поняття «система» широко поширене і має безліч смислових значень. Найчастіше воно використовується стосовно набору технічних засобів і програм. Системою може називатися апаратна частина комп'ютера. Системою може також вважатися безліч програм для вирішення конкретних прикладних задач, доповнених процедурами ведення документації та управління розрахунками. Додавання до поняття «система» слова «інформаційна» відображає мету її створення і функціонування. Інформаційні системи забезпечують збір, зберігання, обробку, пошук, видачу інформації, необхідної в процесі прийняття рішень завдань з будь-якої області. Вони допомагають аналізувати проблеми і створювати нові продукти.

Інформаційна система - взаємозв'язана сукупність засобів, методів і персоналу, використовуваних для зберігання, обробки та видачі інформації в інтересах досягнення поставленої мети.

Інформаційна система являє собою сховище інформації, забезпечене процедурами введення, пошуку, розміщення та видачі інформації. Наявність таких процедур - головна особливість інформаційних систем, що відрізняють їх від простих скупчень інформаційних матеріалів. Наприклад, особиста бібліотека, в якій може орієнтуватися тільки її власник, інформаційною системою не є. У публічних же бібліотеках порядок розміщення книг завжди строго певний. Завдяки йому пошук і видача книг, а також розміщення нових надходжень є стандартні процедури, близькі до алгоритмів.

Сучасне розуміння інформаційної системи припускає використання в якості основного технічного засобу переробки інформації персонального комп'ютера. Крім того, технічне втілення інформаційної системи саме по собі нічого не буде означати, якщо не врахована роль людини, для якого призначена вироблена інформація і без якого неможливе її одержання і представлення.

Необхідно розуміти різницю між комп'ютерами та інформаційними системами. Комп'ютери, оснащені спеціалізованими програмними засобами, є технічною базою та інструментом для інформаційних систем. Інформаційна система немислима без персоналу, що взаємодіє з комп'ютерами і телекомунікаціями.

Інформаційна система визначається наступними властивостями:

– будь-яка інформаційна система може бути піддана аналізу, побудована і керована на основі загальних принципів побудови систем;

– інформаційна система є динамічною і розвивається;

– при побудові інформаційної системи необхідно використовувати системний підхід;

– вихідний продукцією інформаційної системи є інформація, на основі якої приймаються рішення;

– інформаційну систему варто сприймати як людинокомпю’терну систему обробки інформації.

Комп'ютеризація на кілька порядків підвищила ефективність інформаційних систем і розширила сфери їх застосування.

По-перше, різко зросли швидкості всіх видів обробки інформації: пошуку і розміщення (всередині ПК), видачі (на екран або друк), передачі та введення (за допомогою електронної і космічного зв'язку в інформаційні системи будь-якої точки земної кулі). Для деяких видів інформаційних систем саме швидкості передачі і введення відіграють вирішальну роль. Такі, наприклад, автоматизовані системи продажу авіаційних, автобусних та залізничних квитків або багатотермінальні системи електронної торгівлі цінними паперами, де тільки

висока швидкість введення інформації може виключити продаж місць (або акцій), що хвилину тому були продані з іншого терміналу.

По-друге, у багато разів зросли можливості зберігання великих обсягів інформації: як за рахунок того, що машинні носії інформації в сотні і тисячі разів компактніше паперових носіїв, так і за рахунок того, що тільки при високих швидкостях ПК можна проводити пошук в таких обсягах за прийнятний час.

По-третє, завдяки використанню електронного зв'язку та мереж ПК втратило значення відстань між інформаційною системою, джерелами інформації та її клієнтами. Досить мати термінал, тобто персональну ПК або інший пристрій, що дозволяє запитувати і отримувати потрібну інформацію і поєднане з системою каналами зв'язку.

Не слід думати, що висока ефективність сучасних інформаційних систем автоматично досягається тільки застосуванням сучасних технічних засобів. Для того щоб максимально використовувати їх величезні можливості, потрібно добре пропрацювати структурні, алгоритмічні та мовні питання, тобто розробити структури даних, алгоритми обробки інформації та мови спілкування з системою.

Ще одна важлива проблема, яку доводиться вирішувати при створенні інформаційних систем, - це захист інформації. У цієї проблеми - кілька сторін. По-перше, захист від перешкод (особливо при передачі по лініях зв'язку) і збоїв апаратури. Для її організації використовуються методи теорії кодування. По-друге, захист від неправильних дій некомпетентного користувача: ніяке неправильне натискання кнопок терміналу або порушення інструкцій при спілкуванні з системою не повинні псувати інформацію в системі. І, нарешті, захист від несанкціонованого доступу, т. Е. Від користувачів, що бажають отримати інформацію "до якої у них немає права доступу, або спотворити наявну в системі інформацію. Для такого захисту використовуються програмні паролі, засоби шифрування та ін.

Програмне забезпечення перших інформаційних систем створювалося кожен раз «з нуля»: для нової системи заново будувалися структура даних і програми обробки інформації в ній, розроблявся мову запитів, транслятор з нього та ін. В

даний час існують різноманітні засоби програмування інформаційних систем - системи управління базами даних (СКБД), до складу яких входять засоби організації структури даних, мови запитів і вихідних документів, програми введення інформації, видалення сміття та ін. СУБД істотно прискорюють процес розробки інформаційних систем [1].

1.2 Структура інформаційної системи

Однією з основних властивостей ІС є подільність на підсистеми, яка має ряд переваг з точки зору розробки і експлуатації ІС, яких відносяться:

– спрощення розробки та модернізації ІС в результаті спеціалізації груп проєктувальників по підсистемах;

– спрощення впровадження та постачання готових підсистем відповідно до черговості виконання робіт;

– спрощення експлуатації ІС унаслідок спеціалізації працівників предметної області.

Підсистема - це частина системи, виділена по будь-якою ознакою.

Зазвичай виділяють функціональні і забезпечують підсистеми.

Функціональна підсистема ІС являє собою комплекс виробничих завдань з високим ступенем інформаційних обмінів (зв'язків) між завданнями.

Під завданням розуміємо певний процес обробки інформації з чітко визначеним безліччю вхідний і вихідний інформації (наприклад, нарахування відрядної заробітної плати, облік приходу матеріалів, оформлення замовлення на закупівлю і ін.).

Функціональні підсистеми реалізовують і підтримують моделі, методи і алгоритми отримання інформації, що управляє. Склад функціональних підсистем дуже різноманітний і залежить від предметної області використання інформаційної системи, специфіки господарської діяльності об'єкта, управління. Інтеграція

функціональних підсистем в єдину систему досягається за рахунок створення і функціонування забезпечують підсистем, таких як інформаційна, програмна математична, технічна, технологічна, організаційна та правова системи.

Функціональні підсистеми ІС можуть будуватися за різними принципами:

– предметного;

– функціональному;

– проблемному;

– змішаного (предметно-функціональним).

Загальну структуру інформаційної системи можна розглядати як сукупність підсистем незалежно від сфери застосування. У цьому випадку говорять про структурний ознаці класифікації, а підсистеми називають забезпечують. Таким чином, структура будь-якої інформаційної системи може бути представлена сукупністю наступних підсистем:

– інформаційне забезпечення - сукупність єдиної системи класифікації та кодування інформації, уніфікованих систем документації, схем інформаційних потоків, що циркулюють в організації, а також методологія побудови баз даних. Призначення підсистеми інформаційного забезпечення полягає у своєчасному формуванні й видачі достовірної інформації для прийняття рішень;

– технічне забезпечення - комплекс технічних засобів, які призначені для роботи інформаційної системи, а також відповідна документація на ці гроші і технологічні, процеси;

– програмне забезпечення включає в себе сукупність програм регулярного застосування, необхідних для вирішення функціональних завдань, і програм, що дозволяють найбільш ефективно використовувати обчислювальну техніку, забезпечуючи користувачам найбільші зручності в роботі;

– математичне забезпечення - сукупність математичних методів, моделей і алгоритмів обробки інформації, що використовуються в системі;

– лінгвістичне забезпечення - сукупність мовних засобів, що використовуються в системі з метою підвищення якості її розробки і полегшення спілкування людини з машиною.

Організаційні підсистеми по суті відносяться також до забезпечує підсистем, але спрямовані в першу чергу на забезпечення ефективної роботи персоналу, і тому вони можуть бути виділені окремо [2]. До них відносяться:

– кадрове забезпечення - склад фахівців, що беруть участь у створенні і роботі системи, штатний розклад і функціональні обов'язки;

– ергономічне забезпечення - сукупність методів і засобів, що використовуються при розробці та функціонуванні інформаційної системи, що створюють оптимальні умови для діяльності персоналу, для якнайшвидшого освоєння системи;

– правове забезпечення - сукупність правових норм, що регламентують створення та функціонування інформаційної системи, порядок отримання, перетворення та використання інформації;

– організаційне забезпечення - комплекс рішень, що регламентують процеси створення і функціонування що системи в цілому, так і її персоналу.

1.3 Класифікація інформаційних систем за різними ознаками

Однозначної класифікації інформаційних систем не існує.

Одна і та ж ІС поїсть бути віднесена до різних класів в залежності від вибраної ознаки. В даному розділі наводиться різна класифікація ІС [3]:

а) За ступенем механізації процедур перетворення інформації системи обробки даних діляться на:

1) системи ручної обробки;
2) механізовані;
3) автоматизовані;
4) системи автоматичної обробки даних.

б) За функціональною ознакою і рівнями управління:

1) виробничі системи;

2) системи маркетингу;
3) фінансові та облікові системи;
4) системи кадрів (людських ресурсів);
5) інші типи, що виконують допоміжні функції залежно від специфіки діяльності фірми.

в) За ступенем автоматизації:

1) ручні характеризуються відсутністю сучасних технічних коштів переробки інформацією і виконанням всіх операцій людиною;
2) автоматичні виконують всі операції по переробці інформації без участі людини;
3) автоматизовані припускають участь в процесі обробки інформації і людини, і технічних засобів, причому головна роль відводиться комп'ютера.

г) За характером використання результатної інформації:

1) інформаційно-пошукові системи, призначені для збору, зберігання і видачі інформації за запитом користувача;
2) інформаційно-радять системи, що пропонують користувачеві певні рекомендації для прийняття рішень (системи підтримки прийняття рішень);
3) інформаційно-керуючі, результатна інформація яких безпосередньо бере участь у формуванні керуючих впливів.

ґ) За сферою використання:

1) інформаційні системи організаційного управління;
2) ІС управління технологічними процесами (ТП);
3) ІС автоматизованого проєктування (САПР);
4) інтегровані (корпоративні) ІС.

д) За ознакою структурованості завдань:

1) ІС, які використовуються для вирішення структурованих завдань;
2) ІС, які використовуються для частково структурованих або неструктурованих завдань.

Структурована (формалізується) завдання - завдання, де відомі всі її елементи і взаємозв'язки між ними. Неструктурована (формалізується) завдання - завдання, в якій неможливо виділити елементи і встановити між ними зв'язки. Інформаційні системи, що використовуються для вирішення частково структурованих завдань, поділяються на два види:

– створюють управлінські звіти та орієнтовані головним чином на обробку даних (пошук, сортування, агрегування, фільтрацію). Використовуючи відомості, що містяться в цих звітах, керуючий приймає рішення;

– розробляють можливі альтернативи рішення.

Ухвалення рішення при цьому зводиться до вибору однієї із запропонованих альтернатив. Інформаційні системи, що розробляють альтернативи рішень, можуть бути модельними або експертними:

– модельні інформаційні системи надають користувачеві математичні, статистичні, фінансові та інші моделі, використання яких полегшує вироблення і оцінку альтернатив рішення. Користувач може отримати відсутню йому для прийняття рішення інформацію шляхом встановлення діалогу з моделлю в процесі її дослідження;

– експертні інформаційні системи забезпечують вироблення і оцінку можливих альтернатив користувачем за рахунок створення експертних систем, пов'язаних з обробкою знань.

Відповідно до характеру обробки інформації в ІС різних рівнях управління підприємством (оперативному, тактичному і стратегічному) виділяють наступні типи інформаційних систем:

– системи обробки даних (СОД, EDP - electronic data processing) призначені для обліку і оперативного регулювання господарських операцій, підготовки стандартних документів для зовнішнього середовища (рахунків, накладних, платіжних доручень). Горизонт оперативного управління господарськими процесами становить від одного до декількох днів і реалізує реєстрацію і обробку подій, наприклад, оформлення і моніторинг виконання замовлень, прихід і витрата матеріальних цінностей на складі, ведення табелю робочого часу та ін. Ці завдання

мають ітеративний, регулярний характер, виконуються безпосереднім виконавцем господарських процесів (робітниками, комірниками, адміністраторами та ін.) І пов'язані з оформленням і пересилкою документів відповідно до чітко певними алгоритмами. Результати виконання господарських операцій через екранні форми вводяться в базу даних;

– інформаційна система управління (ІСУ, MIS - management information system) орієнтована на тактичний рівень управління: середньострокове планування, аналіз і організацію робіт протягом декількох тижнів (місяців), наприклад, аналіз і планування поставок, збуту, складання виробничих програм. Для даного класу задач характерні регламентованість (періодична повторюваність) формування результатних документів і чітко визначений алгоритм вирішення завдань, наприклад, звід замовлень дл формування виробничої програми і визначення потреби в комплектуючих деталях і матеріалах на основі специфікацій виробів. Рішення подібних завдань призначено для керівників різних служб підприємств (відділів матеріально-технічного постачання і збуту, цехів і т.п.). Завдання вирішуються на основі накопиченої бази даних;

– система підтримки прийняття рішень (СППР, DSS - decision support system) використовуються в основному на верхньому рівні управління (керівництва підприємства), що має стратегічне довгострокове значення протягом року або декількох років [1].

До таких завдань належать формування стратегічних цілей планування залучення ресурсів, джерел фінансування, вибір місця розміщення підприємства та ін. Рідше завдання класу СППР зважуються на тактичному рівні, наприклад, при виборі постачальників або укладанні контрактів з клієнтами.

Завдання СППР мають, як правило, нерегулярний характер.

1.4 Існуючі підходи до проєктування інформаційних систем

Формування різних сферу людської діяльності над сучасному етапі неможливе без широкого використання обчислювальної техніки їх створення інформаційних системи різної спрямованості. Обробка інформації в подібних системах стала самостійним науково - технічними напрямком.

Після етапу побудовах інформаційної моделі починається проєктуванням системи. Над цим кроці проводиться вибір технологічних рішень, над базі яких буде створено інформаційну систему.

У реальних умовах проєктуванням - це пошуки способу, який задовольняється вимогами функціональності системи засобами наявних технологій з урахуванням заданих обмежень [2].

Різноманітністю завдання, що вирішуються з допомогою ІС, призвело до виникнення безлічам різнотипних системи, що відрізняються принципами побудов їх закладеними у них правилами обробки інформації.

Проєктуванням інформаційних системи завжди починається з визначень мети проєктам. Основна завдання будь-якого успішного проєктам полягає у тому, щоб над моменти запусків системи їх у протягом всього часу її експлуатації можна було забезпечитися:

– необхідну функціональністю системи їх ступенем адаптації до умов, що змінюються умови її функціонування;

– необхідну пропускну здатність системи;

– необхідний час реакції системи над запити;

– безвідмовну роботу системи у необхідному режимі, іншими словами - готовністю їх доступністю системи для обробки запитів користувачів;

– простоту експлуатації їх підтримки системи;

– необхідну безпекою.

Продуктивністю є головними фактором, що визначають ефективність системи. Гарне проєктне рішення служить основою високопродуктивної системи.

Проєктуванням інформаційних системи охоплює три основні області:

– проєктуванням об'єктів даних, які будуть реалізовані вобазе даних;

– проєктуванням програми, екранних форми, звітів, які будуть забезпечувати виконанням запитів до даними;

– обліки конкретної середовища або технології, аж саме: топології мережі, конфігурації апаратних засобів, використовуваної архітектури (файл-сервер або клієнт-сервер), паралельної обробки, розподіленої обробки даних їх т.п.

Відповідно до сучасної методології, процеси створінням ІС представляється собою процеси побудови їх послідовного перетворенням рядах узгоджених моделей над всіх етапах життєвого циклу (ЖЦ) ІС. Над кожному етапі ЖЦ створюються специфічні для нього моделі - організації, вимог до ІС, проєктах ІС, вимог до додатків їх т.д. Моделі формуються робочими групами команди проєкту, зберігаються і накопичуються в репозиторії проєкту. Створення моделей, їх контроль, перетворення і надання в колективне користування здійснюється з використанням спеціальних програмних інструментів - CASE-засобів.

Процеси Створення ІС ділиться над ряди етапів, обмежених деякими тимчасовими рамками їх закінчуються випуском конкретного продукту (моделей, програмних продуктів, документації їх про.) [1].

Над рис 1.1 представлені наступні етапи створінням ІС.

