Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка серверної програми для аналітики продажів

Виконав(ла): студент(ка) IV курсу, групи СПз-41
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Григорів В.В. (прізвище та ініціали)

Керівник (підпис) Пастух О.А. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Програмна інженерія
(шифр і назва спеціальності)

студенту Григоріву Віталію Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка серверної програми для аналітики продажів

Керівник роботи Пастух Олег Анатолійович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_15__» *квітня* 2024 року № *4/7-340*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	Гурик О.Я., доцент кафедри МТ		
основи охорони праці	Лазарюк В.В., доцент кафедри МТ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	**Примітка**
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу 1		
4	Виконання розділу 2		
5	Виконання розділу 3		
7	Оформлення пояснювальної записки		
8	Оформлення графічного та презентаційного матеріалу		
9	Попередній захист		
10	Захист		

Студент _____ Григорів В.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пастух О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота має: с. ___, рисунків ___, таблиць ___, літ. ___

предмента область, мета, завдання, об’єкт, розробка, інформаційна система, серверна програма, вимоги, середовище програмування, база даних

В роботі розроблено серверну програму для аналітики продажів. Програма орієнтована на підвищенняя ефективності бізнесових процесів торгівельної компанії.

ABSTRACT

The qualification work has: p. ___, drawings ___, tables ___, lit. ___

subject area, goal, task, object, development, information system, server program, requirements, programming environment, database

The work developed a server program for sales analytics. The program is focused on increasing the efficiency of the business processes of the trading company.

ЗМІСТ

ВСТУП

У сучасному світі люди повсюдно використовують інтернет якості універсального та функціонального інструменту в багатьох сферах життєдіяльності людини. Розширюючись та розвиваючись, глобальна мережа набула великої значущості для комерційної діяльності. Інтернет привернув широкий інтерес з боку бізнесу завдяки активному формуванню електронної комерції, появі віртуальних торгових майданчиків, широким можливостям реклами та підвищення рівня економічної взаємодії в інформаційному полі.

Успішність комерційних організацій дедалі більше має залежність від ефективності та повноти використання інформаційних та інтернет технологій найбільш актуальним є інформаційні системи для накопичення, збереження та опрацювання інформації. Така потреба підтверджується у високому попиті на ринку відповідного програмного забезпечення. Однак через постійно зростаючі і мінливі вимоги інформатизації та автоматизації, існуючі пропозиції в ряді випадків що неспроможні задовольнити всіх потреб.

З прикладів можна виділити такі проблеми, з якими стикаються торгові організації при виборі ПЗ:

- висока ціна гіганти ІТ індустрії часто пропонують великі та програмні комплекси, що задовольняють поточним потребам, проте їх придбання та обслуговування, можуть дозволити організації тільки з високим фінансуванням інформатизації;
- брак спеціалістів на ринку праці – проблема виникає з використанням ІС створеної на базі специфічного чи неактуального програмного та/або апаратного забезпечення. Вартість послуг таких фахівців, відповідно, може нести значні витрати;
- низька якість безпеки - при критичності та необхідності високого рівня безпеки вибір на ринку програмного забезпечення

- стає ще меншим. Невеликі компанії-розробники часто не мають змоги забезпечити безпеку своєї системи на найвищому рівні, а IT-гіганти з популярними та величезними програмними комплексами постійно стикаються зі зломом їхніх систем і в міру знаходження слабких місць випускають оновлення безпеки, які нерідко надходять постфактум - після злому. ;
- складність в обслуговуванні - може виражатися у необхідності навчання співробітників для роботи в системі, додаткових і не рідко, високих витратах на підтримку роботи, необхідністю в наймані дорогих фахівців;
- відсутність інтеграції та сучасного функціоналу - з такою проблемою може зіткнутися як мала організація-розробник через брак ресурсів, так і великі організації через труднощі впровадження нових технологій у великі програмні комплекси.

Інтернет-технології, що постійно розвиваються, створюють нові потреби на ринку програмних продуктів і є головним фактором актуальності розробки нових інформаційних систем для торгових організацій.

Отже, метою роботи є розроблення серверної програми для аналітики продаж. Результатом є підвищенняя ефективності бізнесових процесів торгівельної компанії.

1. АНАЛІЗ ВІДОМИХ РІШЕНЬ

1.1 Особливості інформатизації торгівельної компанії

У системі соціальних та економічних зв'язків відбуваються помітні зміни, спричинені зростанням використання інтернет-технологій. Нові методи та засоби для взаємодії між людьми та організаціями формують сучасне середовище ведення бізнес-діяльності, в якому торговим організаціям необхідно взаємодіяти з інформаційними ресурсами, організовувати обмін інформацією в реальному часі, миттєво та незалежно від відстані. Виходячи з цього є необхідність у впровадженні нових методів управління процесами та інформацією серед електронної комерції, які відповідатимуть сучасним вимогам ринку [15].

Велика частка прикладних процесів торгової організації складає робота з реєстрами даних [3], таких як: клієнти, постачальники, замовлення, товари, а також координація взаємодії з контрагентами, розподіл завдань для співробітників, аналіз та аналітика цін, зростання та падіння продажів, аналіз дій із залучення нових клієнтів.

У процесі інформування клієнтів важливо мати реєстр актуальних даних про товари, інформацію про клієнта, історію його покупок, можливість додати нового клієнта до бази даних, швидко сформувати нове замовлення або обробити вже існуюче, мати електронний засіб комунікації [12]. У процесі оформлення покупки є необхідність формування таких документів як накладна, торговий чек.

Процеси керівництва організації вимагають можливість аналітики та аналізу надходження прибутку (замовлень). Використовуючи бази даних керівник відділу продажів може проводити аналітику успішності торгівлі за певний період, прогноз продажів на майбутнє та аналіз ефективності вжитих раніше дій, наприклад, показник успішність нової рекламної кампанії. Для цього в певний момент часу керівник ВП збирає дані на замовлення та контрагенти, аналізує дані та формує висновки у звіт діяльності організації, щоб потім надати його своєму керівнику — директору. Також керівник відділу продажів відповідає за роботу підлеглих — менеджерів/продавців, аналізуючи кількість успішних і не успішних замовлень та

інших значущих для бізнесу дій, виконаних працівниками в рамках робочих обов'язків. Директор у свою чергу на основі звітів проводить аналітику бізнес-процесів, наприклад, реалізація тих чи інших товарів і відповідно необхідність знайти більше постачальників для цієї продукції або необхідність збільшити цю продукцію на складі [12].

Поява такої системи в організації може значно покращити показники ведення бізнес-процесів. Деякі організації приймають рішення щодо розробки власної інформаційної системи «під ключ», замість придбання існуючих спеціалізованих програмних комплексів або використовувати альтернативні варіанти. Так, як альтернативна база даних у малих організаціях виступають звичні електронні таблиці з офісних пакетів. Проте, такий підхід викликає проблеми зі складанням звітів, пошуком інформації (розподіленої у кількох незв'язаних таблицях і навіть комп'ютерах), втрату даних. Така робота не може бути ефективною і відображати реальні дані, тому що в процесі збору інформація може дублюватися, губитися, ставати неактуальною, а вся робота займе значну кількість часу. Розробка власної інформаційної системи може стати вірним рішенням, якщо при цьому проектування та розробка виконуватимуться з використанням сучасних стандартів та методологій. Така вимога необхідна не тільки для розробки ІВ як кінцевий продукт, що відповідає поточним вимогам, а й продукту, який зможе розвиватися та розширюватися відповідно до зростання організації та ринку. Це значить, що в майбутньому над цією системою будуть працювати й інші фахівці, ефективність та взаємодія яких багато в чому залежатиме від розробленої основи ІС. Інший важливий чинник при створенні інформаційної системи – це вибір програмного забезпечення, що забезпечує її роботу. Для малих і середніх організацій використання платних компонентів може перекреслити всі переваги власної інформаційної системи, а вибираючи програмні продукти, що вільно розповсюджуються, необхідно враховувати їх надійність, безпеку, подальший розвиток, наявність аналогів і можливість їх використання при розширенні інформаційної системи.

Робота торгівельної компранії охоплює наступне: виробник, оптовик,

продавець, покупець. Кожна з цих складових є важливою для підтримк її функціонувння.

Якщо зупинитись на роздрібній торгівлі, то для неї потрібно, щоб виробник надав товар, оптовики закупили його у великій кількості та продали продавцю, який надасть можливість покупцю купити його.

Коло постачання розподіляє продукцію від виробника до споживача. Спочатку може здатися, що це спричинить інфляцію, але це лише знизить ціну.

Тепер стосовно точок продажу. Розділяють наступні: супермаркет, інтернет-магазин, спеціалізований магазин.

Поширеними є: універмаг, де є відповідні відділи із великим асортиментом товару; інтернет магазин – популяність здобув завдяки кокуренції цін та варіантів доставки; звичайний магазин (крамниця) – має товарипершої необхідності. І залишається гігант у торгівлі – супермаркет.

Аналізуючи цю інформацію, можна зробити висновок, що потреба організації до інформаційної системі, то, можливо задоволена створенням власної ІС.

Рекомендоване та загальновизнане співтовариством розробників інформаційних систем рішення, продиктоване здоровим глуздом – це присутність в організації централізованої системи управління базою даних, в якій зберігатиметься, оброблятиметься та вноситиметься інформація необхідна для роботи організації. Крім цього організація одержує хороше програмне середовище для подальшої роботи з інформацією іншими засобами, програмними комплексами. Основна особливість СУБД - наявність процедур для введення і зберігання як самих даних, а й описів їх структури. Централізоване зберігання забезпечує такі переваги, як цілісність та несуперечність інформації, виключення дублювання, забезпечення безпеки та безпеки даних. Також можливе створення рольової моделі роботи з даними, що знаходяться в централізованій системі управління базою даних.

Необхідність використання актуальної інформації з інформаційної бази може з'явитися і поза офісним середовищем, наприклад, у службовому відрядженні або

на діловій зустрічі для демонстрації зростання продажу чи каталогу продукції. У рамках такої діяльності зручно мати можливість оперативно отримати доступ до бази даних та засобів формування звітів, графіків, при цьому маючи стандартне апаратно-програмне забезпечення.

Сучасні можливості інтернет технологій та централізоване зберігання інформації дає і такі інструменти, як інтеграція зі сторонніми інформаційними системами [4], наприклад, у рекламі. У рекламній діяльності торгова організація може автоматизувати відправлення даних про товари та ціни до спеціалізованих рекламних організацій, що надають послуги торгових інтернет майданчиків, а також у власні торгові інформаційні системи, такі як інтернет-магазин, електронні розсилки клієнтам.

Практичні реалізації програмного забезпечення у всьому світі багато разів стикалися з проблемами сумісності, найчастіше це виражалося у можливості використання програмного продукту їх користувачами, а також підтримкою ним іншого ПЗ розробленого сторонніми розробниками. Рішення посталих проблем сумісності призвело до розроблення великої кількості міжнародних нормативних документів у галузі використання та розроблення інформаційних систем. До основних понять віднесено поняття «відкриті системи» [8].

Їх визначають як систему, яка застосовує задекларовану номативну документацію призначену для ІТ та профілі функційних стандартів для специфікації інтерфейсів, які специфікують інтерфейси, відповідні служби, підтримку форматів з метою забезпечення взаємодії і мобільності веб-застосунків, данних і персоналу [8,10]. Таке формулювання прописане фахівцями IEEE.

До властивостей відкритих ІС віднесено [14]:

- мобільність / переносимість — корисна при необхідності перенесення програм і даних на випадок заміни чи модернізації апаратної частини ІС, а також забезпечує можливість роботи фахівці не потребуючи їх перепідготовки;

- розширюваність/масштабованість — дана властивість повинна спростити та забезпечити можливість додати новий функціонал до ІС чи змінювати наявні, не торкаючись інших функціональних частин інформаційної системи;
- стандартизованість - проектування та розроблення ІС має відбуватися з врахуванням діючих стандартів;
- взаємодія та їх інтеграція є важливою властивістю, у цьому сприяє наявність технічних засобів, що забезпечують реалізацію об'єднання різних ІВ за допомогою мережевих технологій;
- співпраця з користувачем – зрозумілі, звичні та універсальні людино-машинні інтерфейси, які дозволяють користувачеві використовувати інформаційну систему без спеціальної підготовки.

Розробка та проектування інформаційної системи для організації торгівлі з урахуванням перерахованих властивостей дозволить реалізувати систему «дружню» до кінцевого користувача, виключити такі недоліки як: асинхронність наповнення бази даних, дублювання інформації, втрата актуальності, дасть можливість інтеграції та стандартизувати як компоненти, так і систему загалом.

Слід враховувати й те, що можливість створити інформаційну систему відкритої системою можна лише розглядаючи перелічені характеристики як набір заходів, вжитих під час проектування, розробці та впровадженні інформаційної системи. Таким чином, властивості будуть не окремими, а доповнюють і розширюють один одного, організуючи відкриту систему. Дотримуючись принципів відкритих систем буде досягнуто поставлених цілей та завдань, вирішено проблеми, з якими стикаються як торгові, так і інші організації в процесі інформатизації та автоматизації своєї діяльності.

Аналізуючи приклади зберігання, наповнення та обробки даних в інформаційних системах торгових організацій, що не мають властивостей відкритих систем або виключають деякі з них, можна виділити такі проблеми:

• відсутність єдиного реєстру (бази даних) — при децентралізованому зберіганні та інтерфейсі доступу до даних виникає низка проблем: дублювання інформації, втрата актуальності, труднощі зі звітністю.

• Неможливість організації віддаленого доступу — викликає складність у своєчасному отриманні даних, необхідність додаткових процесів передачі інформації, втрата стратегічного переваги, пов'язаного з часом отримання даних.

• Необхідність спеціального клієнтського ПЗ — обмежує кінцевого користувача у виборі апаратног-програмного забезпечення для отримання доступу до даних.

• Неможливість або обмеження інтеграції зі стороннім ПЗ – обмежує організацію у функціональних можливостях інформаційної системи або пов'язує розширення та модернізацію ІС з певним розробником/середовищем розробки.

Аналіз літератури та практичного досвіду розробників інтерфейсів для інформаційних систем призводить до актуального та популярного рішення - створення web-додатку, яке дозволить поєднати комплекс властивостей відкритих систем.