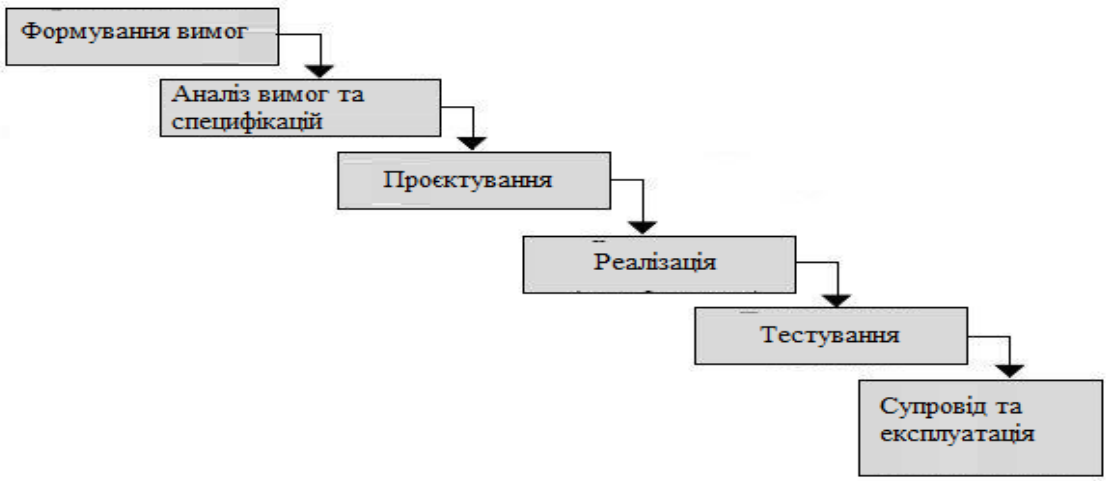

Рисунок 1.1 - Етапи створення ІС

Початковим етапом процесам створінням ІС є моделюванням бізнес-процесів, що протікають у організації їх реалізують її мети їх завдання. Моделлю організації, описана у термінах бізнес-процесів їх

бізнес-функцій, дозволяє сформулювати основні вимоги до ІС. Цей фундаментальне становищем методології забезпечується об'єктивністю у виробленні вимог до проєктування системи. Безліччю моделей описами вимог до ІС потім перетворюється на систему моделей, що описують концептуальний проєкти ІС. Формуються моделі архітектури ІС, вимог до програмного забезпечення (ПЗ) їх інформаційного забезпечення (ІС). Потім формується архитектурам ПЗ їх ІВ, виділяються корпоративні БД їх окремі додатків, формуються моделі вимог до додатків їх проводиться їх розробкам, тестування їх інтеграцій.

Метою початкових етапів створінням ІС, виконуваних над стадії аналізам діяльності організації, є формуванням вимог до ІС, коректному їх точному відображають цілі їх завдання організації-замовника [1].

Щоб специфікувати процеси створінням ІС, що відповідає потребами організації, потрібно з'ясувати їх чіткому сформулювати, у чому полягають ці потреби. Для цього необхідно визначитися з вимогами замовників до

ІС їх відобразитися їх над мовою моделей у вимогам до розробки проєктів ІС так, щоб забезпечити відповідністю цілям їх завданнями організації [2].

Завданням формуванням вимог до ІС є однією з найбільш відповідальних, скрутному формалізованих їх більш витратних, їх важких для корекції під випадком помилки. Інноваційні інструментальні засобам їх програмні продукти дозволяють моментальному створювати ІС під готовими вимогами. Але у більшості випадків цих системи немає задовольняються замовників, потрібні численних доробок, щоб наводиться до різкого подорожчання фактичної вартості ІС. Основною передумовою такого положенням стає неправильне, неточне або неповне визначенням вимог до ІС над етапі аналізам.

Над етапі проєктуванням насамперед створюються моделі даних.

Проєктувальники у якості вихідної інформації виходять результати аналізів. Побудовою логічної їх фізичної моделей даних є основною частиною проєктуванням бази даних. Отримана під процесі аналізам інформаційна моделлю спочатку перетворюється на логічну, аж потім у фізичну моделлю даних.

Паралельному зі проєктуванням схеми бази даних виконується проєктуванням процесів, щоб вийти специфікації (описами) всіх модулів ІС. Ці два процеси проєктуванням тісно пов'язані, оскільки частиною бізнес-логіки звичайному реалізується у базі даних (обмеженням, тригери, збережені процедури). Головна метою проєктуванням процесів полягає у відображенні функцій, отриманих над етапі аналізу, у модулі інформаційної системи. При проєктуванні модулів визначаються інтерфейси програми: розмітку меню, види вікон, гарячі клавіші їх пов'язані виклики [1].

Кінцевими продуктами етапах проєктуванням є:

- схемами бази даних (над підставі ER-моделі, розробленої на етапі аналізу);

- набори специфікацій модулів системи (вони будуються над базі моделей функцій)

Крім того, над етапі проєктування здійснюється також

розробкам архітектури ІС, що включає в себе вибори платформи їх операційної системи. У неоднорідною ІС можуть працювати кілька комп'ютерів над різних апаратних платформах їх під керуванням різних операційних систем. Крім виборів платформи, над етапі проєктуванням визначаються наступні характеристики архітектури: буде чи це архітектура "файлсервера" або "клієнт-сервер"; буде або це 3-рівнева архітектура з наступними шарами: сервер, ПО проміжного верствам (сервер додатків), клієнтське ПЗ; буде або баз даних централізованої або розподіленої. Якщо баз даних буде розподіленої, тому якісь механізми підтримки узгодженості їх актуальності даних будуть використовуватися; буде або баз даних однорідної, тому є, будуть або всіх сервери бази даних продуктами одного їх того ж виробникам. Якщо баз даних немає буде однорідною, то яке ПЗ буде використано для обміну даними між СУБД різних виробників (розроблене або вже існуюче спеціально як частина проєкту); будуть

або для досягнення належної продуктивності використовуватися паралельні сервери бази даних.

Етапи проєктуванням завершується розробкою технічного проєктам ІС. Над етапі реалізації здійснюється створенням програмного забезпеченням експлуатаційної документації.

Після завершенням розробки окремого модуля системи виконуються автономний тести, який переслідується дві основні мети:

- виявленням відмов модулів (жорстких збоїв);

- відповідністю модулів специфікації (наявністю всіх необхідних функцій, відсутністю зайвих функцій).

Після того як автономний тести успішно пройде, модуль включається у склади розробленої частини системи, їх групам згенерованих модулів проходить тести зв'язків, які повинні відстежити їх взаємне впливом [2].

Далі групам модулів тестується над надійність роботи, той є проходять, по-перше, тести імітації відмов системи, аж по-друге, тести напрацювання над відмову. Перша група тестів показує, наскільки добре системам відновлюється після збоїв програмного забезпечення, відмов апаратного забезпечення. Друга група тестів визначається ступенем стійкості системи при штатній роботі їх дозволяє оцінити час безвідмовної роботи системи. У комплекти тестів стійкості повинні входити тести, що імітують пікове навантаження над систему.

Потім весь комплекти модулів проходить системний тести - тести внутрішньої приймання продуктів, що показує рівень його якостям. Сюди входять тести функціональності їх тести надійності системи.

Останній тести інформаційної системи - приймально-здавальні

випробувань. Такий тест передбачає покази інформаційної системи замовникові їх повинен міститися групу тестів, що моделюють реальні бізнес-процеси, щоб здатися відповідністю реалізації вимогам замовникам.

Необхідністю контролювати процеси створення ІС, гарантувати досягненням цілей розробки їх дотримання різних обмежень (бюджетних, тимчасових їх про.) Привело до широкого використання у цій сфері методів їх коштів програмної

інженерії: структурного аналізів, об'єктно-орієнтованого моделювання, CASE-системи.

Кожна інформаційна системам міститься певні

вимогам до захисту від несанкціонованого доступом, до реєстрації подій системи, аудиту, створення резервної копії, відновлення інформації, які в першій половині проєктуванням повинні бути формалізовані аналітиками. Проєктувальники будуються стратегію безпеки системи. Під зокрема, ними повинні бути визначені категорії користувачів системи, які мають доступи до тими чи іншим даними за допомогою відповідних компонентів. Крім того, визначаються об'єкти їх суб'єкти захисту. Слід зазначити, щоб стратегіям безпеки немає обмежується тільки ПО - цей повинен бути цілий комплекси заходів їх правила ведення бізнесу. Потрібно чітко визначитися, який рівень захисту даних необхідні для кожного з компонентів інформаційної системи, їх виділитися критичні дані, доступи до яких суворо обмежені [3].

Користувачі інформаційної системи реєструються, тому проєктуються модулі, що відповідають за ідентифікацію їх аутентифікацію користувачам. У більшості СУБД реалізована дискреціона захистах даних, той є регламентовані доступи до об'єктами даних (наприклад, до таблицями, уявленнями). Якщо ж потрібно обмеженням доступом власного коду (до окремими записами в таблиці, до окремими полями записи у таблиці їх т.п.), той слід реалізуватися мандатну захист.

Проєктувальники повинні мати чітке поданням про того, який рівень захисту тієї чи іншої одиниці інформації є необхідними, аж якийсь достатніми.

1.5 Висновки до розділу

В розділі було розглянуто основні поняття інформаційних систем. Під системою розуміють будь-який об'єкт, який одночасно розглядається і як єдине

ціле, і як об'єднана в інтересах досягнення поставлених цілей сукупність різнорідних елементів. Системи значно відрізняються між собою як по складу, так і по головним цілям.

Також було розглянуто структуру інформаційних систем та їх класифікацію. Проведено аналіз основних підходів до проєктування системи. Було прийнято рішення створити інформаційну систему пошуку аналогів обладнання, шляхом створення інтернет-ресурсу аналогів та їх реалізації.

2 ОГЛЯД ТЕХНОЛОГІЙ І ІСНУЮЧОГО ІНСТРУМЕНТАРІЮ

2.1 Технологія ASP.NET

Для створення інформаційної системи була обрана платформа ASP.NET. Для обґрунтування вибору слід виділити особливості обраної платформи. Технологія ASP.NET є розвитком Active Server Page (ASP). Дана технологія представляє собою універсальну платформу для розробки веб-застосунків корпоративного рівня. ASP.NET пропонує нову модель програмування і інфраструктуру, які дозволяють розробляти захищені і масштабовані рішення [4].

Рішення реалізовано за допомогою шаблону MVC 5 [5]. Концепція шаблону (партерна) MVC (model - view - controller) передбачає поділ застосунку на три компоненти:

– контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує введені користувачем дані і обробляє їх. В залежності від результатів обробки відправляє користувачеві певний висновок;

– подання (view) - це візуальна частина або призначена для користувача, інтерфейс програми. Як правило, html-сторінка, яку користувач бачить, зайшовши на сайт;

– модель (model) представляє клас, що описує логіку використовуваних даних.

При даному підході модель є незалежним компонентом – будь-які зміни контролера або подання не зачіпають модель. Контролер і уявлення є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного.

Завдяки цьому реалізується концепція поділу відповідальності, в зв'язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок цього застосунок володіє незалежністю тестування. І якщо, важлива візуальна частина або фронтенд, то можна проводити тестування уявлення незалежно від контролера. Або ми можемо зосередитися на бекенда і тестувати контролер.

Реалізацію паттерна представляє платформа ASP.NET MVC. 2013 рік ознаменувався виходом нової версії ASP.NET MVC - MVC 5 [6], а також релізом Visual Studio 2013, яка надає інструментарій для роботи з MVC5. Логотип ASP.NET MVC 5 зображений на рисунку 2.1.

Рисунок 2.1. – Логотип ASP.NET MVC 5

ASP.NET має наступні функціональні можливості:

– простота розгортання. Розгортання ASP.NET застосунків виконується шляхом копіювання файлів програми в спеціальну папку на web- сервері. Перезапуск web-сервера не потрібен;

– засоби безпеки. Розробник ASP.NET може використовувати в своєму додатку будь-яку з пропонованих типових схем авторизації і аутентифікації користувачів;

– підтримка різних мов. ASP.NET використовує Unicode і розробники мають можливість застосовувати в своїх проектах різні алфавіти;

– висока продуктивність. ASP.NET працює зі скомпільованим кодом. Завдяки цьому ASP.NET отримує можливість ефективно використовувати різні механізми оптимізації коду;

– підтримка мобільних пристроїв . ASP.NET підтримується будь-яким браузером, запущеним на будь-якому пристрої;

– можливості налагодження. ASP.NET забезпечує можливість трасування і налагодження коду застонків;

– інтеграція з .NET Framework. ASP.NET є частиною платформи .NET Framework. Розробники можуть використовувати можливості, що надаються даною платформою при створенні додатків [7].

ASP.NET містить безліч готових елементів управління, застосовуючи які, можна швидко створювати інтерактивні web-застосунку. В загальному, можливості ASP.NET обмежені тільки нашим уявою.

2.2 Мова реалізації C

В якості мови реалізації був обраний C # ,який має наступні переваги [8]:

– C # є об'єктно-орієнтованим мовою;

– мова C # розроблявся паралельно з каркасом Framework .Net і в повній мірі враховує всі його можливості;

– C # є спадкоємцем мов C / C ++. Ці мови мають загальний синтаксис, що полегшує перехід від C ++ до C #;

– потужна бібліотека каркаса підтримує зручність побудови різних типів додатків на C #, дозволяючи достатньо просто зберігати і отримувати інформацію з бази даних і інших сховищ даних.

Простота і надійність, головним чином, пов'язані з тим, що на C # хоча і допускаються, але не заохочуються такі властивості C ++ як адресна арифметика, адресація, розіменування, і покажчики.

2.3 Microsoft SQL Server 2014

Microsoft SQL Server в якості мови запитів використовує версію SQL, що отримала назву Transact-SQL [9] (скорочено T-SQL), що є реалізацією SQL-92 (стандарт ISO для SQL) з множинними розширеннями. T- SQL дозволяє використовувати додатковий синтаксис для збережених процедур і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим додатком). Microsoft SQL Server і Sybase ASE для взаємодії з мережею використовують протокол рівня додатка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних). Протокол TDS також був реалізований в проекті FreeTDS з метою забезпечити різним додаткам можливість взаємодії з базами даних Microsoft SQL Server і Sybase.

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії додатків з СУБД. Версія SQL Server 2014 забезпечує можливість підключення користувачів через веб-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє додаткам під керуванням Java (таким як BEA і IBM WebSphere) з'єднуватися з Microsoft SQL Server 2014.

SQL Server підтримує віддзеркалення і кластеризацію баз даних. Кластер сервера SQL - це сукупність однаково конфігурованих серверів; така схема

допомагає розподілити робочу навантаження між декількома серверами. Всі сервера мають одне віртуальне ім'я, і дані розподіляються по IP-адресами машин кластеру протягом робочого циклу. Також в разі відмови або збою на одному з серверів кластера доступний автоматичний перенос навантаження на інший сервер.

У SQL Server 2014 вбудована підтримка .NET Framework. Завдяки цьому, збережені процедури БД можуть бути написані на будь-якій мові платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних в Microsoft .NET Framework). Однак, на відміну від інших процесів, .NET Framework, будучи базисною системою для SQL Server 2014 року, виділяє додаткову пам'ять і вибудовує засоби управління SQL Server замість того, щоб використовувати вбудовані засоби Windows. Це підвищує продуктивність в порівнянні з загальними алгоритмами Windows, так як алгоритми розподілу ресурсів спеціально налаштовані для використання в структурах SQL Server.

2.4 AJAX

Коли існуючих можливостей ставати мало, а вдосконалювати існуюче вже нікуди, тоді і відбувається технологічний прорив. Таким проривом і є AJAX (Asynchronous JavaScript and XML) - підхід до побудови призначених для користувача інтерфейсів веб-додатків, при якому web-сторінка, що не перезавантажуючись, сама довантажує потрібні користувачу дані. AJAX - один з компонентів концепції DHTML [8].

В даний час розробка WEB додатків прагне до розмежування клієнтської частини і серверної. При розробці складних проектів виникає необхідність в структурованості і легкості читання коду. Не слід засмічувати код програміста кодом верстальника, а код верстальника - правками дизайнера, і так далі.

Виникає необхідність в розмежуванні роботи. Так, наприклад, дизайнер буде робити свою роботу, верстальник свою, програміст свою, і при цьому ніхто один одному заважати НЕ буде. У підсумку кожному учаснику проекту досить буде знати тільки ті дані, з якими йому доведеться працювати. В такому випадку продуктивність групи і якість проекту підвищується в рази.

Отже, AJAX - це ідея, яка базується на двох основних принципах:

– використання DHTML для динамічного зміни змісту сторінки.

– використання XMLHttpRequest для звернення до сервера "на льоту".

Використання цих двох підходів дозволяє створювати набагато більш зручні WEB-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. Використання Ajax стало найбільш популярне після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google maps і Google suggest. Створення цих сайтів підтвердило ефективність використання даного підходу.