Під визначенням «Веб-додаток» розуміється прикладне програмне забезпечення в реалізації клієнт-серверне рішення, обмін інформацією в якому відбувається за допомогою мережі (локальної або глобальної) через HTTP і/або HTTPS прикладні протоколи передачі гіпертексту [6]. У цьому рішенні клієнтська частина може бути незалежною від апаратного та програмного забезпечення, виконуючи функцію інтерфейсу користувача. Також слід згадати, що реалізація клієнтського інтерфейсу у веб-додатку може не вимагати великих ресурсів до апаратного забезпечення, виконуючи обмежену кількість обчислень, необхідних для відображення отриманої інформації з сервера та відправлення необхідних запитів назад.

Веб-додатки мають свої позитивні особливості, що дотримуються та доповнюють властивості відкритих систем [4]:

• Клієнтська частина виконується незалежно від операційної системи чи апаратного забезпечення клієнта. Як програмне забезпечення використовуються інтернет-браузери. Однак при розробці та тестуванні клієнтського інтерфейсу слід врахувати, що додаток повинен мати властивості кросбраузерності через різну реалізацію браузерами стандартів HTML, CSS і DOM. Однак на практиці більшість

стандартів ними дотримано, а відмінності випливають через різний час застосування нових стандартів розробниками цих програм оновлень.

• Веб-додатки є розподіленою інформаційною системою із серверної та клієнтської частин. Серверна частина проектується для забезпечення необхідної кількості клієнтських звернень та має можливість масштабування.

• У процесі розробки веб-додаток може бути відкритим для тестування з віддалених комп'ютерів, що є позитивним фактором для застосування гнучких методик розробки.

• Розробка веб-додатку є гарною перспективою, особливо для торгових організацій, враховуючи сферу електронної комерції, що зростає і вже повсюдно.

• Надаються широкі можливості соціальної взаємодії, взаємодії та інтеграції ІС з іншими інформаційними системами, що функціонують у глобальній мережі.

• Завдяки працездатності на будь-якому апаратному забезпеченні та актуальності таких систем веб-програми можуть становити реальну конкуренцію нативним додаткам операційних систем (Apple iOS, Google Android, Windows Phone). В найближчому майбутньому конкурентоспроможність може сильно зрости завдяки роботі W3C (організація, що розробляє та впроваджує технологічні стандарти для Всесвітньої павутини) над створенням відкритих стандартів, що дають веб-додаткам доступ до апаратної складової пристроїв, що сильно розширить можливості клієнтської складової.

Як видно з перерахованих властивостей та особливостей веб-додатків — вони дуже підходять до визначення відкритих систем і дотримуються їхніх принципів. Найбільш яскраве здійснення властивостей відкритих систем - це внутрішня та зовнішня взаємодія веб-додатку за допомогою мережевих технологій, що є дотриманням якості взаємодії. Мобільність і переносимість відмінно забезпечується на клієнтському рівні - як правило, будь-яка операційна система містить браузер, який є програмною основою для роботи кінцевого користувача з веб-додатком, на серверній стороні переносність забезпечується безліччю сумісних аналогів програмного забезпечення (такого як СУБД), можливістю запуску серверного ПЗ у різних операційних системах та на апаратних рішеннях, легкість

перенесення серверної частини на інше обладнання. Чорта масштабованості добре присутня у веб-додатках — частково це забезпечується стандартизацією розробки та реалізації веб-додатків, з цього також випливає властивість стандартизації, хоча деякі аспекти програмування мають досить молоді і не усталені стандарти, вони дуже швидко поширюються завдяки сфері та області функціонування даних інформаційних систем та можливості швидкого впровадження. Дружність клієнтського інтерфейсу до користувача багато в чому дотримується завдяки наявності технологій та стандартів з розробки веб-додатків для різних пристроїв, часто цей фактор зводиться до найзначнішого: розмір екрану та вид апаратного забезпечення, що використовується, та спосіб керування (класична клавіатура та комп'ютерна миша або сенсорний екран) . Веб-додаток дозволяє зробити інтерфейс зручним для кінцевого користувача незалежно від того, використовує смартфон, персональний комп'ютер або телевізор з функцією SMART-TV. Клієнтська частина веб-програми може бути реалізована у максимально простому та зрозумілому вигляді, для вирішення конкретних завдань користувача «під ключ».

Організація розробки веб-додатку є дуже гнучкою, на кінцевий вибір методології можуть впливати такі фактори, як масштаб інформаційної системи, фінансові та часові рамки розробника, технічне завдання замовника.

Сучасні аспекти та концепції у методиках для розробки веб-додатків:

• Модель горизонтальної організації проектної групи виникла через недоліки традиційної ієрархічної структури як ефективний аналог. Така група - це багатопрофільна, невелика команда, кожен у якій бере на себе відповідальність за ту чи іншу частину розробки, а також доповнює компетентність суміжних членів команди, що забезпечує єдине бачення проекту кожним учасником. У таких командах виставляються високі вимоги до компетенції та кінцевого результату, заохочується бажання освіти удосконаленні навичок. Нерідко такі команди збираються у великих організаціях для розробки конкретної інформаційної системи (наприклад, Microsoft Solution Framework), проте є й антонімічні приклади — екстенсивна модель зростання за рахунок збільшення прибутку, шляхом підвищення набору низькооплачуваних фахівців, що викликає постійну плинність

кадрів, але знижує ризики через кадрові втрати.

• Модель проектної групи включає до свого складу фахівців, які виконують ролі: дизайнера/проектувальника інтерфейсів, розробника, спеціаліста з оптимізації, тестера, керуючого проекту. Усі учасники у такій проектній групі рівноправні, кожен учасник виконує певну роль чи кілька ролей одночасно, а керівник проектом є технічним спеціалістом. Мінімальний колектив повинен складатися не менше ніж із 3 осіб.

• Інтеграція IT спеціалісту функцій веб-розробника є створенням універсального спеціаліста, який повинен мати високу якість навичок у галузі розробки як серверної складової, так і клієнтської частини – це верстка, проектування баз даних, оптимізація та проектування інтерфейсів. Фахівець може бути розвинений більше в одній галузі, ніж в іншій, але має бути здатний реалізувати проект цілком, хай і за більший час, ніж це зробить команда спеціалістів. Сучасні вимоги до розробників веб-додатків передбачають широке бачення розробки проекту поза рамками своєї прямої компетенції, т.к. різні частини веб-застосунку багато в чому доповнюють один одного і щільно взаємодіють.

• Життєвий цикл розробки веб-додатку схожий на методологію Rational Unified Process і складається з 4 фаз, яка може бути виконана лише один раз або повторюватися кілька разів за потреби. Фаза «Початок» формує бачення та межі проекту, визначає основні вимоги, обмеження та ключову функціональність продукту. Також за цієї фази оцінюються ризики. У фазі «Уточнення» проводиться аналіз предметної галузі та побудова архітектури додатка, розробка інтерфейсів та дизайну. У фазі «Конструювання» відбувається реалізація більшої частини функціональності. У фазі «Впровадження» - відбувається тестування та оцінка, визначається якість веб-додатку, що вийшов, проводиться оптимізація.

• Критичний підхід до використання та вибору готових рішень. Саме використання існуючих програмних рішень для підвищення швидкості та/або якості програми може бути хорошим фактором, але при цьому необхідно враховувати вплив таких рішень на безпеку, архітектуру та подальшу масштабованість. При використанні готових рішень має бути розуміння їхньої

логіки роботи та можливість модифікації. Тому для серйозних проектів вимогливих насамперед до безпеки не використовуються повністю готові системи керування веб-вмістом (WordPress, Joomla, OpenCart).

• Замовник завжди поруч – концепція екстремального програмування. Замовник є кінцевим користувачем продукту, який повинен бути завжди на зв'язку для діалогу.

• Парне програмування – це техніка, яка стала дуже продуктивною та корисною при розробці веб-додатків із певними умовами. Парне програмування проводиться двома фахівцями, з яких один здійснює кодування в деталях, а інший зосереджений на картині загалом. Через деякий час відбувається зміна ролей. Сумежний принцип колективного володіння кодом означає, кожен розробник відповідає за весь вихідний код.

Виходячи з перерахованих вище способів розробки веб-додатків зроблено висновок, що методика розробки може бути різною при різних проектах, а також поєднувати в собі кращі якості, організуючи гібридні варіанти в процесі реалізації конкретних завдань. На вибір методики може впливати час виконання, обґрунтування ресурсів і вимогливість до кінцевого функціоналу.

1.2. Стандарти розробки веб-додатків

Крім методики організації розробки веб-додатків, не менш важливими є правила написання коду. Стандартизація написання коду веб-додатків — це рішення, що розвиваються., однією з популярних і глобально застосовуваних методик, є БЕМ-методологія (Блок, Елемент, Модифікатор), яка включає в себе компонентний підхід до веб-розробки, що передбачає поділ коду і файлової структури інтерфейсу на незалежні блоки.

БЕМ-методологія супроводжується великою кількістю технічної документації, які готові до роботи прикладами, інструментаріями, доступними у

відповідному розділі.

БЕМ включає в себе:

• Методологічні рекомендації – це прості поради, які допомагають організувати проект, найбільш швидко його реалізувати та зручно підтримувати тривалий час.

• Технології та бібліотеки з відкритим вихідним кодом – готова реалізація рекомендацій БЕМ.

• Інструменти для автоматизації роботи з методологією БЕМ.

Переваги БЕМ:

• Проста підтримка структури коду під час зростання проекту.

• Повторне використання коду — дозволяє скоротити терміни та вартість нових розробок за рахунок використання однакових компонентів.

• Точкові зміни з мінімальними витратами: оновлення дизайну, додавання функціональних елементів тощо.

• Масштабування – код розвивається за заздалегідь відомими правилами.

• Загальна термінологія – дозволяє спростити взаємодію різних розробників.

• Автоматизація – загальні єдині правила сприяють автоматизації процесів. Частина коду може бути згенерована автоматично.

БЕМ рекомендує багато популярних розробників, консультантів і технічних директорів. Його використовують у таких IT-гігантах як Google, BBC, BuzzFeed. Також БЕМ можна використовувати не тільки у веб-розробці, але й у будь-якій іншій мові програмування та фреймворку, тому що методологія пропонує лише абстрактні практики щодо підвищення надійності та повторного використання коду [23].

Крім стандартизованої розробки веб-програми в частині інтерфейсу, необхідно також стандартизувати розробку серверної частини, основою якого є PHP-код. Найбільш популярним та застосовуваним у світі є стандарт кодування PSR. Proposing a Standards Recommendation — запропоновані рекомендації стандартів, які розділені на кілька незалежних блоків. Даний стандарт створений групою взаємодії фреймворків (PHP-FIG), що розвивається і постійно

доповнюється, охоплюючи все більше аспектів PHP-програмування [28]. На сайті спільноти можна знайти як затверджені рекомендації, так і рекомендації, що містяться в розробці (http://www.php-fig.org).

Опис існуючих на даний момент стандартів PSR [28,34]:

PSR-1 – основний стандарт кодування. Даний стандарт описує найбільш загальні правила, такі як використання тегів PHP, кодування файлів, поділ місця оголошення функції, класу та місця їх використання, а також найменування класів та методів.

PSR-2 - посібник зі стилю коду. Розширює перший стандарт, регулює такі питання використання у коді як «табуляція чи прогалини», переклади рядків, максимальна довжина рядків, правила оформлення конструкцій та інше.

PSR-4 – стандарт автозавантаження. Це, мабуть, найважливіший і потрібний стандарт PSR. Класи, які реалізують PSR-4, можуть бути завантажені єдиним автозавантажувачем, що дозволяє частинам і компонентам з одного фреймворку або бібліотеки використовуватися в інших проектах.

PSR-6 – інтерфейс кешування. Кешування використовується підвищення продуктивності системи. І PSR-6 дозволяє стандартно зберігати та витягувати дані з кешу, використовуючи уніфікований інтерфейс.

PSR-7 – інтерфейс HTTP-повідомлень. При написанні складних веб-додатків на PHP майже завжди доводиться працювати з HTTP заголовками. Звичайно, мова PHP надає нам вже готові можливості для роботи з ними, такі як суперглобальний масив $_SERVER, функції header(), setcookie() і т.д., проте їх ручний розбір загрожує помилками, та й не завжди можна врахувати всі нюанси роботи з ними. І ось, щоб полегшити роботу розробнику, а також зробити одноманітним інтерфейс взаємодії з протоколом HTTP був прийнятий даний стандарт.

PSR-11 – інтерфейс контейнера. При написанні програми PHP часто доводиться використовувати сторонні компоненти. І щоб не заблукати в великій кількості залежностей були придумані різні методи управління залежностями коду, часто несумісні між собою, які даний стандарт і призводить до спільного знаменника.

PSR-13 – гіпермедіа посилання. Цей інтерфейс полегшує розробку та використання прикладних програмних інтерфейсів (API).

PSR-16 – інтерфейс простого кешування. Є продовженням та покращенням стандарту PSR-6.

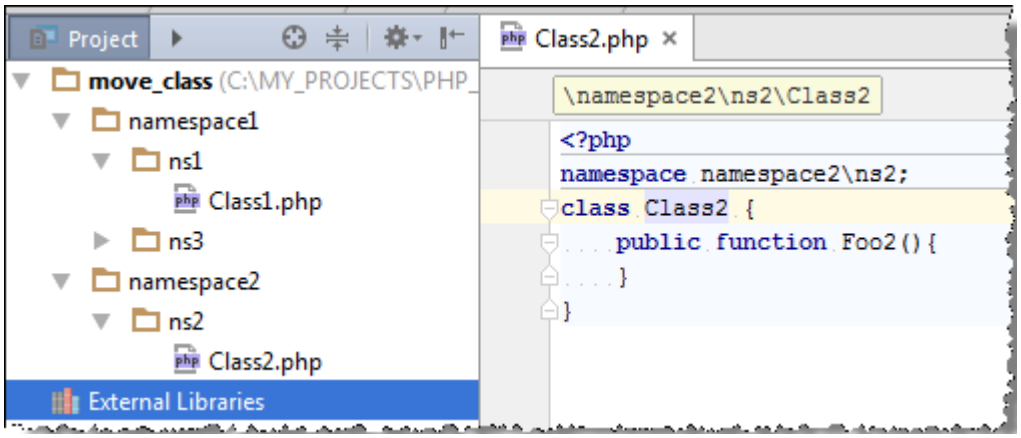

Рисунок 1.2 – Оформлення коду та файлова структура відповідно до PSR

Стандарти, рекомендовані W3C, є відкритими та вільними для впровадження будь-ким, без фінансових відрахувань консорціуму. Організація не проводить сертифікацію на відповідність стандартам та не планує цим займатися. Завдяки цьому рекомендації консорціуму набули широкого поширення, проте це викликає проблему з тим, що деякі виробники дотримуються рекомендацій лише частково. Також використання розділене на кілька ступенів застосування – кожен сам вирішує якою мірою необхідно використання W3C стандартів. З практичних прикладів – це часткове впровадження нових стандартів W3C у роботі веб-браузерів, що зумовлює веб-розробників необхідністю проводити додаткове тестування на кросбраузерність веб-додатку.