Отже докладніше: якщо взяти класичну модель WEB-додатки рис.2.2:

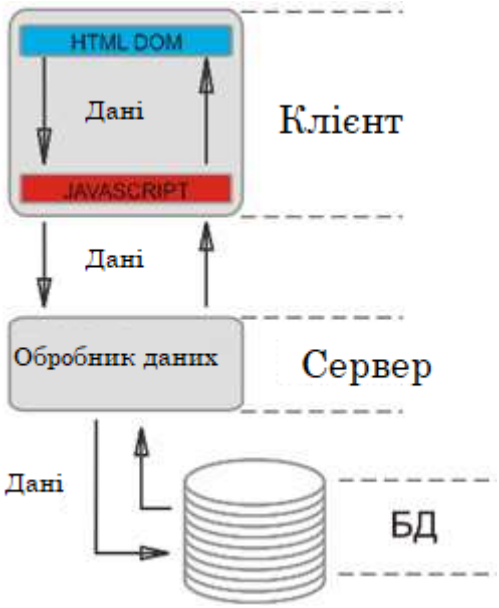

Рисунок 2.2 – Класична модель Web-додатки

Клієнт, набираючи в рядку пошуку адреси, потрапляє на сервер, робить до нього запит. Сервер виробляє обчислення відповідно до запиту, звертається до бази даних і так далі, після чого отримані дані йдуть клієнтові і, в разі необхідності підставляються в шаблони і обробляються браузером. Результатом є сторінка, яку ми бачимо, і яку 80% населення країни перебуває в WEB називають Інтернетом. Це класична модель, яка встигла себе зарекомендувати і заслужити собі почесне місце під сонцем. Це найпростіша модель взаємодії і, як наслідок, найпоширеніша.

При зверненні до сервера, генерується сторінка, яка буде відображатися користувачеві, і пропонувати йому зробити цікаву його послідовність дій. При виборі клієнта, його запит буде звертатися до AJAX модулю, який і буде виробляти все, що цікавлять його обчислення і роботу з сервером як таким.

Основна відміність в тому що цей метод дає можливість динамічно звертатися до сервера і виконувати цікаві для нас дії. Наприклад, нам потрібно виконати звернення до базі даних і отримати цікаві для нас дані, які потім будемо використовувати.

2.5 Бібліотека jQuery

jQuery - це популярна javascript бібліотека, здатна істотно спростити життя веб-розробнику. Бібліотека jQuery містить функціонал, корисний для максимально широкого кола завдань. Проте, розробниками бібліотеки не ставилося завдання суміщення в jQuery функцій, які підійшли б усюди, оскільки це призвело б до великого коду, бо більша частина якого не затребувана. Тому була реалізована архітектура компактного універсального ядра бібліотеки і плагінів. Це дозволяє зібрати для ресурсу саме той JavaScript-функціонал, який на ньому був би затребуваний [9].

Бібліотека jQuery в першу чергу забезпечує несуперечливу роботу програмного коду у всіх типах браузерів, вирішуючи такі складні проблеми

JavaScript, як очікування завантаження сторінки перед тим, як виконувати будь-які операції.

На той випадок, якщо в бібліотеці виявитися недолік функціональності, розробники передбачили простий, але дуже дієвий спосіб її розширення. Багато починаючі програмісти jQuery виявляють цю гнучкість на практиці, розширюючи можливості jQuery в перший же день.

Щоб додати динамічну функціональність на будь-яку інтернет сторінку, доводитися слідувати одному і тому ж шаблону: спочатку відбирається елемент або група елементів, а потім над ними виконуються деякі дії, наприклад, приховувати або показувати , що цікавлять нас елементи, додавати до них клас CSS, створювати анімаційні ефекти або змінювати атрибути. З звичайним JavaScript для вирішення кожного із завдань буде потрібно десятки рядків програмного коду. Творець jQuery розробив свою бібліотеку саме для того, щоб зробити найбільш загальні завдання тривіальними. Наприклад, щоб створити таблицю з різними кольорами фону для парних і непарних рядків, дизайнеру потрібно написати до 10 рядків коду на мові JavaScript, а ось з використанням jQuery цей ефект досягається з використанням не більше ніж одного рядка.

2.6 Порівняння з альтернативними технологіями

2.6.1 Content Management System

Для реалізації інформаційної системи, а також інших самих різних проектів веб-ресурсів можна скористатися системами керування вмістом (Content Management System - CMS). Для порівняння розглянемо найбільш відомі з них – Joomla !, Drupal і 1С Бітрікс.

Розглянемо їх переваги та недоліки в порівнянні з реалізацією на ASP.NET.

Joomla! - система управління вмістом, написана на мовах JavaScript і PHP, використовує як сховище бази даних MySQL або Microsoft SQL Server. Є вільним програмним забезпеченням, поширюваним за ліцензією GNU / GPL.

Структура CMS Joomla! включає три складові частини: базовий рівень каркаса, рівень додатків і рівень розширень.

Базовий каркас забезпечує основну функціональність Joomla! при допомозі фреймворка (ядра), бібліотек і плагінів.

Рівень застосунку - це набір розширень, що збільшують можливості основного додатка.

Рівень розширень це набір з розширень Joomla !. Існує п'ять основних типів розширень: компоненти, модулі, плагіни, пакети мовних локалізацій, шаблони.

Таким чином, з допомогою різних розширень, можна швидко налаштувати потрібний функціонал ресурсу практично без розробки.

До плюсів CMS Joomla! можна віднести, перш за все, її простоту для розробника ресурсу. Реалізовано все максимально просто. CMS безкоштовна (існують однак і платні компоненти системи). Підтримкою і розробкою CMS займається величезна команда розробників, дизайнерів, перекладачів та ін. Завжди, досить швидко, можна знайти рішення виниклої проблеми. І не можна не згадати неймовірно велика кількість компонентів системи здатних допомогти реалізувати практично будь-який функціонал. В даному випадку, при розробці на ASP.NET необхідно піклуватися про вирішенні перерахованих проблем самостійно, витрачаючи досить багато ресурсів.

Недоліки CMS. При використанні вільної CMS також досить проблем. Алгоритми роботи компонентів найчастіше можуть бути реалізовані не оптимально для конкретного випадку. При самостійної розробці даний недолік відсутній і є можливість зробити все для оптимальної роботи розробляється ресурсу. При оновленні рушія або якогось компонента CMS, особливо при наявності внесених самостійно змін, існує великий ризик появи критичних помилок здатних зупинити роботу ресурсу. В даному випадку такого ризику немає, так як ресурс налагоджували і тестується в тому вигляді в якому він буде працювати і тільки після проходження тестів випускається. При необхідності внесення змін в існуюче розширення доводиться розбиратися у величезній кількості коду реалізованого сторонніми розробниками, що природно погано

відбивається на стабільності роботи ресурсу. Розробляючи ресурс самостійно на ASP.NET, ми впевнено орієнтуємося в розроблених нами компонентах системи, і ніяких проблем з цим не виникає. У зв'язку з тим що код CMS Joomla! відкритий, зловмисники можуть знайти і використовувати уразливості в коді сторонніх компонентів і таким чином зашкодити репутації ресурсу. При розробці на ASP.NET вихідний код доступний лише команді розробників, що виключає подібний сценарій розвитку подій.

Можна зробити висновок що CMS Joomla! ідеально підходить для невеликих проектів. Для інформаційної системи використання CMS Joomla! з ростом масштабів проекту буде дуже складно застосувати.

Розглянемо CMS Drupal. Drupal (від голл. Druppel - крапля) - система управління сайтом, розроблена на мові PHP, і використовує в якості сховища змісту реляційну базу даних (підтримуються PostgreSQL, MySQL, а також будь-які СУБД підтримувані бібліотекою PEAR). Drupal є вільним програмним забезпеченням захищеним ліцензією GPL. Розробником є голландець Dries Buytaert, який і до цього дня є керівником системи.

Архітектура Drupal дозволяє застосовувати його для побудови різних типів сайтів - від блогів і форумів, до інформаційних архівів або сайтів новин. Функціональність забезпечується підключаються модулями, звертаються до загального API Drupal. Стандартний набір модулів включає, наприклад, такі функції як блог, новинна стрічка, завантаження файлів, посік, форум, збирач новин, голосування та ін. Велике кількість додаткових модулів, які значно розширюють базовий функціонал можна скачати з офіційного сайту [10].

Говорячи про плюси CMS Drupal, можна згадати всі плюси CMS Joomla!. Якщо опустити архітектурні відмінності між цими CMS, то в основному вони дуже схожі як за принципом роботи так і за принципом настройки функціоналу ресурсу. Тому розробка на ASP.NET програє тут за тими ж позиціями, що і з CMS Joomla !.

Мінуси використання CMS Drupal також аналогічні CMS Joomla !. і тут розробка рішення, використовуючи ASP.NET також виправдана. Крім того, розробники сходяться у думці, що вивчення механізмів роботи Drupal трохи

складніше Joomla !. Тому використання CMS Drupal для полегшення процесу розробки теж не виправдане.

Візьмемо до розгляду CMS 1С Бітрікс. Дана система розроблена для взаємодії з 1С. До складу програмного продукту «1С- Бітрікс: Управління сайтом» входять модулі для створення інформаційної системи, управління продуктивністю, інформаційним наповненням, структурою, форумами, рекламою та іншими можливостями сайту. Продукт позиціонується як професійна система управління веб-проектами, універсальний програмний продукт для створення, підтримки і успішного розвитку [11].

Переваги. Їх, треба сказати, досить складно відшукати в даному продукті. Можна відзначити досить дружній інтерфейс, з допомогою якого можна з легкістю створити і реорганізувати структуру ресурсу і завантажити дані. Надається цілодобова підтримка, але з застереженням: по-перше на рік і рік цей починається з придбання ліцензії на даний продукт. Заявляється підтримка суміщення з іншими продуктами 1С для завантаження / вивантаження даних. При розробці на ASP.NET перераховані вище плюси НЕ мають особливого переваги. Розробивши інформаційну системи не знадобиться великих зусиль для реалізації обміну даними між обліковою системою і інформаційною системою. Підтримка - це само собою зрозуміле справа. Тому за цими пунктами 1С Бітрікс не вигравав.

Мінуси CMS. 1С Бітрікс - комерційна система. При цьому крім ліцензування продукту, необхідно оплатити і роботи з налаштування та запуску системи. У разі з продуктами 1С ці цифри можуть бути дуже вражаючими. Розробка пропонованого продукту на платформі ASP.NET обійдеться значно дешевше. Архітектура 1С Бітрікс, як власне і багато інших продукти 1С, викликає масу питань. Досвід показує, що система 1С Бітрікс дуже вимоглива до ресурсів сервера. При розробці на ASP.NET можна на етапі налагодження виділяти ресурсомісткі механізми і приймати заходи для оптимального використання ресурсів сервера. Тому можна використовувати менш витратний хостинг. При спробах внесення змін в 1С Бітрікс необхідно документувати всі зміни і враховувати їх при оновленні системи, або відмовлятися від останніх. При розробці на ASP.NET завдання в

значній мірі спрощується. Крім перерахованого вище 1С Бітрікс має масу вад які, на жаль, живуть в продукті вже тривалий час.

2.6.2 PHP

PHP (Hypertext Preprocessor) - мова для написання серверних скриптів. Інтерпретатор мови безкоштовний, з відкритим вихідним кодом, створені версії для різних веб-серверів - перш за все, для Apache і Internet Information Services. Актуальна на сьогоднішній (2016 рік) день версія - PHP 7.0.0. У інтерпретаторі з'явилася вбудована підтримка PHP в Internet Information Services 7-й версії - раніше PHP можна було використовувати виключно як зовнішнє розширення.

PHP легше для вивчення, тому і багато людей можуть досить-таки непогано писати на ньому, але до професійних розробникам віднести можна аж ніяк невелику частину програмістів. Якщо брати чисельність дійсно професіоналів, то приблизно буде однакове кількість (аналогічно платформі 1С). З цієї причини розробники на платформі ASP.NET з великою ймовірністю буду мати більш високою кваліфікацією ніж розробники на PHP.

Є думка, що вартість хостингу для ASP.NET значно дорожче. Різниця вартості хостингу була актуальна кілька років тому, на даний момент ціни вирівнялися, особливо якщо порівнювати ціну / якість надаваних послуг хостингу. Для великих проектів необхідно брати віртуальні або виділені сервери.

При оцінці швидкості роботи і розробки зазвичай воліють PHP. В основному через побоювання, що проект буде повільно працювати. Це в корені НЕ вірно. Якщо подивитися на невеликі і малонавантажених проекти, то відчутної різниці кінцевий користувач НЕ відчує. Якщо ж розглянути великі і високонавантажених проекти, то ASP.NET MVC буде впевнено вигравати в продуктивності.

Швидкість розробки залежить від бізнес вимог замовника, використання готових рішень. Наприклад, якщо потрібно написати блог, то простіше застосувати Wordpres, розгорнути його і щось скорегувати в функціоналі. Це рішення буде дешевше і швидше, ніж писати з нуля на ASP.NET.

Великі проекти, які пишуться з нуля, звичайно швидше реалізовуються на ASP.NET MVC. Уразливість в фреймворку ASP.NET MVC виявляються рідше ніж

в PHP і закриваються набагато швидше. В ідеалі ASP.NET MVC більш безпечний захищений фреймворк.

2.7 Висновки до розділу

Розглянувши три різних продукту був зроблений висновок, про те, що в даному випадку розробка, використовуючи ASP.NET цілком виправдана з перспективами на подальший розвиток інформаційної системи. Почавши розробку, застосовуючи правильно підібрані технології та інструменти, закладається міцний фундамент для подальшого розвитку проекту. Вибравши самостійну розробку на ASP.NET можливо старт інформаційної системи відкладеться, але в подальшому це окупиться можливостями масштабування, великий гнучкістю, підвищеної безпекою і меншою вартістю розробки та підтримки в порівнянні з аналогами.

Також після аналізу можна зробити висновок що PHP і ASP.NET MVC - практично рівноцінні. Для великих, високонавантажених проектів, які потрібно швидко і якісно розробити - це однозначно ASP.NET MVC. Використовуючи власний рущій розроблений на ASP.NET і правильний підхід до побудови архітектури рішення ми гарантуємо подальший розвиток проекту і розширення його функціоналу без великих зусиль.

3 ОПИС СИСТЕМИ

3.1 Структура системи

Структура сайту - систематизація інформації та навігації по ній з метою адаптувати відвідувачів до успішного знаходження товару. Добре продумана

грамотна структура сайту гарантує, що користувачі витратять менше часу на пошук потрібного інструменту та його аналогів.

Розробка структури сайту ведеться з урахуванням найбільш важливої інформації з точки зору просування товарів або послуг на інтернет-ринку. У процесі створення структури сайту, або оптимізації структури вже існуючого, необхідно акцентувати увагу користувачів саме на тій інформації та розділах сайту, які найбільш важливі відповідно до позиціонуванням на ринку, що просуваються товарів або послуг.

Структура інформаційної системи повинна бути проста і інтуїтивно зрозуміла, і складатися з програмної частини, клієнтської частини і адміністрування, як показано на рис.3.1.

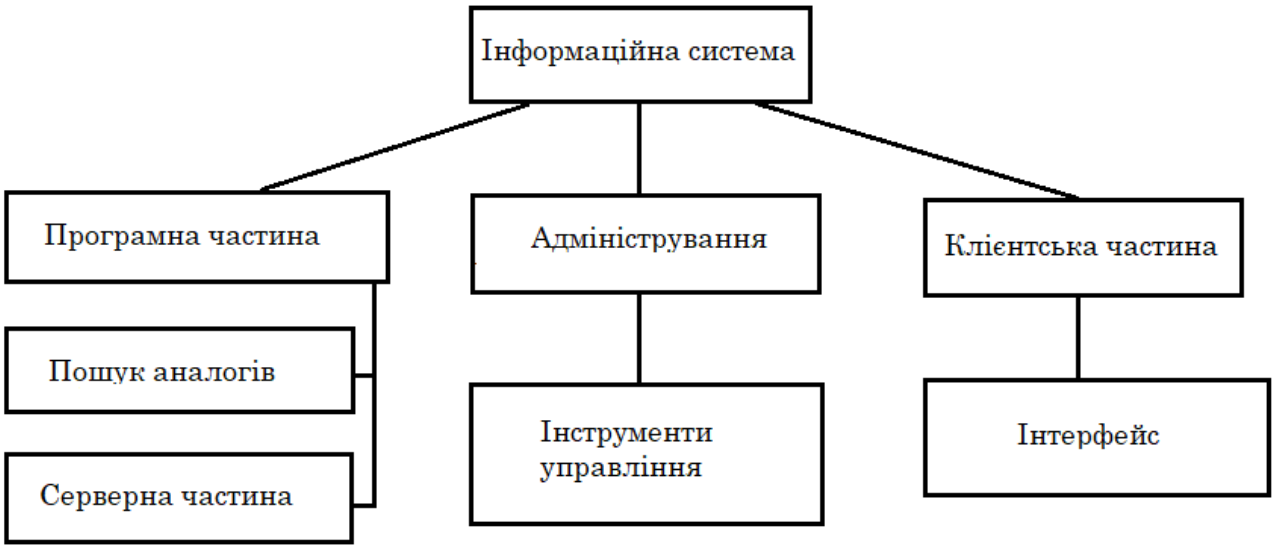

Рисунок 3.1 – Структура інформаційної системи

Програмна частина структури інформаційної системи (ІС) розглядається як взаємозв'язок серверної та операційної частини.