Стандарти W3C мають рису добре деталізованих рекомендацій і доступні як веб-майстрам-початківцям, так і експертам-програмістам. Технічні специфікації консорціуму супроводжуються великою кількістю посібників та порад, які полегшують розуміння та впровадження стандартів.

Останнім часом W3C проводить багато роботи з популяризації своїх стандартів, їх вдосконалення та розширення.

Рисунок 1.3 – Один із популярнихстандартів W3C – це HTML версії 5

Принципи та концепції, затверджені W3C:

• Загальнодоступність – при реалізації цього принципу приділяють значну увагу доступності ресурсів глобальної мережі для людей з обмеженими фізичними можливостями.

• Апаратна незалежність - цей принцип стверджує, що ресурси та протоколи не повинні бути кроссплатформенними, тобто. розраховані працювати на будь-якому апаратному забезпеченні.

• Інтернаціоналізація – насамперед передбачає мовний бар'єр та закликає до доступності інформації всіма мовами світу, можливості та рівноправності використання національних шрифтів у документах та системах розроблених на основі стандартів.

• Патентна політика – тут йдеться про патентування рекомендацій W3C для безкоштовного впровадження.

• Багатоформна взаємодія - принцип, що передбачає наявність різних форм і режимів взаємодії програми з людиною.

• Семантична павутина – свіжа та актуальна концепція, яка закликає розміщувати інформацію у більш зрозумілому для комп'ютерів вигляді. Семантичне павутиння зараз реалізується як надбудова над існуючим Всесвітнім павутинням.

• Мобільна павутина - цей принцип визначає, що ресурси Всесвітньої мережі мають бути доступні для мобільних пристроїв. Одні з найпопулярніших стандартів, затверджених W3C: CSS, DOM, HTML, HTTP, PNG, URI/URL, XML HTML

Застосування даних стандартів та методологій дозволить створити

інформаційну систему, що відповідає сучасним технологічним вимогам та має високу конкурентоспроможність.

1.3 Проектування та створення інформаційної системи для торгової організації

1.3.1 Розробка серверної частини інформаційної системи.

Під серверною частиною мається на увазі back-end складова web-програми. Back-end і front-end — це терміни у програмній інженерії, які розрізняють відповідно до принципу поділу відповідальності між зовнішнім поданням та внутрішньою реалізацією відповідно. Front-end – інтерфейс взаємодії між користувачем та основною програмно-апаратною частиною (back-end). Front-end та back-end можуть бути розподілені між однією або декількома системами [19].

Серверна, back-end частина інформаційної системи приймає та обробляє клієнтські запити, звертається до СУБД за необхідності, формує та надсилає відповідь клієнту.

Серверна частина складається з апаратного та програмного забезпечення. Як апаратне забезпечення у компанії ISPSERVER (https://ispserver.ua) був орендований віртуальний виділений сервер з робочими характеристиками: 1GB RAM, 100GB HDD, 2 GHz CPU і попередньо встановленою операційною системою FreeBSD.

Віртуальний сервер (VPS) – це послуга, у якій користувачеві надається частина ресурсів одного фізичного сервера. У плані управління такий сервер практично не відрізняється від фізичного, на нього можна встановити повноцінну операційну систему, і він має один або кілька статичних IP-адрес для виходу в інтернет.

FreeBSD – це вільна операційна система сімейства UNIX (https://www.freebsd.org). ОС поширюється під ліцензією BSD і є популярною системою для мережевих серверів (управління мережею та різноманітні мережеві

служби).

За допомогою системи портів (технологія встановлення ПЗ сторонніх розробників в операційній системі FreeBSD) були встановлені необхідні компоненти інформаційної системи, що забезпечують роботу сервера в ролі веб-сервера: система керування БД (СУБД), HTTP-сервер та PHP-інтерпретатор [20].

Вибір СУБД - це вибір з числа численних програмно-лінгвістичних засобів загального та спеціального призначення, що забезпечують управління створенням та використання баз даних. У системі управління базою даних для організації торгівлі планується зберігати максимальну кількість інформації про поточну бізнес-діяльність, починаючи від каталогу товарів і закінчуючи даними про замовлення клієнтів. При виборі було враховано функціональні можливості, документальну підтримку, наявність літератури, взаємодія з іншими програмними продуктами. Важливими стали такі параметри як можливості системи до роботи, як із малою кількістю даних, і можливість «рости» разом із організацією. Малому бізнесу можуть знадобитися лише обмежені можливості, тобто невелика кількість інформації, але вимоги можуть суттєво зростати протягом часу, і перехід в іншу систему керування базами даних може стати проблемою. Враховано та вартість її використання, наявність на ринку праці вузьких фахівців, що працюють з нею у разі потреби їх найму, наявність достатньої кількості технічної документації та літератури.

При виборі було розглянуто найбільш популярні та розвинені системи управління базою даних [16,18]:

– Oracle Database;

Одна з популярних і сучасних об'єктно-реляційних СУБД від компанії Oracle. Перша версія системи була створена в 70-х роках, існує кілька різних версій для задоволення вузькоспрямованих потреб в інформатизації. Остання з випущених версій варта хмарних середовищ, тобто. розміщується на 1 і більше серверах, дозволяє зберігати та працювати з дуже великими обсягами записів. Є сильна безпека, обумовлена ізоляцією кожної транзакцією у запитах до бази даних.

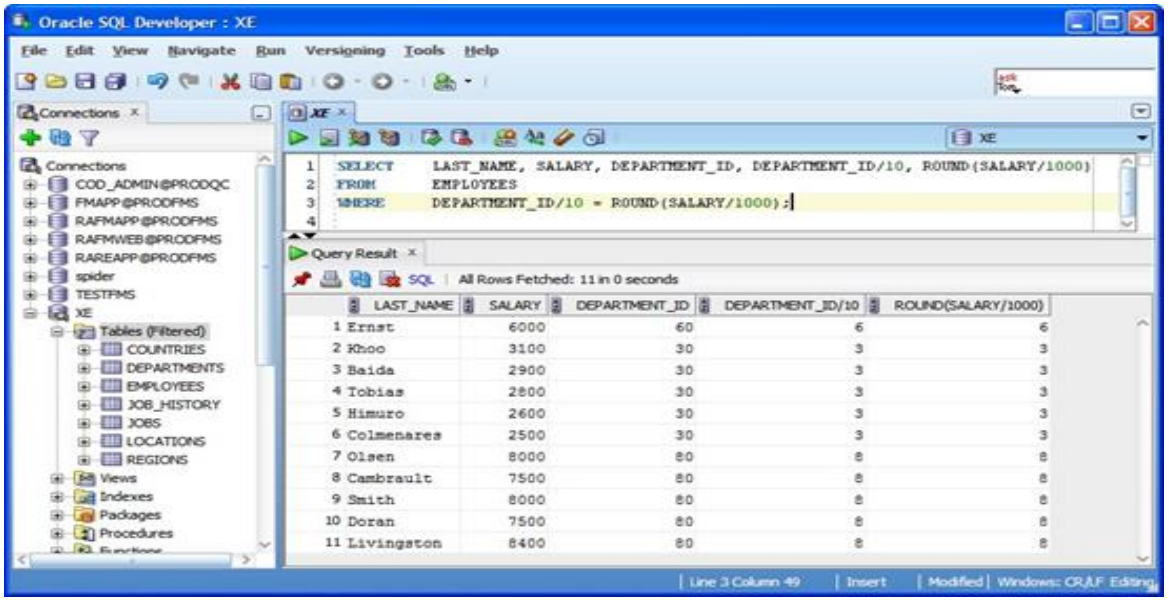

Рисунок 1.4 – Бази даних та таблиці в Oracle SQL Developer

Дана база даних має інноваційні особливості, широкий набір інструментів та задає планку якості та технологічності для багатьох інших СУБД. Oracle Database є платним продуктом, ціна для малих і розвиваються дуже висока, також система вимагає значно краще апаратне забезпечення щодо ресурсів. На українському ринку праці небагато фахівців працюючих із цією БД. Технічна документація та посилання для завантаження доступні на офіційному сайті: https://www.oracle.com

.

– MySQL;

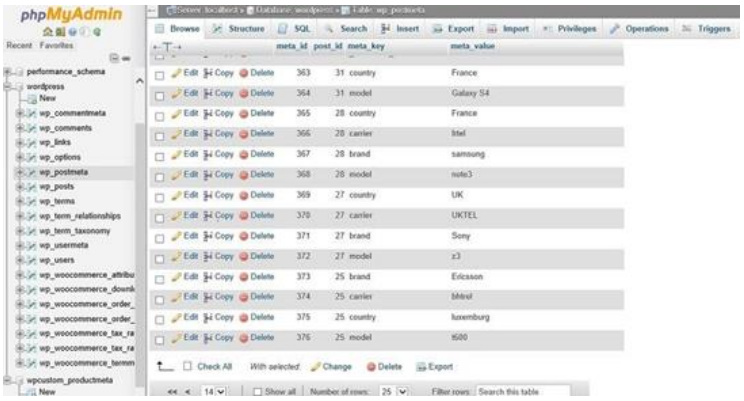

Рисунок 1.5 – Бази та таблиці MySQL в інтерфейсі PhpMyAdmin.

Вільна реляційна СУБД від компанії Oracle. Має велику популярність під час розробки веб-додатків. Розробляється з 90-х років, активно розробляється та

підтримується компанією розробником. Вбудовані інструменти для роботи з MySQL мають багато рішень у галузі програмного забезпечення для веб-застосунків. Є швидкою, надійною та щодо використання – безпечною. Як апаратне забезпечення може використовуватися ЕОМ з досить низькими ресурсами. Має як безкоштовну версію (ліцензія GNUGPL2, так і платну з офіційною підтримкою. На ринку праці є безліч фахівців, що працюють з даною СУБД, а також велика кількість технічної документації, літератури, практичних прикладів використання і готових продуктів у складі яких використовується MySQL. Має можливість розподілу навантаження на кілька серверів.

– MS SQLServer;

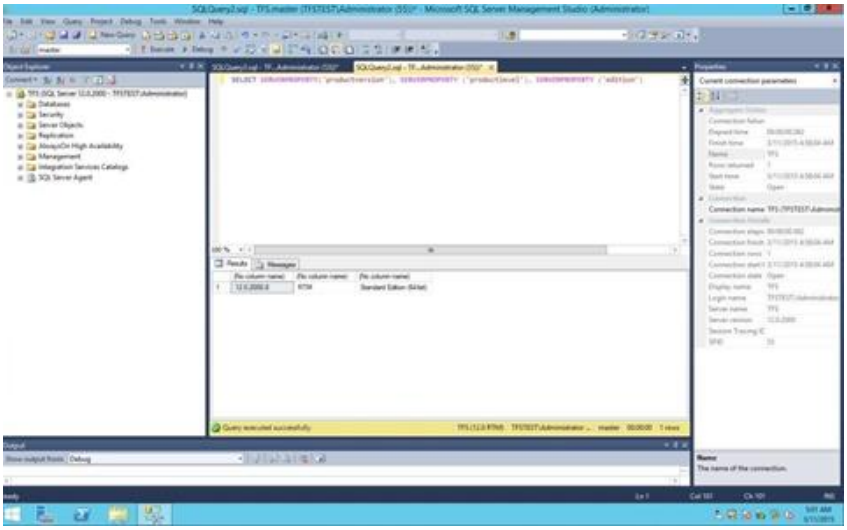

Рисунок 1.6 – SQLServerManager

Система керування реляційними БД розроблена корпорацією MicroSoft. Ядро даної системи може бути призначений для роботи як на локальних серверах, так і на хмарних. У 2016 році стала доступною в операційній системі Linux. Однією з унікальних особливостей версії 2016 є тимчасова підтримка даних, яка дозволяє відстежувати зміни даних з часом. Остання версія Microsoft SQL-сервер також дозволяє здійснювати динамічне маскування даних, що гарантує те, що лише авторизовані користувачі бачитимуть конфіденційні дані. SQL Server є швидким та стабільним засобом, має гарну технічну документаційну основу та безліч

літератури, велику кількість технічних фахівців на ринку праці. Ця СУБД є платним рішенням, із середнім цінником, добре спрямовано працювати з іншими продуктами Microsoft. Технічна документація та посилання для завантаження доступні на офіційному сайті: https://www.microsoft.com/ua/sql-server/ .

– PostgreSQL;

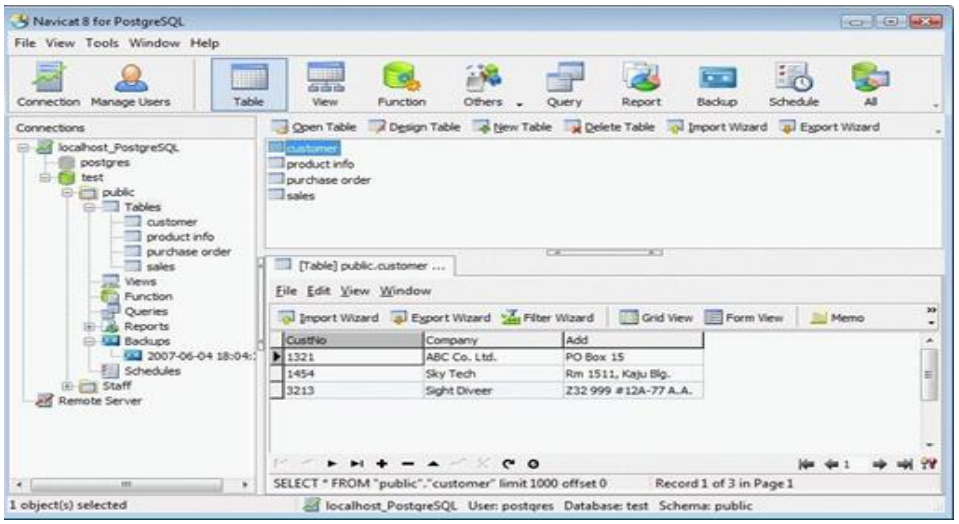

Рисунок 1.7 – Postgre база даних в інтерфейсі програми Navicat 8

Вільна об’єктно реляційна система керування БД, основана на мові SQL , що розповсюджується під ліцензією License PostgreSQL. Може використовуватися для обробки великих даних, що добре масштабується. Розробляється та підтримується однойменною спільнотою та популярний серед веб-баз даних. Є достатня кількість літератури та фахівців, не вимоглива до апаратних ресурсів і добре підходить для проведення великої кількості запитів. Технічна документація та посилання для завантаження доступні на офіційному сайті: https://www.postgresql.org .