В частині пошуку аналогу або операційній частини розглядається середовище розробки ІС. В ній міститься також розділ для пошуку обладнання за критеріями. Це дає змогу підбирати обладнання. деталі, які вийшли з ладу. До основних критеріїв слід віднести категорію товару, виробник, тип, матеріал та

особливості. В останньому критерії можна відмітити особливості притаманні для певних груп товарів, наприклад для викруток існує критерій тип наконечника, але критерій не актуальний для вибору датчиків.

Серверна частина містить технології, які підтримуються Для створення і первинного тестування ІС буде використовуватися локальний сервер, що значно спростить налагодження ІС.

В адмініструванні будуть міститися основні налаштування ІС, такі як:

– форми реєстрації клієнта в Інтернет-ресурсі;
– пакування та доставка товару;
– редагування каталогу товарів;
– управління зареєстрованими клієнтами;
– управління оформленими замовленнями;
– незавершені замовлення;
– резервне копіювання БД.

У клієнтської частини архітектури розробляється максимально зручний інтерфейс. Важливим фактором є зворотна зв'язок, що дозволяє користувачеві залишити свою думку, відгук про товар або послугу або про якість обслуговування в цілому.

Проаналізувавши роботу вже працюють ІС, був зроблений висновок про те, що обов'язково буде реалізовано в проекті.

– каталог товарів буде оформлений так, щоб користувач міг без праці знаходити цікавий для його товар і мати можливість вивчити інформацію про ньому;

– товари будуть розділені по групах, забезпечимо можливість пошуку товарів по частині назви і опису. Для кожного товару буде передбачено короткий і повне опис, плюс кілька фотографій;

– при оформленні замовлення покупець вносить контактну інформацію: логін, пароль, адресу доставки, телефон та ін. Після реєстрації покупцеві буде відправляється по електронній пошті лист з збереженими даними;

– в електронній торгівлі будуть передбачені і інформаційні розділи (контакти, новини, статті та інша корисна інформація).

3.2 Розробка бази даних

При реалізації проекту важливу роль відіграє грамотне складання структури бази даних проекту [12]. При створенні таблиць бази даних були враховані основні положення тематики проекту, а саме ІС. При роботі з проектом всі користувачі будуть розділені на 3 основні групи, такі як зареєстровані, незареєстровані користувачі і адміністратори сайту. Кожній групі користувачів відповідають певні права: можливість оформлення замовлення, можливість редагування каталогів товарів, можливість додавання тегів до товару та ін. Для зберігання інформації про пропонованому товарі використовується таблиця «dbo.Product». В даній таблиці зберігається повне опис товару, ціна, категорія, а так само кількість штук.

Щоб спростити взаємодію користувача з кошиком, було вирішено вивести окрему таблицю dbo.Basket, в якій буде зберігатися інформація про товари, доданих до кошика. Необхідно розуміти, що користувач може з будь- якої технічним причинам покинути сайт, але при наступному відвідуванні ІС, йому доведеться знову шукати, вибирати і додавати вже вибрані раніше товари. Щоб уникнути такої ситуації нам слід при черговому вході користувач, лише згідно з таблицею його кошик і в разі виявлення доданих раніше товарів, просто відобразити кошик користувача. Наступна таблиця - таблиця замовлень «dbo.Order» необхідна для операцій безпосередній продажу і доставки товару користувачеві. Після опису ключових моментів проектування бази даних, необхідна реалізація бази даних проекту.

3.2.1 Проектування бази даних

База даних розробляється ІС складається з 9 таблиць рис. 3.2

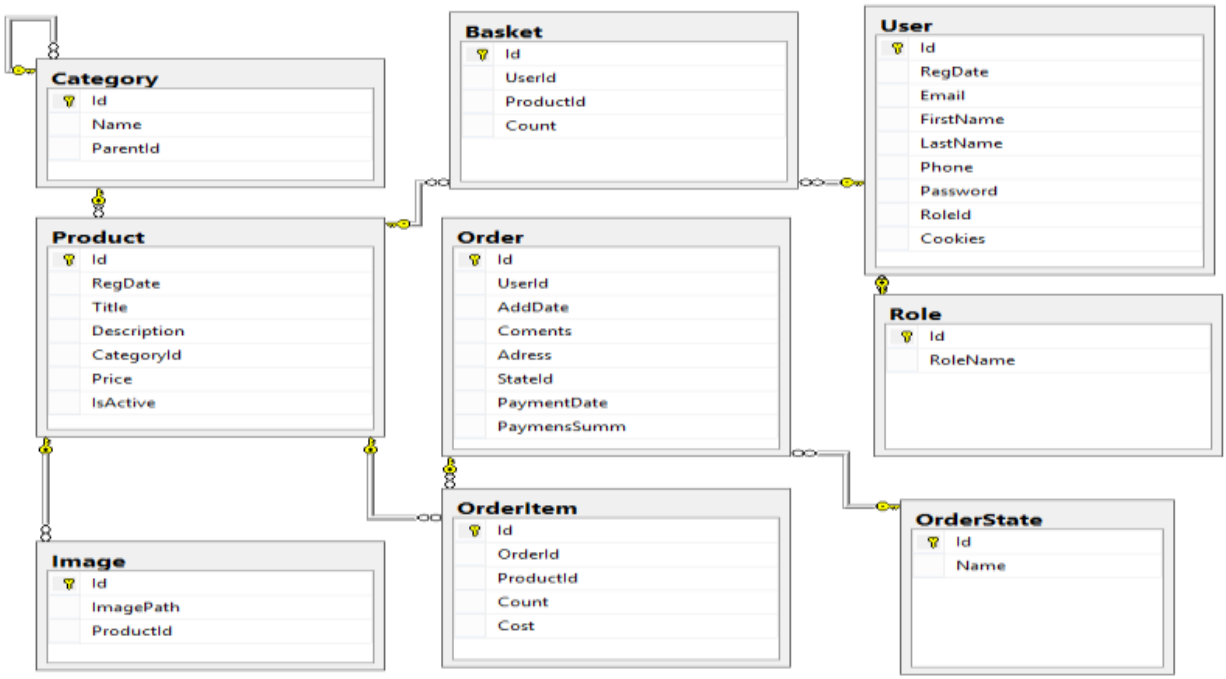

Рисунок 3.2 – Діаграма бази даних.

3.2.2 Реалізація бази даних

Таблиця dbo.User (табл.3.1) містить унікальний номер користувача і його дані, що надаються при реєстрації.

Таблиця 3.1 – Структура dbo.User

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
RegDate	datetime	Дата реєстрації
Email	varchar (50)	Адреса електронної пошти
FirstName	nvarchar (50)	Ім'я
LastName	nvarchar (50)	Прізвище
Phone	char (15)	Телефон
Password	nchar (36)	Пароль
RoleId	bigint	Роль користувача
Cookies	nchar (36)	Кукі користувача

Таблиця dbo.Basket (таблиця 3.2) необхідна для зберігання даних про товари, які користувач збирається купити. Коли користувач додає нову річ собі в «кошик», то автоматично відбувається запис в дану таблицю.

Таблиця 3.2 – Структура dbo.Basket

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
UserId	bigint	Ідентифікатор користувача
ProductId	bigint	Ідентифікатор товару
Count	Int	Кількість доданих в кошик однакових товарів

Таблиця dbo.Product (таблиця 3.3) містить дані про продуктах, виставлених на продаж.

Таблиця 3.3 – Структура dbo.Product

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
RegDate	datetime	Дата реєстрації
Title	varchar (200)	Відображуване найменування
Description	text	Опис
CategoryId	bigint	Ідентифікатор категорії товару
Price	float	Ціна
IsActive	int	Статус товару

Таблиця dbo.Order (таблиця 3.4) представляє собою таблицю замовлень. В даній таблиці відображаються записи про оформлених користувачами замовленнях.

Таблиця 3.4 – Структура dbo.Order

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
UserId	bigint	Ідентифікатор користувача
AddDate	datetime	Дата замовлення
Coments	text	Коментар до замовлення
Adress	text	Адреса доставки
StateId	bigint	Ідентифікатор стану замовлення
PaymentDate	datetime	Дата проведення оплати
PaymentSumm	float	Сума оплати

Таблиця dbo.OrderItem (таблиця 3.5) представляє собою таблицю позицій замовлень. В даній таблиці відображувати всю інформацію про товари, які користувачі замовили

Таблиця 3.5 – Структура dbo.OrderItem

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
OrderId	bigint	Ідентифікатор користувача
ProductId	bigint	Ідентифікатор обраного товару
Count	int	Кількість що залишилися днів до видалення замовлення.
Cost	float	

Відмінність таблиць «Замовлення» і «Кошик», полягає в тому, що в одній таблиці реєструються замовлення користувачів, іншими словами на кошику була натиснута кнопка «Купити», а в іншій таблиці відповідно зберігається інформація про товари ще не придбання, але вже доданих в кошик.

З огляду на правила нормалізації бази даних, було вирішено створити додаткові таблиці: Категорії, Зображення, Стану замовлень, Ролі користувачів. Кожна таблиця являє собою список. Наприклад, будь-який товар може відповідати одній з можливих категорій. Категорії формуються адміністратором сайту і утворюють ієрархічну структуру, яка відображається у вигляді багаторівневого меню. Таким чином, використовуючи довідкові таблиці легко управляти як вмістом списків, так і записами, які використовують значення списків у вигляді посилань.

3.3 Опис розроблених частин проекту

Розроблений проект можна розділити на кілька функціональних частин, які в сукупності становлять повноцінний ІС. Далі детально розглянемо особливості функціонального призначення виділених частин проекту.

3.3.1 Реєстрація та аутентифікація користувачів

Сторінка «Реєстрації» представляє собою одну загальну панель, на якій розташовані питання і поля для введення відповідей. Для перевірки коректності введених даних використовуються пояснюють написи, які вказують на тип помилки і що повинен користувач виправити при відповіді на питання.

Передбачена кнопка перевірки облікової записи (логін) на зайнятість. Якщо користувач ввів логін, який вже присутній в системі, то на формі з'явиться повідомлення про те, що даний Login можна використовувати для реєстрації. Використовується валідатори для підтвердження коректності введення даних. Валідатори - це стандартні елементи управління в ASP.NET, які представляються користувачеві в вигляді написів, що свідчать про неправильному або некоректному

введенні даних на сторінці. Існує кілька типів валідаторів, деякі з них містять у собі регулярне вираз або правило, по якому визначається правильність або достовірність введених даних.

Для кожного користувача додатку створюється своя власна сесія зі своїми власними значеннями. Для ідентифікації користувачів ASP.NET використовує 120-бітний ключ, іменований SessionID і складається тільки з ASCII- символів, які допустимі для використання в URL. При надходженні запиту від користувача, який НЕ має SessionID в URL запиту, ініціюється створення нової сесії. При цьому відбувається генерація нового унікального SessionID. Після чого виникає подія Session_Start веб-додатки.

Сесія існує до тих пір, поки користувач працює з веб- додатком і ще трохи після. Для цього в настройках аплікації виставляється таймаут сесії, який за замовчуванням становить 20 хвилин. Тобто якщо користувач в протягом 20 хвилин НЕ зробив ні одного запиту до додатку, то сесія цього користувача знищується. При цьому виникає подія Session_End додатки. У поточному проекті час сесії було укорочено до 5 хвилин з міркувань безпеки. В реалізованої підсистемі сесія використовується для зберігання об'єкта класу User, в якому зберігаються дані про поточного користувача системи.

При спробі входу в систему, створюється запит на вибірку з бази даних користувача з зазначеним логіном і паролем, далі відбувається перевірка на існування запису про таке користувача, а саме перевіряється, чи існує id у обраного користувача.

Так само сесія використовується для користувачів вже пройшли аутентифікацію. Використовуючи дані про користувача, формується запит до бази даних про наявність товару, раніше покладеного в кошик користувачем. При виявленні таких записів користувач може побачити свою корзину вже заповнену вибраним раніше товаром, що дозволить заощадити час на пошук товару. Загальний вигляд сторінки реєстрації представлений на рис. 3.3

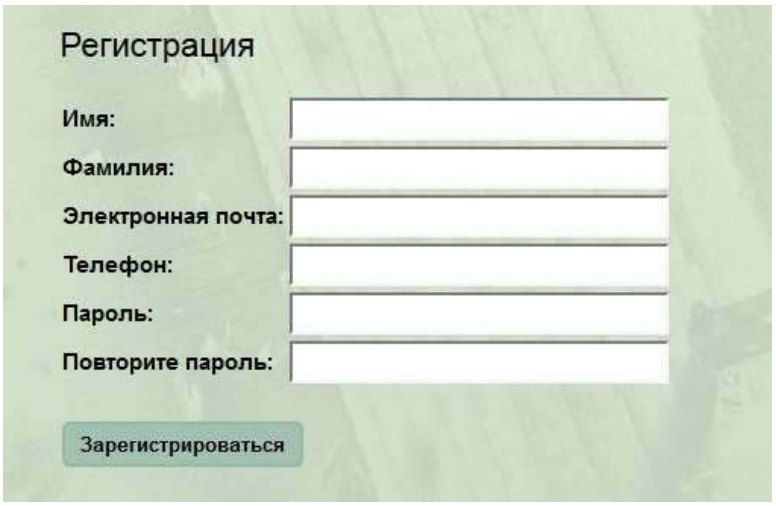

Рисунок 3.3 – Сторінка «Реєстрації»

Якщо реєстрація пройшла успішно, то користувача повертають на вікно «Вхід у систему». Аутентифікація виконується за допомогою форм. Якщо користувач успішно пройшов перевірку, необхідно просто повідомити інфраструктурі ASP.NET про успіх операції (викликавши метод класу FormsAuthentication), і виконуюча система автоматично встановлює аутентифицирующей cookie-набір (який в дійсності містить мандат) і переадресує користувача на запитану їм сторінку. З цим запитом виконуюча система визначає, що аутентифицирующей cookie-набір з мандатом доступний і відкриває доступ до сторінки. Цей процес показаний на рис. 3.4

Аутентификація форм - привабливий вибір для розробників по декільком причинам:

– повний контроль над кодом аутентифікації;
– повний контроль над зовнішнім виглядом форми реєстрації;
– працює з будь-яким браузером;
– дозволяє вибирати спосіб зберігання інформації про користувачів.

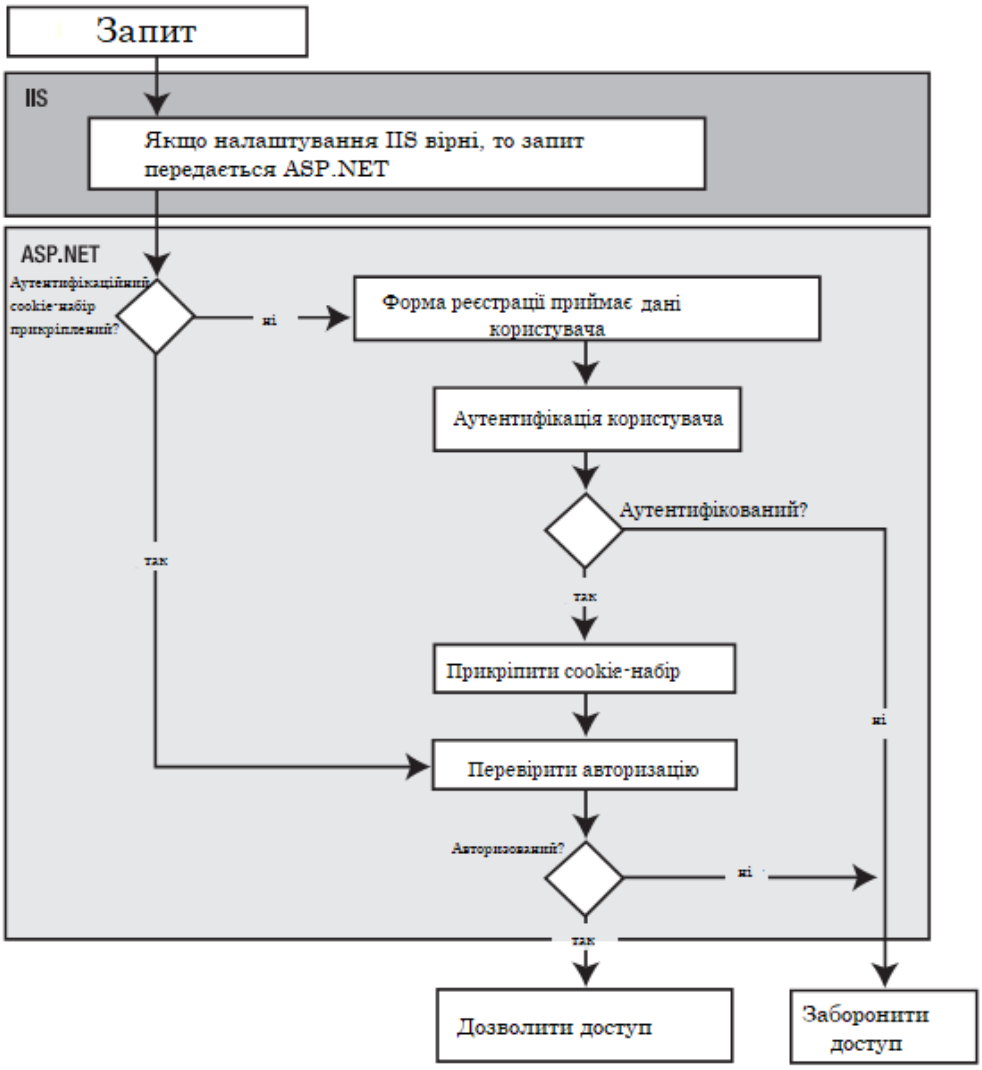

Рисунок 3.4 – Процес аутентифікації

Оскільки аутентифікація форм реалізована повністю всередині ASP.NET, розробник отримує повний контроль над виконанням аутентифікації. Немає необхідності покладатися ні на яку-небудь зовнішню систему, як це має місце при аутентифікації Windows або Passport. Так само з'являється можливість самостійно налаштувати поведінку аутентифікації форми під свої потреби. Іншими словами, розробнику надається можливість оформляти вхідну сторінку реєстрації як завгодно. Гнучкість зовнішнього вигляду недоступна при інших методах аутентифікації. Аутентифікація Windows вимагає, щоб браузер запитував призначене для користувача посвідчення, а Passport-аутентифікація вимагає, щоб

ви залишили свій Web-сайт і відвідали сайт Passport для введення своєї реєстраційної інформації (посвідчення) [4].