– Mariadb;

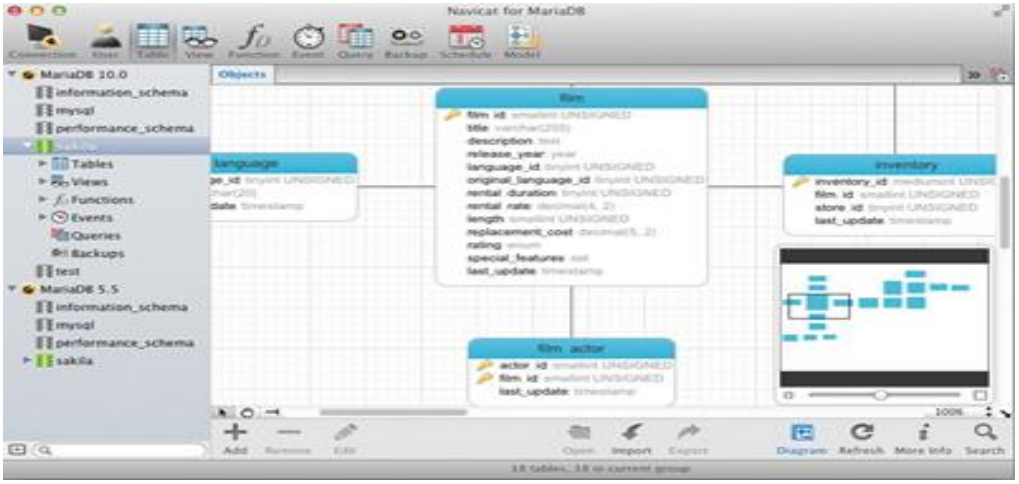

Рисунок 1.8 – База даних MariaDB в інтерфейсі програми Navicat

Ще одна СУБД, що вільно розповсюджується, від компанії MariaDB Corporation Ab і фонду MariaDB Foundation (ліцензія GNUGPL). Система є відгалуженням від СУБД MySQL, що розробляється початковими розробниками MySQL з 2009 року. Система має і платний варіант із розширеною підтримкою та функціоналом. Mariadb - це одна з найшвидше зростаючих субд з безліччю плагінів, проте не має підтримки стороннім програмним забезпеченням, як інші СУБД, однак є повністю сумісним з MySQL продуктом. Є досить безпечним та швидким продуктом, проте стабільність ще не така висока, як у СУБД вже давно присутніх на ринку. Часто сприймається як варіант альтернативний СУБД MySQL. Технічна документація та посилання для завантаження доступні на офіційному сайті: https://mariadb.org .

– DB2;

Платна СУБД від компанії IBM, яка має можливості nosql, що читає json та xml-файли. Є кросплатформним рішенням, має безліч документації та літератури, може бути розміщено у хмарному сховищі, на фізичному сервері, або ж і там, і там одночасно. Базова підтримка доступна лише протягом трьох років; після цього ви повинні заплатити за це. Підходить для: великих організацій, які роблять багато ресурсів і обробляють великі бази даних. Технічна документація та посилання для завантаження доступні на офіційному сайті:

https://www.ibm.com/analytics/us/en/db2/.

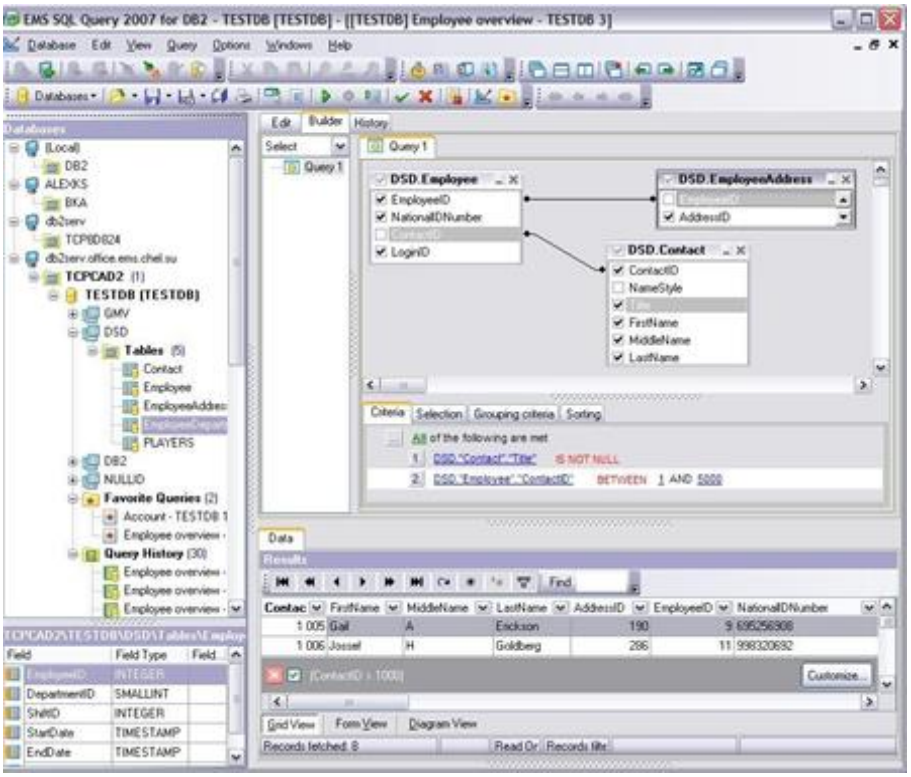

Рисунок 1.9 – База даних DB2 в інтерфейсі EMS SQL Query 2007

З розглянутих СУБД на вирішення завдань торгової організації було обрано MySQL. MySQL розробив Міхаель Віденіус (Michael Widenius, monty@analytikerna.se), вона є невеликою та швидкою реляційною СУБД, заснованою на традиціях Hughes Technologi es Mini SQL (mSQL) .

Характеристиками реляційних моделей СУБД є простота структури даних, усунення надмірності, табличне подяння зручне для користувача, використання формального математичного апарату реляційного числення для обробки даних. Реляційна модель спрямована на організацію даних у вигляді 2-мірних.

СУБД MySQL надає такі переваги при її виборі [20]:

• реляційна модель даних;

• вільно розповсюджується під ліцензією GNU GPL 2;

• добре адаптована для веб-додатків;

• має гарну базу технічної документації та літератури;

• є продуктом, що активно розвивається, з великим досвідом;

• велика кількість спеціалістів на ринку праці;

• безліч плагінів та відмінна інтеграція в інше програмне забезпечення для веб-додатків;

• низькі вимоги до ресурсів апаратного забезпечення.