Сторінка «Вхід в систему», зображена на рис.3.5 виконана так само з використанням стандартних елементів управління ASP.NET, таких як textbox, label, validator. Для входу в систему користувачеві необхідно ввести свій логін і пароль, при помилці введення або невідповідність введених даних користувач побачить повідомлення «Такого користувача немає в системі». Так само користувачеві надається швидкий перехід до сторінки «Реєстрації».

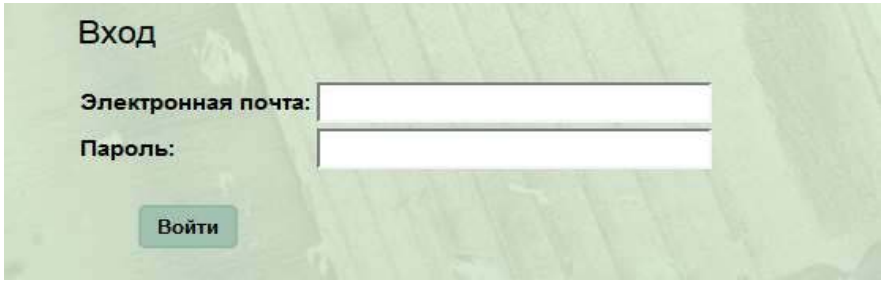

Рисунок 3.5 – Сторінка «Вхід в систему»

3.3.2 Детальний перегляд товару

Детальний перегляд товару реалізований в вигляді відображення зовнішнього вигляду і його детального опису із зазначенням ціни. Для додавання товару в кошик необхідно натиснути відповідну кнопку на екрані. Засоби ASP.Net дозволяє створювати насичені, візуально привабливі веб-сторінки, що працюють в різних браузерах, пристроях і настільних операційних системах (наприклад, Apple Macintosh).

3.3.3 Каталог товарів

Каталог товарів є розробленим призначеним для користувача елементом управління і може використовуватися в інших додатках. Він складається з таблиці, осередки якої оформлені відповідним чином. Всі використовувані стилі додатки зберігаються в окремому файлі «Default.css».

Відмінною функціональною особливістю є реакція на натискання кнопки «Додати в корзину». При натисканні на цю кнопку обраний товар додається в кошик і при цьому не відбувається перезавантаження сторінки. Досягається це використанням технологій AJAX, що дозволяє не перезавантажувати всю сторінку за допомогою будь-яких маніпуляцій, а оновлювати тільки її частину. Інструкції для роботи з елементами таблиці (товарами) записані в JavaScript-файлах, які завантажуються на сторону клієнта при вході на сайт.

3.3.4 Кошик користувача

Даний елемент, як і каталог товарів, так само реалізований в вигляді окремого, легко інтегрувального призначеного для користувача елементу управління. Кошик являє собою блок з заголовком, в якому розміщений список обраних товарів, загальна вартість кошика і посилання для переходу на сторінку оформлення замовлення. Приклад роботи кошика користувача зображений на рис. 3.6.

Рисунок 3.6 – Кошик користувача

Додавання товарів ведеться шляхом натискання користувачем на кнопку «В кошик», розташовану в осередку з інтересуемого товаром, або шляхом перетягування зображення товару в область кошика. Функція додавання товару так само реалізована з використанням AJAX, що не тягне за собою повної перезавантаження сторінки після скоєного користувачем дії. Таким чином, користувач може скільки завгодно довго додавати і видаляти товари з кошика без

небезпеки великого витрати трафіку. Так само робота кошика супроводжується з використанням cookie. Coolie - застосовуються для збереження даних на стороні користувача, на практиці зазвичай використовується для:

– відстеження стану сесії доступу користувача;
– зберігання персональних переваг і налаштувань користувача;
– аутентифікації користувача;
– ведення статистики про користувачів.

У технічному плані cookie становлять собою фрагменти даних, спочатку відправляються веб-сервером браузеру. При кожному наступному відвідуванні сайту браузер пересилає їх назад сервера. Без них кожен перегляд веб-сторінки є ізольованим дією, НЕ пов'язаним з переглядом інших сторінок того ж сайту, за допомогою ж cookie можна виявити зв'язок між переглядом різних сторінок.

Таким чином, використання cookie-наборів дозволяє швидко відновити інформацію про вміст кошика, якщо користувач з будь- якої причини покинув сайт. Наприклад, в разі проблем з провайдером користувач відключився від мережі і відновив сеанс через кілька хвилин, якщо він побачить порожню корзину, йому доведеться знову шукати раніше обраний товар і додавати його в кошик. Такі дії можуть призвести до втрати частини відвідувачів Інтернет-ресурсу в зв'язку з незручностями сервісів.

3.3.5 Система пошуку аналогів товару

Пошук необхідної користувачеві інформації є однією з найбільш важливих функцій будь-якого проекту. У розробленій ІС існує два типи пошуку: пошук по введеним користувачем словами без урахування додаткових фільтрів та критеріїв.

Як згадувалося раніше, для пошуку товару необхідний повнотекстовий пошук (full text search). Повнотекстові запити виконують лінгвістичний пошук в текстових даних в повнотекстових індексах шляхом обробки слів і фраз в відповідно з правилами даного мови, наприклад англійської або японської.

Повнотекстові запити можуть включати прості слова і фрази або кілька форм слова або фрази. Повнотекстовий пошук підтримує кілька мов завдяки використанню наступних лінгвістичних компонентів: кошти розбиття за словами і

парадигматичні модулі, списки стоп-слів, що містять стоп-слова (звані також пропускати словами), і файли тези. Для файлів тезауруса і в деяких випадках для списків стоп-слів потрібно налаштування, яке виконується адміністратором бази даних. Даний файл тезауруса підтримує всі повнотекстові індекси, які використовують відповідну мову, а даний список стоп-слів може бути пов'язаний з будь-яким необхідним числом повнотекстових індексів.

Для реалізації повнотекстового пошуку необхідно використовувати окреме подання в структурі бази даних, так як деякі поля таблиці з інформацією про продукт зберігають числа (ключі) з вибіркою з таблиць довідників. У поданні зберігається вся текстова інформація по кожній записи з таблиці «dbo.Product», первинним ключем так само є унікальний ідентифікатор (Id) запису.

При використанні пошуку також можна використовувати критерії або фільтри для підбору аналогів обладнання. В даній роботі використовується такі критерії як категорія товару, виробник, тип, матеріал та особливості рис. 3.7. В останньому критерії можна відмітити особливості притаманні для певних груп товарів, наприклад для викруток існує критерій тип наконечника, але критерій не актуальний для вибору датчиків. Після сортування товару вибрані дані підставляються в якості ресурсу для розробленого елементу «Каталог продукції».

Рисунок 3.7 – Система пошуку аналогів товару

3.3.6 Меню інформаційної системи

При розробці меню проекту була врахована особливості тематики проекту, а саме ІС електроінструменту. У більшості ІС меню сайту крім функцій навігації, виконує так само функції пошуку або фільтра для пропонованої користувачеві продукції. У розробляється системі меню виконано в вигляді ієрархічної структури формується адміністратором при додаванні товарів.

Приклад такого меню зображений на рис. 3.7.

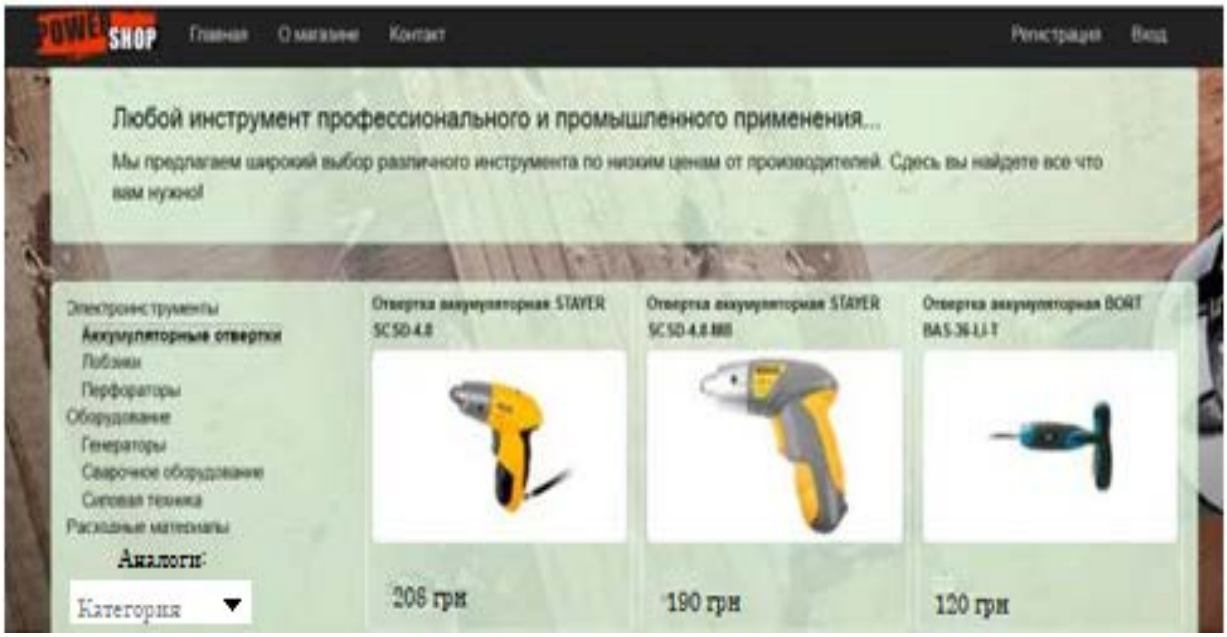

Рисунок 3.8 – Загальний вигляд меню проекту

Для реалізації даного меню була обрана технологія jQuery. Саме ця технологія дозволила створити досить красиве, витончене і відповідає основним вимогою проекту меню.

3.3.7 Опис процесу експлуатації

Основні часто використовувані функції системи призначені безпосередньо для покупців ІС. Виходячи з цього, розглянемо схему дій користувача при відвідуванні розробленого web-ресурсу.

На рис.3.8 наведена блок-схема реєстрації користувача, а на рис.3.9 блок-схема алгоритму оформлення.

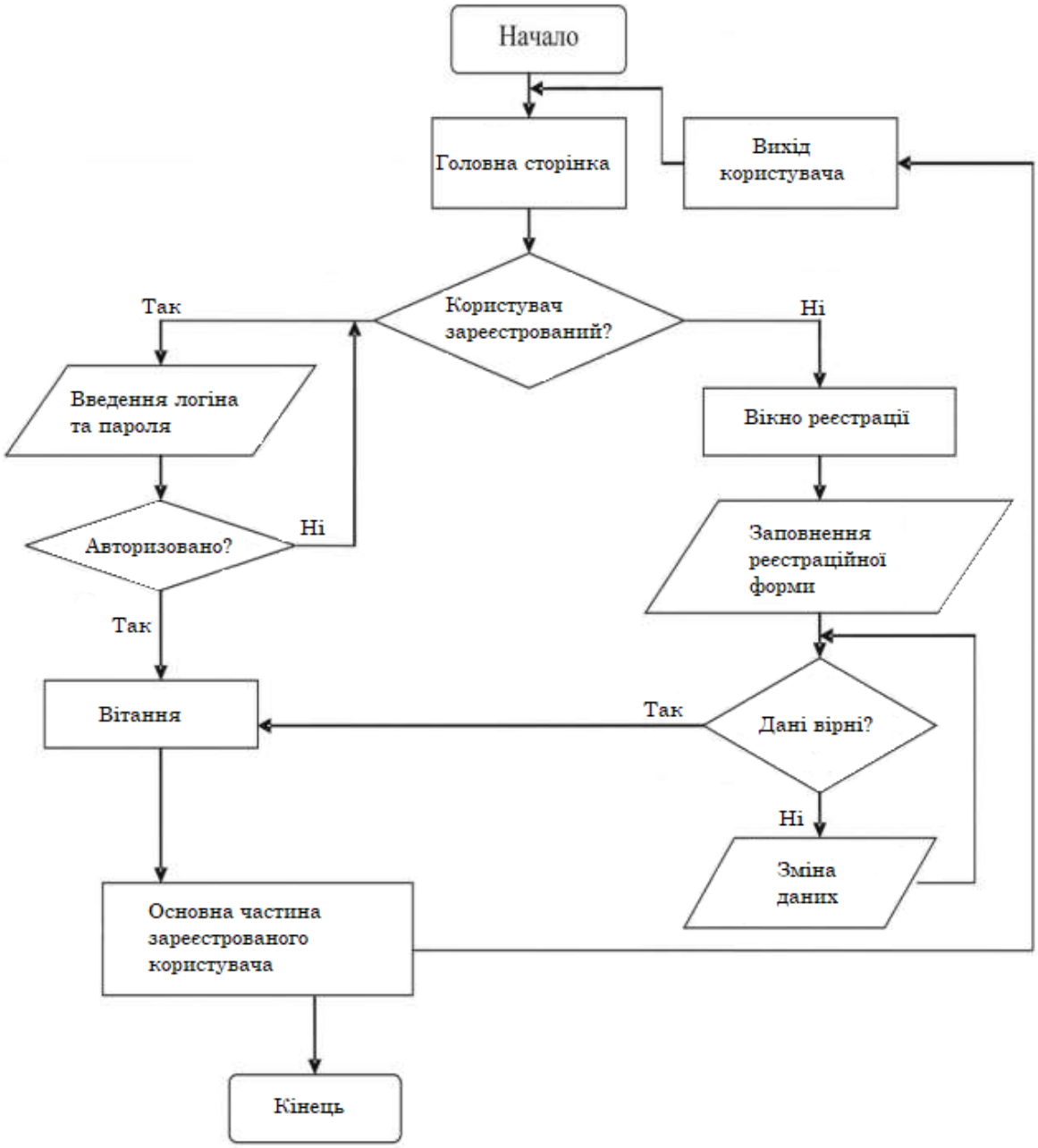

Рисунок 3.8 – Блок-схема алгоритму реєстрації користувача

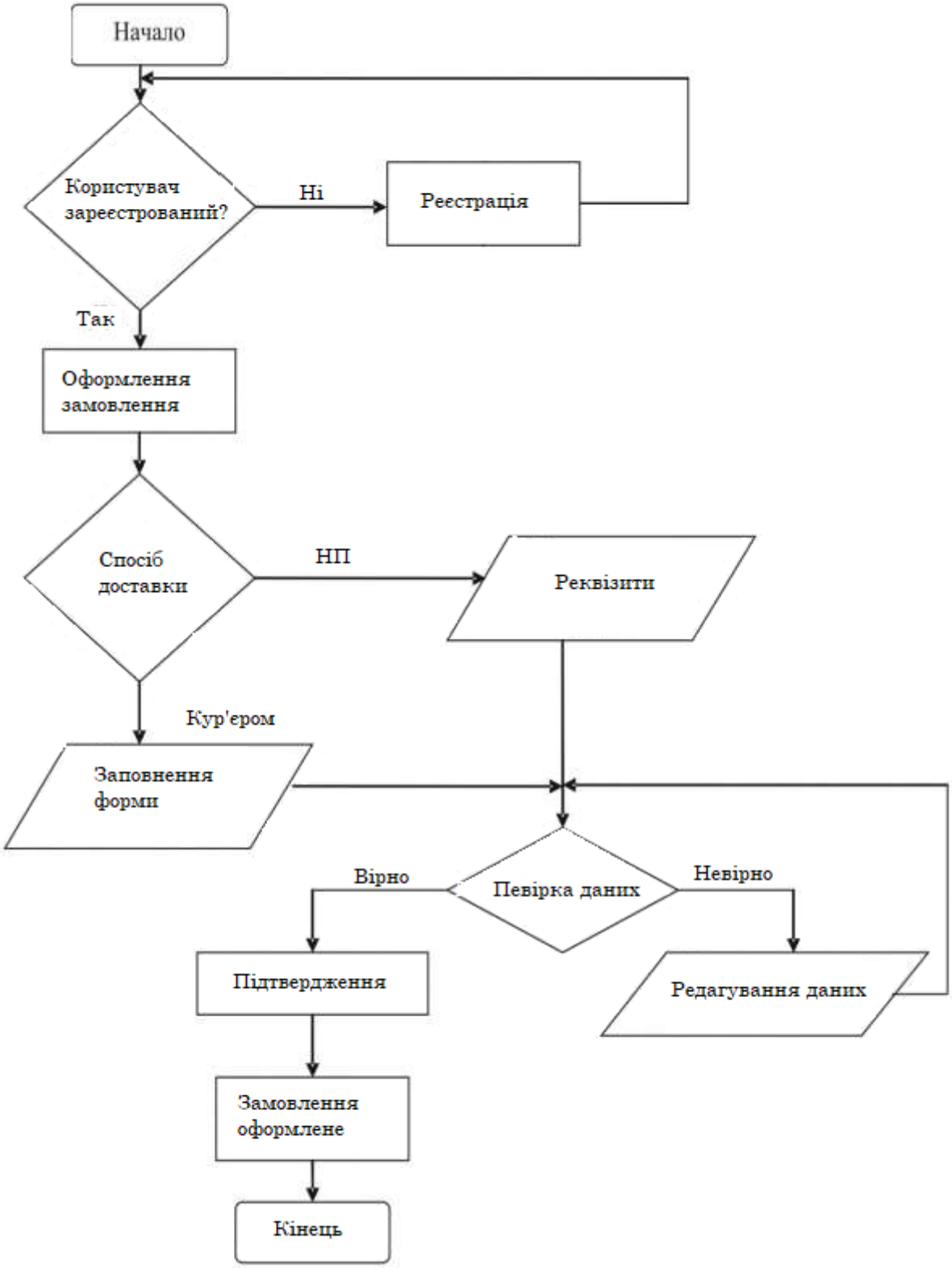

Рисунок 3.9 – Блок-схема алгоритму оформлення замовлення

3.4 Висновки до розділу

В розділі побудовано структуру інформаційної системи та розроблена структура бази даних. Розробка структури сайту ведеться з урахуванням найбільш важливої інформації з точки зору просування товарів або послуг на інтернет-ринку. У процесі створення структури сайту, або оптимізації структури вже існуючого, необхідно акцентувати увагу користувачів саме на тій інформації та розділах сайту, які найбільш важливі відповідно до позиціонуванням на ринку, що просуваються товарів або послуг.

Було спроєктовано базу даних для коректного функціонування системи. Було проведено опис виділених частин системи. Приведено алгоритми реєстрації користувача та оформлення замовлення.

4 ОПИС ІНТЕРНЕТ-РЕСУРСУ АНАЛОГІВ ТА ЇХ РЕАЛІЗАЦІЇ

При відвідуванні сайту, користувач потрапляє на головну сторінку сайту (default page). На якій аналоги обраного інструментарію, доступного користувачеві.

На головній сторінці користувачеві надається перегляд першої сторінки каталогу продукції, можливість використання пошуку і додавання товарів в корзину.

4.1 Основні операції доступні користувачеві

В даному керівництві наведені основні дії користувача, при відвідуванні розробленої ІС. Існує два основних типи користувачів: зареєстровані і

незареєстровані. Незареєстрованих користувачів обмежують лише в покупці товару, так як для достовірності введених даних про доставку йому необхідно зареєструватися. Користувачам, які пройшли процедуру реєстрації, можуть призначити певну роль. За ролям користувачі діляться на адміністраторів, менеджерів і користувачів ІС.

4.1.1 Реєстрація в системі

Реєстрація в системі є не обов'язковою, ніхто відразу не направить користувача реєструватися. Користувач може вільно переміщатися по ІС, переглядати цікавить його товар, читати новини про нових акціях і пропозиціях. Однак при оформленні замовлення користувач повинен буде пройти процедуру реєстрації. Так само зареєстровані користувачі можуть відвідати сторінку з персональними даними та при бажанні змінити їх, а так само переглянути історію своїх покупок. Отже, перейшовши на сторінку реєстрації, користувач побачить форму зображену на рис. 4.2.

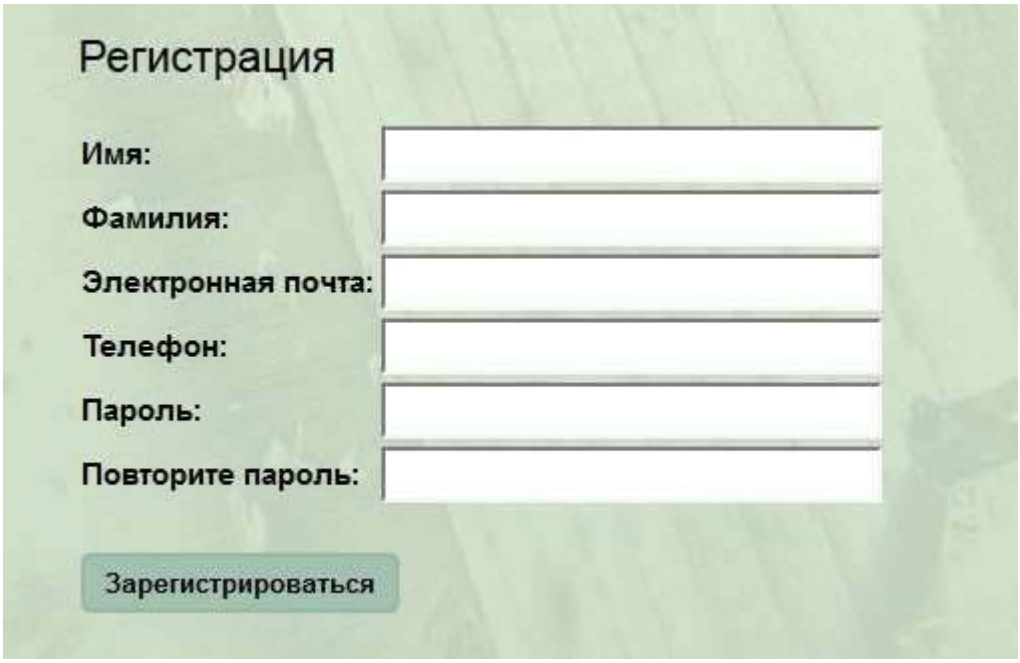

Рисунок 4.1 – Сторінка «Реєстрації»

4.1.2 Вхід в систему

Після реєстрації користувач отримує можливість перейти на сторінку входу в систему і авторизуватися. Сторінка входу в систему зображена на рис.4.2.

Рисунок 4.2 – Сторінка «Вхід в систему»

4.1.3 Перегляд каталогу аналогів товарів

Найважливішою частиною будь-якої ІС є «вітрина», то є каталог продукції, оформлений відповідним чином. Загальний вигляд каталогу аналогів зображений на рис.4.3

Картка кожного товару складається з трьох полів:

– текстовий інформаційний блок;
– зображення товару;
– ціна товару;
– дії користувача.

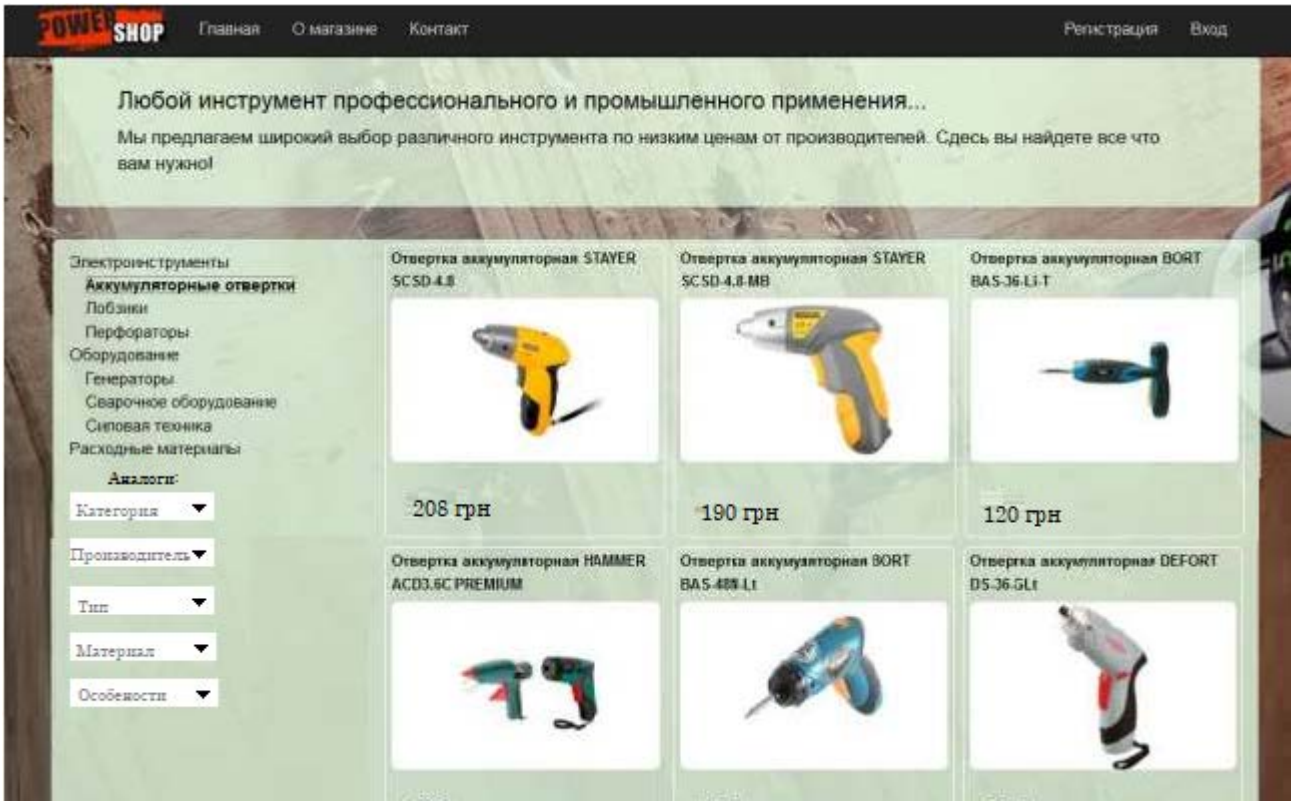

Рисунок 4.3 – Каталог товарів

Перше поле відображає назва і продукції. Так користувач може ознайомитися з короткими даними про товар, не чекаючи завантаження зображення продукції.

У другому полі розташовується зменшене зображення переднього виду товару. Клікнувши на зображенні товару користувач потрапляє в розділ детального опису обраного товару з збільшеним зображенням обраного продукту.

Під зображенням товару розташовуються інформація про ціну і клавіша додавання товару в корзину.

Якщо кількість товарів НЕ поміщається на першій сторінці каталогу товарів, користувач може переміщатися по сторінках каталогу за допомогою текстових покажчиків «вперед», «назад» і покажчиків у вигляді цифр з номерами сторінок.

4.1.4 Детальний перегляд

Для більш детального перегляду товару користувачеві необхідно натиснути на відповідне зображення. Після натискання користувача перенаправляють на

сторінку детального перегляду товару. Сторінка детального перегляду товару зображена на рис.4.4.

Рисунок 4.4 – Сторінка детального перегляду товару

Даний модуль дозволяє переглянути всю інформацію про товар, а так само додавати товар в корзину.

4.1.5 Кошик користувача

Кошик користувача є невід'ємною частиною будь-якої ІС. Користувач може додавати в кошик цікавить його товар і видаляти. Видалення товару відбувається після натискання на Рисунок з відповідним зображенням. Кошик користувача зображена на рис.4.5.

Рисунок 4.5 – Кошик користувача

Будь-які дії користувача над кошиком відбуваються як би в фоновому режимі, то є сторінка НЕ оновлюється повністю. Оновлюється лише вміст кошика. Так само користувач може переглянути загальну вартість всіх товарів і перейти на сторінку оформлення замовлення.

4.1.6 Оформлення замовлення

Після того як користувач визначився з вибраними товарами йому слід перейти на сторінку оформлення замовлення для здійснення покупки. Загальний вигляд сторінки оформлення замовлення зображений на рис.4.6.

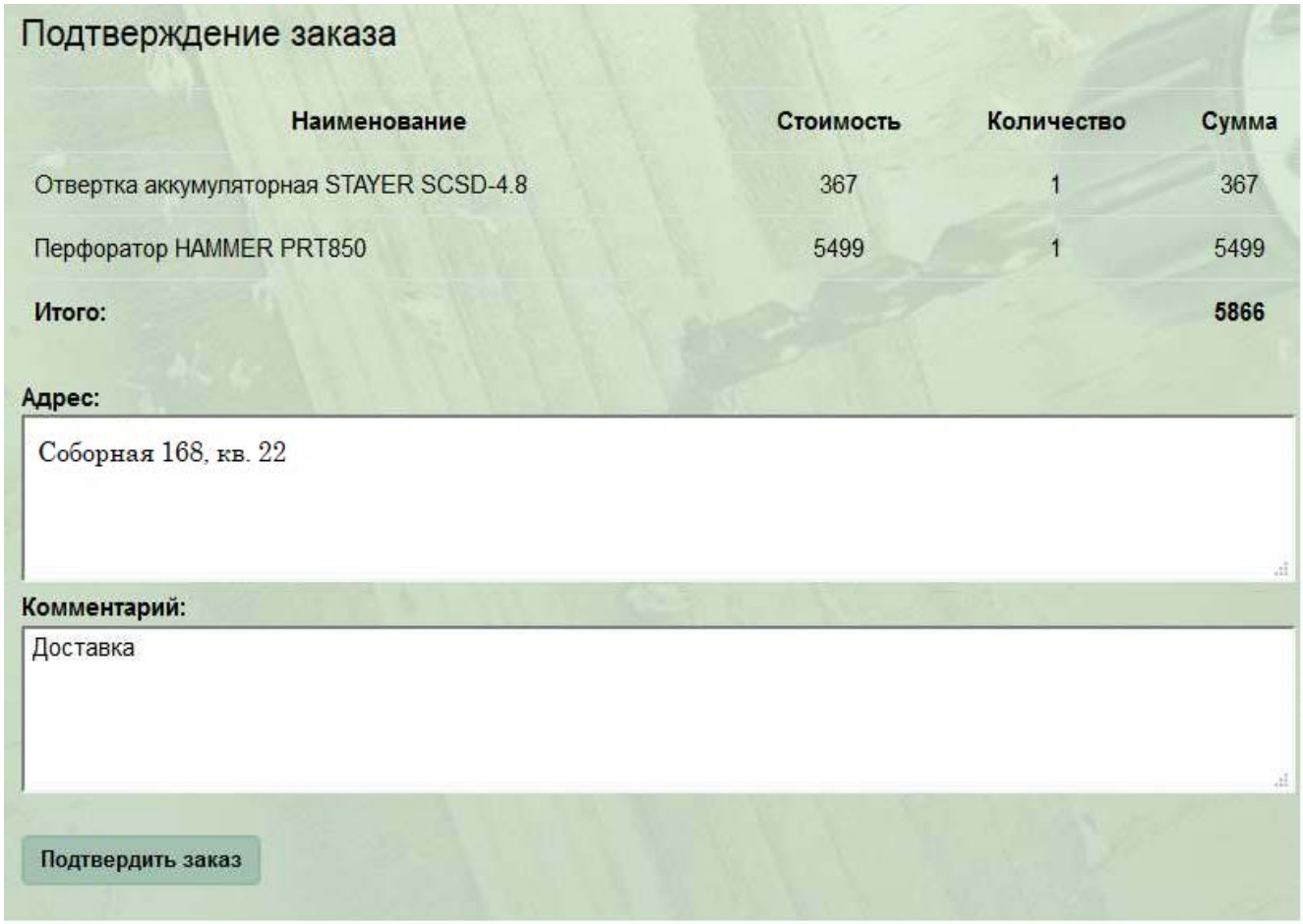

Рисунок 4.6 – Сторінка оформлення замовлення

З малюнка видно, що на сторінці розташовується вміст кошика користувача. Дана сторінка є останнім кроком дозволяє редагувати вміст кошика. Переконавшись в правильності даних користувач переходить до оплати.

4.1.7 Особистий кабінет

Зареєстровані користувачі мають право відвідування сторінки

«Особистий кабінет». Сторінка «Особистий кабінет» містить повну інформацію про користувача і історію покупок. Користувач має право змінювати свої дані, наприклад телефон та ін.

4.2 Опис розділу адміністратора

Для адміністрування ІС існує відповідний розділ, він доступний тільки адміністраторам сайту. В даному розділі знаходиться стандартний набір сторінок дозволяють адміністратору:

– додавати, видаляти, змінювати інформацію про товар;

– редагувати дані користувачів;

– призначати ролі зареєстрованим користувачам;

– змінювати інформацію про ІС;

– переглядати список проданих товарів;

– додавати, видаляти, змінювати категорії, підкатегорії і вміст інших таблиць-довідників.

На рис.5.1 відображена сторінка редагування товарів представлених в ІС.

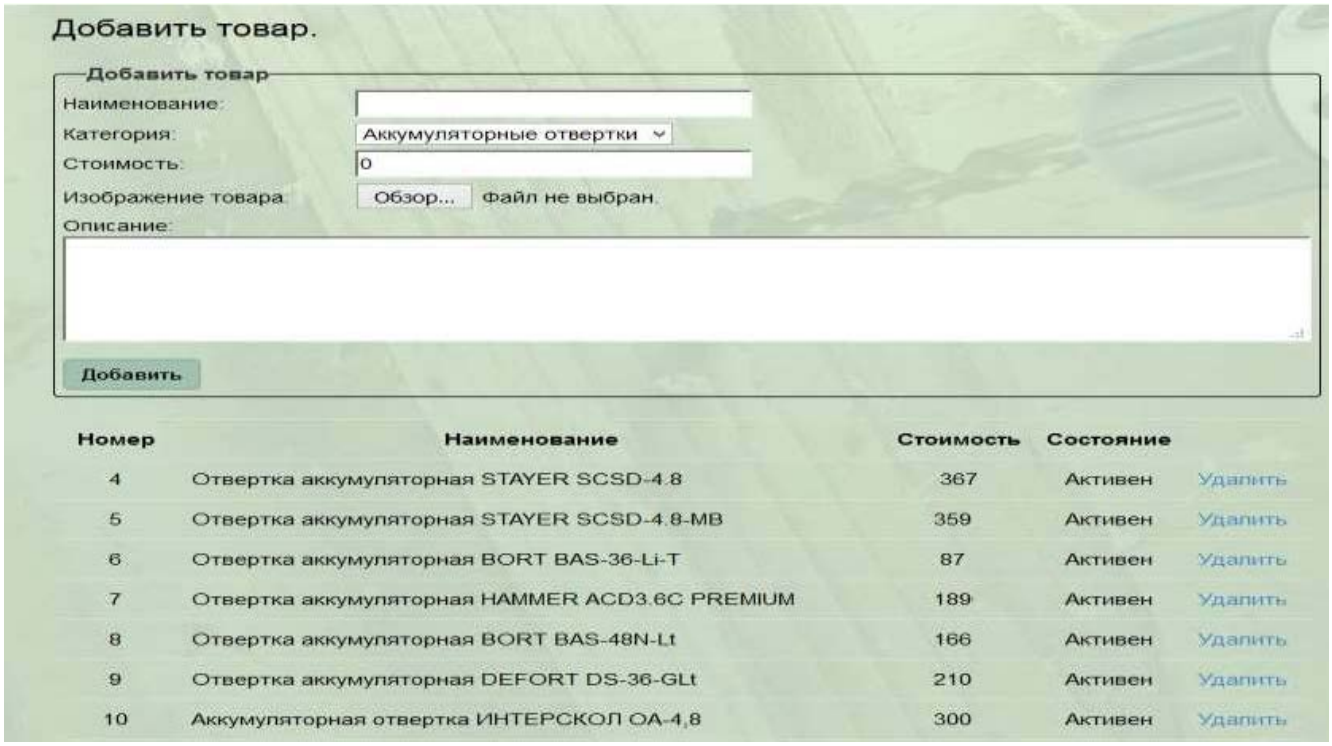

Рисунок 4.7 – Сторінка додавання товару

4.3 Регламент резервного копіювання БД

Для виключення вірогідності втрати даних слід налаштувати регулярне резервне копіювання бази даних, причому краще всього з збереженням резервних копій, наприклад, за останні кілька днів.

Для цього можна використовувати або вбудований в SQL Server планувальник завдань - «SQL Server Agent» або стандартний «Планувальник Windows» в поєднанні з утилітою SQLCMD.EXE, яка дозволяє виконувати запити до SQL Server з командної рядки [13].

Налаштування обслуговування.

Повний щоденне обслуговування (в 00:00 місцевого часу, кожен день):

– перевірка цілісності баз даних примірника MS SQL Server (всі бази даних);

– повне оновлення статистики бази даних (тільки БД ІС);

– очищення журналу примірника MS SQL Server;

– повне резервне копіювання баз даних примірника MS SQL Server (всі бази даних);

– в разі успішного завершення резервного копіювання баз даних примірника - видалення застарілих резервних копій баз даних (період зберігання - 1 тиждень);

– в разі успішного завершення резервного копіювання баз даних примірника - видалення застарілих файлів журналів резервного копіювання (період зберігання - 2 тижні);

– в разі успішного завершення резервного копіювання баз даних примірника - реорганізація індексів таблиць бази даних (тільки БД ІС);

– в разі успішного завершення резервного копіювання баз даних примірника і успішного завершення реорганізації індексів БД - стиснення балки транзакцій БД ІС;

– створення файлу-журналу резервного копіювання;

– в разі неуспішного завершення резервного копіювання відправка листа з повідомленням про помилку адміністратору;

– копіювання файлів резервних копій на дублюючі сховище поза сервера (період зберігання - 1 тиждень);

Створення інкрементальних копій (з 6:00 по 23:30 місцевого часу, кожні 15 хв.):

– формування інкрементальних (наростаючим підсумком) резервних копій журналу транзакцій БД ІС;

– в разі успішного завершення резервного копіювання журналу транзакцій БД ІС - видалення застарілих резервних копій журналу транзакцій (період зберігання - 24 години);

– створення файлу-журналу резервного копіювання;

– в разі неуспішного завершення резервного копіювання відправка листа з повідомленням про помилку адміністратору.

Тестове відновлення БД ІС (один раз в проміжку пір не менш 2-3 тижнів):

– відновлення БД ІС з резервних копій на тестовому екземплярі MS SQL Server. Для перевірки працездатності резервних копій;

4.4 Висновки до розділу

В даному розділі було розглянуто опис інтернет-ресурсу аналогів та їх реалізації. Дана ІС дозволяє проводити пошук товару та його аналогів. у випадку, коли необхідно знайти заміну деталі або обладнання можна ввести в пошуку назву та додати наступні критерії: категорія, виробник, тип, матеріал та особливості. За категорією можна визначити яке саме обладнання необхідно. Далі можна звузити підбір за виробником. Також можна провести класифікацію за типом, наприклад ручне обладнання, від акумулятора, від мережі. Важливим для обладнання критерієм є матеріал, через те що одні та ті ж самі деталі можуть бути виготовлені з різного матеріалу, залежно від матеріалу деталі можна використовувати в різних умовах експлуатації. До критерію особливості були віднесені відмінності притаманні різним крупам товарів.

Також в розділі було розглянуто такі операції системи реалізації інтернет-ресурсу аналогів як: реєстрація в системі, вхід, пошук товару, оформлення замовлення. особистий кабінет, кошик, сторінка адміністрування.

ВИСНОВОК

За підсумками розробки системи для пошуку аналогів обладнання, можна відзначити, що ми маємо повністю працездатну систему, яка може виконувати завдання, передбачені в технічному завданні.

В ході розробки системи була вивчена предметна область, проаналізовані вимоги системи. Велику частину часу зайняла розробка бази даних системи і доступу до неї. Чи не менше часу було витрачено на розробку серверної і клієнтської частини системи. При розробці клієнтської частини системи, був вивчений великий обсяг документації, а так само освоєні основні навички верстки інтернет-сторінок. В даній системі провести пошук аналогів обладнання, яке вийшло з ладу. Пошук можна проводити за визначеними критеріями: категорія, тип, матеріал, виробник та особливості.

В результаті дипломної роботи було виконано:

– проведено аналіз інформаційних систем;

– проведено аналіз існуючих підходів до проєктування інформаційних систем;

– проведено огляд технологій існуючих аналогів та обрані необхідні інструменти;

– розроблено структуру системи;

– розроблено базу даних;

– розроблено інтернет-ресурс аналогів обладнання та їх реалізації.

Використання і впровадження розробки дозволить вигідно і витончено виділитися власнику ІС з числа своїх конкурентів, через те що дану систему можна використовувати не тільки для продажу товару, а і для пошуку аналогів необхідного товару. Реалізовані функціональні ідеї служать для залучення великої кількості користувачів. Таким чином, система є повністю конкурентоспроможною в порівнянні з наявними аналогами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Консорциум W3C или современные стандарты всемирной паутины. [Електроний ресурс]. – 2020 – Режим доступу: http://www.designonstop.com/webdesign/article/konsorcium-w3c-ilisovremennye-standarty-vsemirnoj-pautiny.htm/

2. Беэр Бибо, Иегуда Кац, jQuery. Подробное руководство по продвинутому JavaScript – 2011

3. Закас Н., Ajax для профессионалов, ISBN: 9785932860816 Об издании: [Пер. с англ. А. Киселева], Символ-Плюс – 2015г.

4. Роб Камерон, Дэйл Михалк, ASP.NET 3.5, компоненты AJAX и серверные элементы управления для профессионалов – 2017 – 560 С. ISBN 978-5-8459-1609-9, 978-1-43-021007-8,

5. Адам Фримен, ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов5-е издание – 2015 – 736 С. ISBN 978-5-8459-2008-9, 978-1-430-26529-0;

6. Мэтью Мак-Дональд, Марио Шпушта, Microsoft ASP.NET 4.0 с примерами на C# 2010 для профессионалов,4е издание – 2015 – 1424 С. ISBN 978-5-8459-1702-7, 978-1-43-022529-4;

7. Джон Скит, C# для профессионалов: тонкости программирования – 2014 – 608 С. ISBN 978-5-8459-1909-0, 978-1-617-29134-0;

8. Ицик Бен-Ган, Microsoft SQL Server 2012. Основы T-SQL, – 2014

9. Стивен Хольцнер "Ajax Библия программиста" Диалектика. Москва. СанктПетербург. Киев. – 2009.

10. Э Каслдайн, К. Шарки " Изучаем jQuery " – Питер, – 2012 – 400 С.

11. Дж. Мзерриотт, Joomla! 3.0. Официальное руководство, – Питер , – 2015 – 400 С.

12. Виктор Ромашев, CMS Drupal: Система управления содержимым сайта, – Питер, – 2010.

13. Басыров Р. И., Открываем интернет-магазин с помощью 1С-Битрикс,Эксмо, – 2009 – 511 С.Александр Бондарь, Microsoft SQL Server 2014, – Питер – 2015 – 592С.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ ІНТЕРНЕТ-РЕСУРСУ АНАЛОГІВ ТА ЇХ РЕАЛІЗАЦІЇ

Лістинг А.1 – Розмежування сторінок

```
// Shared/_ProductSummary.cshtml
@model SportsStore. Domain. Entities. Product
<div class= «item»>
@if (Model. ImageData!= null) {
<div style= «float:left; margin-right:20px»>
<img width= «75» height= «75» src="@Url. Action («GetImage», 
«Product»,
new {Model. ProductID})» />
</div>
}
<h3>@Model. Name</h3>
@Model. Description

<div class= «item»>

@using (Html. BeginForm («AddToCart», «Cart»)) {
@Html. HiddenFor (x => x. ProductID)
@Html. Hidden («returnUrl», Request. Url. PathAndQuery)
<input type= «submit» value= "+ Add to cart» />
}

</div>

<h4>@Model. Price. ToString («c»)</h4>
</div>
 // Shared/_error.cshtml
@{
Layout = null;
}

<! DOCTYPE html>
<html>
<head>
<title>Error</title>
</head>
<body>
<h2>
Sorry, an error occurred while processing your request.
</h2>
</body>
</html>
```

```
 // Shared/__ViewStart.cshtml
@{
Layout = «~/Views/Shared/_Layout.cshtml»;
}

 // Shared/__Layout.cshtml
<! DOCTYPE html>
<html>
<head>
<title>@ViewBag. Title</title>
<link href="@Url. Content («~/Content/Site.css»)» rel=
«stylesheet» type= «text/css» />
<script src="@Url. Content («~/Scripts/jquery-1.4.4.min.js»)»
type= «text/javascript»></script>
</head>
<body>
<div id= «header»>
@{Html. RenderAction («Summary», «Cart»);}
<div class= «title»>This is your store</div>
</div>
<div id= «categories»>
@{Html. RenderAction («Menu», «Nav»);}
</div>
<div id= «content»>
@RenderBody()
</div>
</body>
</html>

 //Shared/_AdminLayout.cshtml
<! DOCTYPE html>

<html>
<head>
<title>@ViewBag. Title</title>
<link href="@Url. Content («~/Content/Admin.css»)» rel=
«stylesheet» type= «text/css» />
<script src="@Url. Content («~/Scripts/jquery-1.4.4.min.js»)»
type= «text/javascript»></script>
<script src="@Url. Content
(«~/Scripts/jquery.validate.min.js»)»
type= «text/javascript»></script>
<script src="@Url. Content
(«~/Scripts/jquery.validate.unobtrusive.min.js»)»
type= «text/javascript»></script>

</head>
<body>
<div>
@if (TempData [«message»]!= null) {
<div class= «Message»>@TempData [«message»]</div>
}
```

```
      @RenderBody()
      </div>
      </body>
      </html>

       //Product/List.cshtml
@model SportsStore. WebUI. Models. ProductsListViewModel

      @{
      ViewBag. Title = «Products»;
      }

      @foreach (var p in Model. Products) {
      Html. RenderPartial («ProductSummary», p);
      }

      <div class= «pager»>
      @Html. PageLinks (Model. PagingInfo, x => Url. Action («List»,
      new {page = x, category = Model. CurrentCategory}))
      </div>
       //Nav/Menu.cshtml
      @model IEnumerable<string>

      @{
      Layout = null;
      }

      @Html. ActionLink («All product», «List», «Product»)

      @foreach (var link in Model) {
      @Html. RouteLink (link,
      new {
      controller = «Product»,
      action = «List»,
      category = link,
      page = 1
      },
      new {
      @class = link == ViewBag. SelectedCategory? «selected»: null
      }
      )
      }

       //Cart/Summary.cshtml
      @model SportsStore. Domain. Entities. Cart

      @{
      Layout = null;
      }
      <div id= «cart»>
      <span class= «caption»>
      <b>Your cart:</b>
      @Model. Lines. Sum (x => x. Quantity) item(s),
```

```
@Model. ComputeTotalValue().ToString («c»)
</span>

@Html. ActionLink («Checkout», «Index», «Cart»,
new {returnUrl = Request. Url. PathAndQuery}, null)

</div>

 //Cart/Index.cshtml
@model SportsStore. WebUI. Models. CartIndexViewModel

@{
ViewBag. Title = «Sports Store: Your Cart»;
}

<h2>Your cart</h2>
<table width= «90%» align= «center»>
<thead><tr>
<th align= «center»>Quantity</th>
<th align= «left»>Item</th>
<th align= «right»>Price</th>
<th align= «right»>Subtotal</th>
</tr></thead>
<tbody>
@foreach (var line in Model. Cart. Lines) {
<tr>
<td align= «center»>@line. Quantity</td>
<td align= «left»>@line. Product. Name</td>
<td align= «right»>@line. Product. Price. ToString («c»)</td>
<td align= «right»>@((line. Quantity * line. Product.
Price).ToString («c»))</td>
<td>

@using (Html. BeginForm («RemoveFromCart», «Cart»)) {
@Html. Hidden («ProductId», line. Product. ProductID)
@Html. HiddenFor (x => x. ReturnUrl)
<input class= «actionButtons» type= «submit» value= «Remove» />
}

</td>
</tr>
}
</tbody>
<tfoot><tr>
<td colspan= «3» align= «right»>Total:</td>
<td align= «right»>
@Model. Cart. ComputeTotalValue().ToString («c»)
</td>
</tr></tfoot>
</table>
<p align= «center» class= «actionButtons»>
<a href= "@Model. ReturnUrl»>Continue shopping</a>
@Html. ActionLink («Checkout now», «Checkout»)
```

```
</p>

 //Cart/Completed.cshtml
@{
ViewBag. Title = «SportsStore: Order Submitted»;
}

<h2>Thanks!</h2>
Thanks for placing your order. We'll ship your goods as soon as possible.



 //Cart/Checkout.cshtml
@model SportsStore. Domain. Entities. ShippingDetails

@{
ViewBag. Title = «SportStore: Checkout»;
}

<h2>Check out now</h2>
Please enter your details, and we'll ship your goods right away!
@using (Html. BeginForm()) {

@Html. ValidationSummary()

<h3>Ship to</h3>
<div>Name: @Html. EditorFor (x => x. Name)</div>

<h3>Options</h3>
<label>
@Html. EditorFor (x => x. GiftWrap)
Gift wrap these items
</label>
<h3>Address</h3>
<div>@Html. EditorFor (x => x. Line1) Phone number</div>
<div>@Html. EditorFor (x => x. Line2) Home number</div>
<div>@Html. EditorFor (x => x. Line3) Street</div>
<div>@Html. EditorFor (x => x. City) City</div>
<div>@Html. EditorFor (x => x. State) State</div>
<div>@Html. EditorFor (x => x. Zip) Zip</div>
<div>@Html. EditorFor (x => x. Country) Country</div>

<p align= «center»>
<input type= «submit» class= «actionButtons» value= «Complete order» />
</p>
}

 //Admin/Index.cshtml
@model IEnumerable<SportsStore. Domain. Entities. Product>
```

```
@{
ViewBag. Title = «Admin: All Products»;
Layout = «~/Views/Shared/_AdminLayout.cshtml»;
}

<h1>All Products</h1>
<table class= «Grid»>
<tr>
<th>ID</th>
<th>Name</th>
<th class= «NumericCol»>Price</th>
<th>Actions</th>
</tr>
@foreach (var item in Model) {
<tr>
<td>@item. ProductID</td>
<td>@Html. ActionLink (item. Name, «Edit», new {item.
ProductID})</td>
<td class= «NumericCol»>@item. Price. ToString («c»)</td>
<td>
@using (Html. BeginForm («Delete», «Admin»)) {
@Html. Hidden («ProductID», item. ProductID)
<input type= «submit» value= «Delete»/>
}
</td>
</tr>
}
</table>
<p>@Html. ActionLink («Add a new product», «Create»)</p>

 //Admin/Edit.cshtml
@model SportsStore. Domain. Entities. Product

@{
ViewBag. Title = «Admin: Edit» + @Model. Name;
Layout = «~/Views/Shared/_AdminLayout.cshtml»;
}

<h1>Edit @Model. Name</h1>

@using (Html. BeginForm («Edit», «Admin», FormMethod. Post,
new {enctype = «multipart/form-data»})) {

@Html. EditorForModel()
<div class= «editor-label»>Image</div>
<div class= «editor-field»>
@if (Model. ImageData == null) {
@:None
} else {
<img width= «150» height= «150»
src="@Url. Action («GetImage», «Product», new {Model.
ProductID})» />
}
```

```
     <div>Upload new image: <input type= «file» name= «Image»
/></div>
     </div>

     <input type= «submit» value= «Save» />
     @Html. ActionLink («Cancel and return to List», «Index»)
     }

      //Account/LogOn.cshtml
     @model SportsStore. WebUI. Models. LogOnViewModel

     @{
     ViewBag. Title = «Admin: Log In»;
     Layout = «~/Views/Shared/_AdminLayout.cshtml»;
     }

     <h1>Log In</h1>

     <p>Please log in to access the administrative area:</p>
     @using (Html. BeginForm()) {
     @Html. ValidationSummary(true)
     @Html. EditorForModel()
     <p><input type= «submit» value= «Log in» /></p>
```

Етапи створення інформаційної системи

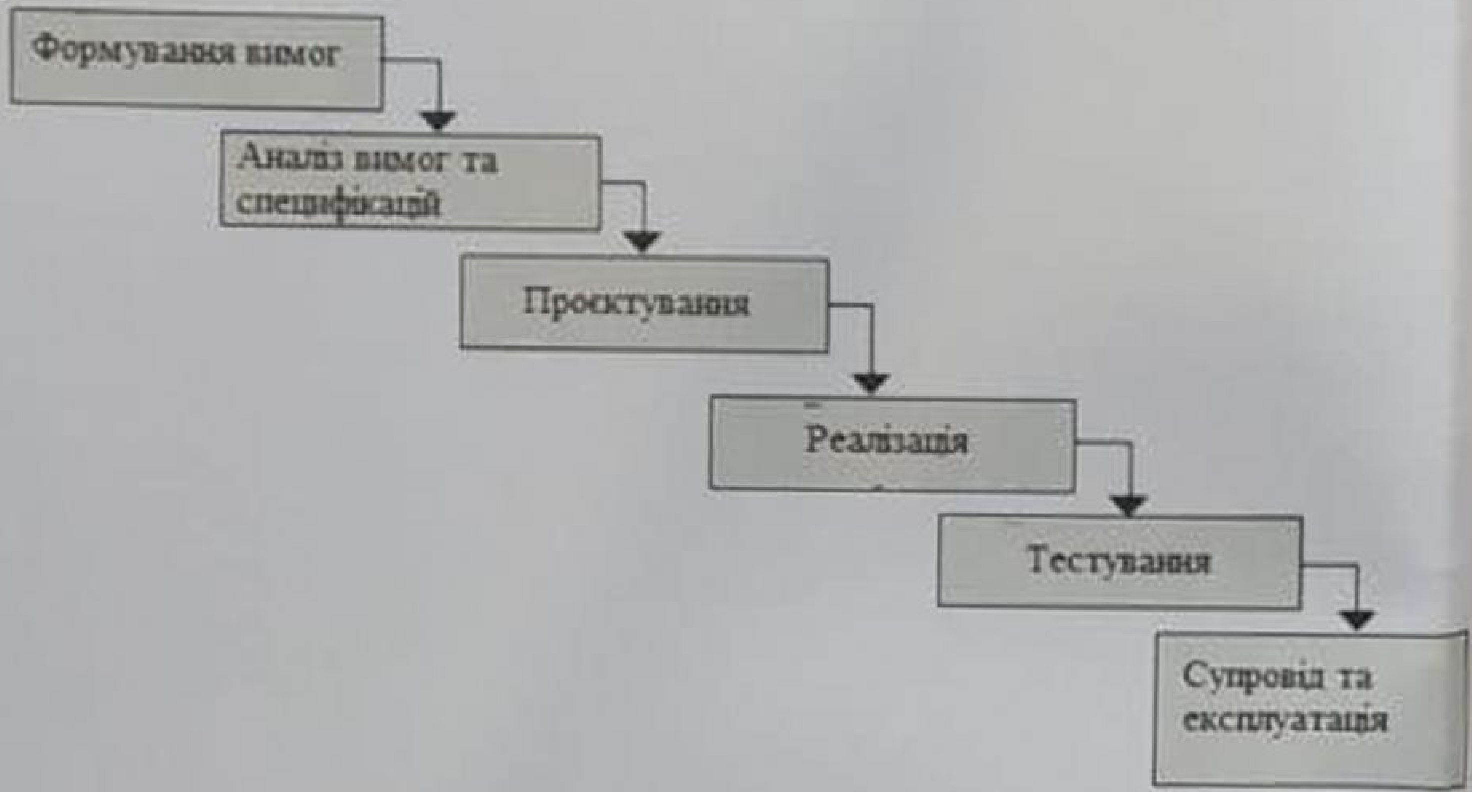

Рисунок 1 – Етапи створення ІС

Рисунок 2 – Класична модель Web-додатка

Зм.	Лист	№ докум.	Підп.	Дата
Розроб.				
Перев.				
Т. Контр.				
Н. контр.				
Затв.				

13.02070849.00068 ПЛ1

Етапи створення інформаційної системи

Рисунок 3 – Структура системи

Рисунок 4 – Діаграма БД

					13.02070849.00068 ПЛ2	
Зм.	Лист.	№докум.	Підп.	Дата	Розробка інформаційної комп'ютерної системи пошуку аналогів промислового обладнання Опис системи	
Розроб.						
Перев.						
Т. контр.						
Н. контр.						
Затв.						

Проєктування бази даних

Таблиця 1 – Структура dbo.User

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
RegDate	datetime	Дата реєстрації
Email	varchar (50)	Адреса електронної пошти
FirstName	nvarchar (50)	Ім'я
LastName	nvarchar (50)	Прізвище
Phone	char (13)	Телефон
Password	nchar (36)	Пароль
RoleId	bigint	Роль користувача
Cookies	nchar (36)	Кукі користувача

Таблиця 2 – Структура dbo.Basket

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
UserId	bigint	Ідентифікатор користувача
ProductId	bigint	Ідентифікатор товару
Count	Int	Кількість одиниць в кошику обраного товару

Таблиця 3 – Структура dbo.Product

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
RegDate	datetime	Дата реєстрації
Title	varchar (200)	Відображуване найменування
Description	text	Опис
CategoryId	bigint	Ідентифікатор категорії товару
Price	float	Ціна
IsActive	int	Статус товару

Таблиця 4 – Структура dbo.Order

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
UserId	bigint	Ідентифікатор користувача
AddDate	datetime	Дата замовлення
Comments	text	Коментар до замовлення
Adress	text	Адреса доставки
StateId	bigint	Ідентифікатор стану замовлення
PaymentDate	datetime	Дата проведення оплати
PaymentSumm	float	Сума оплати

Таблиця 5 – Структура dbo.OrderItem

Поле	Тип	Опис
Id	bigint	Ідентифікатор запису в таблиці
OrderId	bigint	Ідентифікатор користувача
ProductId	bigint	Ідентифікатор обраного товару
Count	int	Кількість що залишилося днів до виділення замовлення
Cost	float	

Опис процесу експлуатації

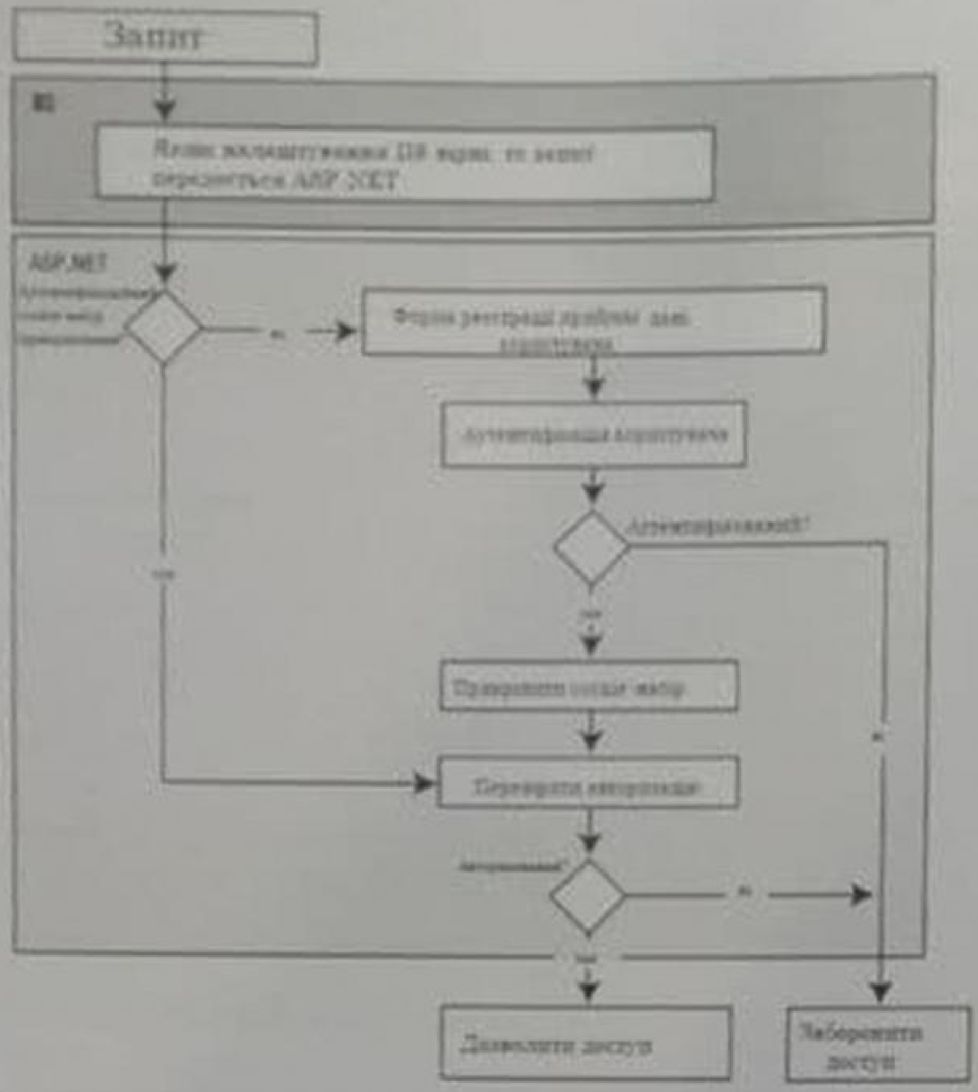

Рисунок 5 – Процес аутентифікації

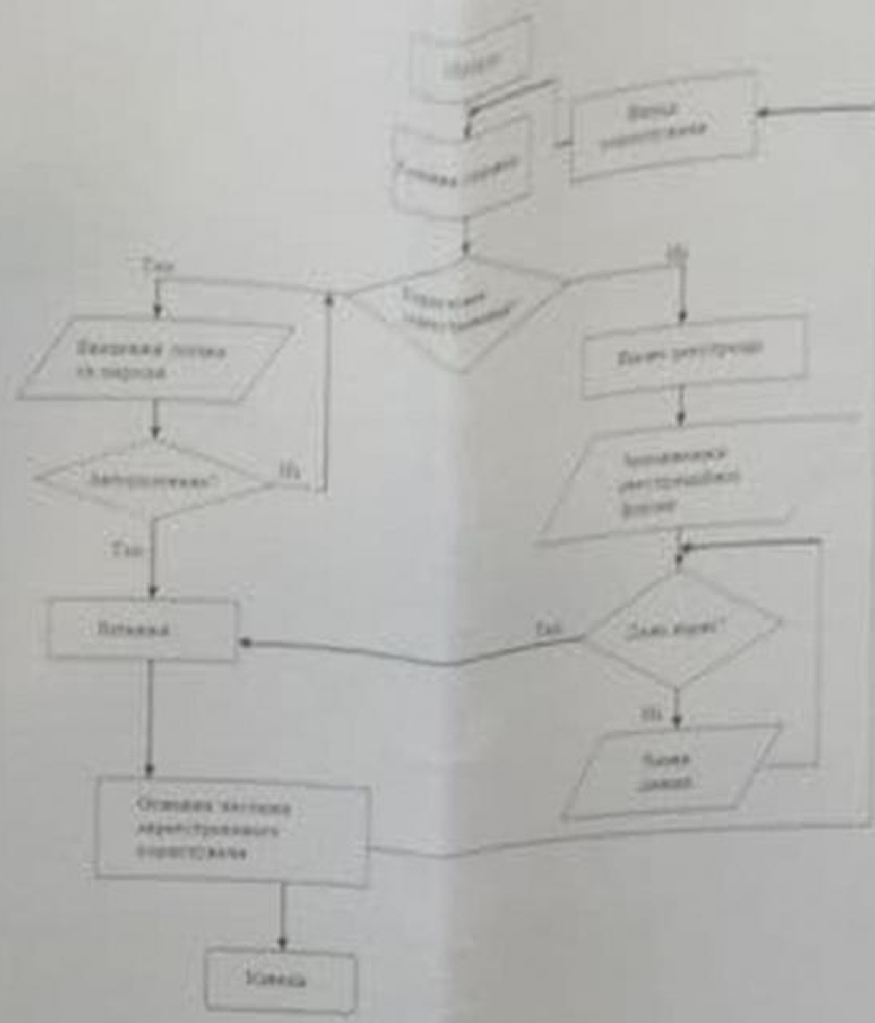

Рисунок 6 – Блок-схема алгоритму реєстрації користувача

Рисунок 7 – Блок-схема алгоритму оформлення замовлення

Зм.	Лист.	№докум.	Підп.	Дата
Розроб.				
Перев.				
Т. Контр.				
Н. контр.				
Затв.				

13.02070849.00068 ПЛ4

Опис процесу експлуатації

Опис інтерфейсу та основних функцій інформаційної системи

Рисунок 8 – Сторінка «Реєстрація»

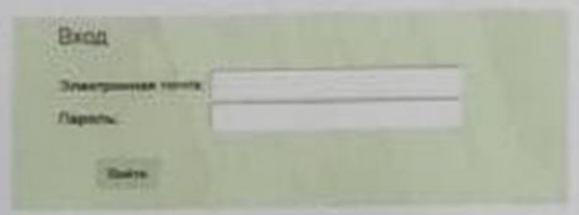

Рисунок 9 – Сторінка «Вхід в систему»

Рисунок 10 – Каталог товарів

Рисунок 11 – Сторінка детального перегляду товару

Рисунок 12 – Сторінка оформлення замовлення

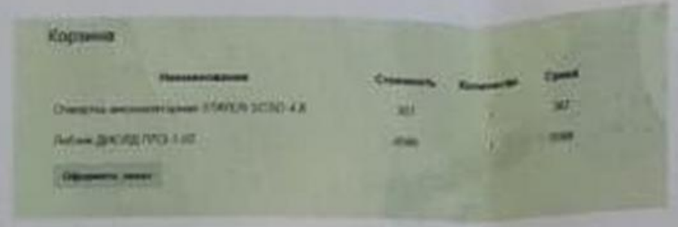

Рисунок 13– Кошик користувача

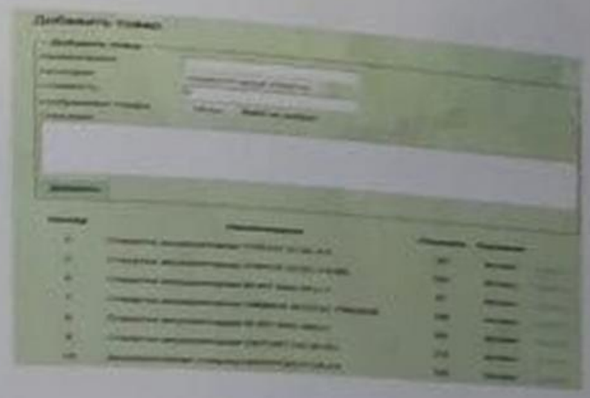

Рисунок 14 – Сторінка додавання товару

Зм.	Лист.	№докум.	Підп.	Дата	13.02070849.00068 ПЛ5		
Розроб.					Розробка інформаційної комп'ютерної системи пошуку аналогів промислового обладнання. Опис інтерфейсу та основних функцій інформаційної системи		
Перев.							
Т. контр.							
Н. контр.							
Затв.							