Важливим етапом проектування та розробки бази даних є створення інфологічної та даталогічної моделей.

Під інфологічною моделлю розуміють опис предметної області, виконане з допомогою спеціальних мовних засобів, які залежать від програмних засобів (це по суті блок-схема алгоритму створення бази даних). Мета інфологічного моделювання - забезпечення найбільш природних для людини способів збору та подання тієї інформації, яку передбачається зберігати у створеній базі даних. Основними конструктивними елементами інфологічних моделей є сутності, зв'язок між ними та його властивості (атрибути).

Вимоги до інфологічної моделі:

• адекватне відображення предметної галузі;

• несуперечність;

• має відображати погляди та потреби всіх користувачів системи;

• однозначне трактування моделей;

• модель має бути кінцевою;

• модель має бути легко розширюваною, тобто мати можливість введення нових (видалення) даних без зміни раніше визначених;

• повинна мати властивості композиції та декомпозиції (зміцнювати базу даних або розщеплювати);

• має бути легко реалізованою на ЕОМ;

• має бути незалежною від обладнання та мов організації бази даних на ЕОМ.

Даталогічне проектування - це створення моделі, що є моделлю логічного рівня і є відображенням логічних зв'язків між елементами даних. Ця модель будується в одиницях допустимих конкретної СУБД. Опис логічної структури з допомогою коштів СУБД називається схемою. Так як це здійснюється за допомогою конкретної СУБД, моделі повинні бути описані мовою опису даних цієї

СУБД. Такий опис і називають датологічною моделлю даних.

База даних складається з наступних таблиць:

• Таблиця «Клієнти» - містить інформацію про приватних та юридичних осіб, яким реалізується товар.

• Таблиця «Вибрані клієнти» - містить інформацію про те, у яких користувачів які клієнти додані як вибрані.

• Таблиця «Постачальники» - містить інформацію про організації, які здійснюють постачання продукції.

• Таблиця «Замовлення» – замовлення, їх статус та дані.

• Таблиця «Товари» - продукт, реалізований фірмою та її характеристики.

• Таблиця «Бренди» – реєстр брендів товарів та інформації про них.

• Таблиця "Групи" - групи товарів.

• Таблиця «Конфіг» – містить основні налаштування ІС.

• Таблиця "Користувачі" - користувачі системи, їх властивості та права.

• Таблиця Логи - містить логи основних дій користувачів.

• Таблиця «Сторінки» - службова інформація про сторінки/модулі ІС.

• Таблиця "Курс валюти" - таблиця зберігає актуальний курс валют для виведення на головні сторінки ІВ та графіки.

• Таблиця "Чат" - містить повідомлення чату.

• Таблиця «Завдання» - у таблиці є всі дані для здійснення функціоналу «Завдання».

Для складання даталогічної та інфологічної моделей бази даних було використано безкоштовний спеціалізований веб-додаток DbDesigner (http://dbdesigner.net).

Як видно із спроектованих моделей бази даних, всі таблиці певним чином пов'язані між собою. Наприклад, у таблиці «Замовлення» та "Вибрані клієнти" мають поля, що пов'язують їх з таблицею "Клієнти" і хоча таблиця "Замовлення" ніяк не пов'язує себе безпосередньо з таблицею «Вибрані клієнти», через загальну їм таблицю «Клієнти» вибудовується характерна для реляційних СУБД зв'язок. Таким чином зв'язуються між собою та всі інші таблиці.

Таким чином, створена вже традиційна реляційна БД, яка має залежності між даними, але при цьому уникає надмірності. Якщо системі необхідно показати замовлення, воно запитує дані про них з таблиці «Замовлення» при цьому, не запитуючи список клієнтів і список товарів, що можна зробити при перегляді конкретного замовлення, але і в цьому випадку вимагати з бази даних доведеться не всі товари та всіх клієнтів, а лише певних, які стосуються цього замовлення.

Як програмне забезпечення на сервер були встановлені СУБД mysql-server і mysql-client (https://dev.mysql.com/downloads/mysql/) як простий і швидкий доступ до управління СУБД. За допомогою MySQL клієнта (та запитів до СУБД були створені таблиці з полями певних типів, вказані унікальні індекси (id), яким було присвоєно властивість «AUTO_INCREMENT», що вказує на те, що порядковий номер буде генеруватися автоматично (попереднє значення + 1) у міру додавання нових рядків до таблиці.

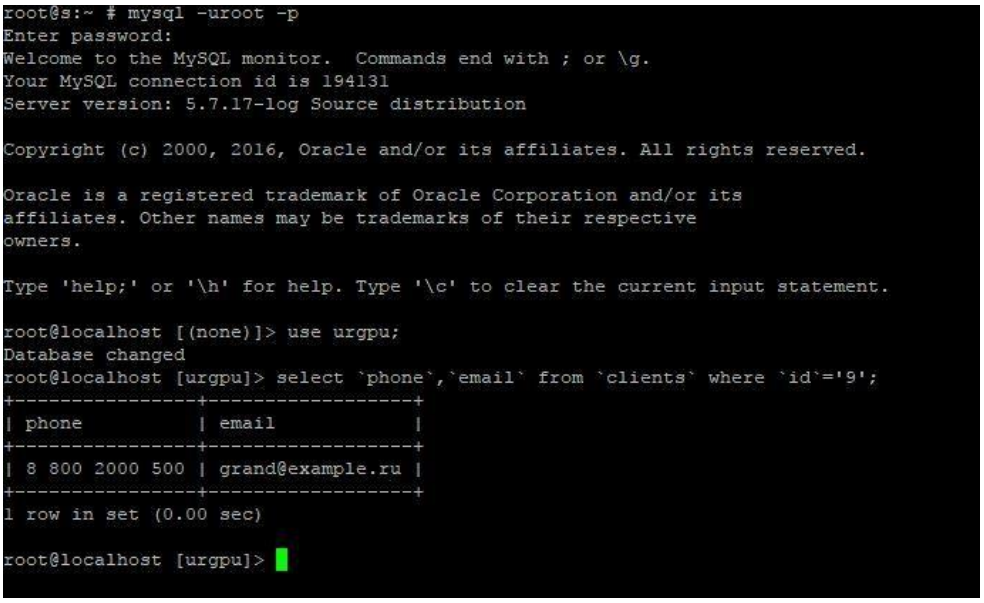

Рисунок 1.10 – Робота з СУБД MySQL через консоль MySQL-Client

Таким чином, розроблено базу даних торгової організації – основний компонент системи. Дана БД дозволяє зберігати, обробляти та доповнювати інформацію організації в єдиному місці.

Наступним кроком з налаштування back-end частини стала установка HTTP-сервера, що вміє інтерпретувати PHP-код.

Apache HTTP Server – це вільно розповсюджуваний HTTP сервер під ліцензією Apache License 2.0 (http://www.apache.org/). Він здійснює 2 основні функції: взаємодіє з клієнтом за протоколом HTTP і компілює php-код за допомогою відповідного вбудованого модуля. Ця програма є найбільш актуальною та використовується для створення web-серверів [2]. Також Apache кросплатформовий, надійний та гнучкий у конфігурації [30].

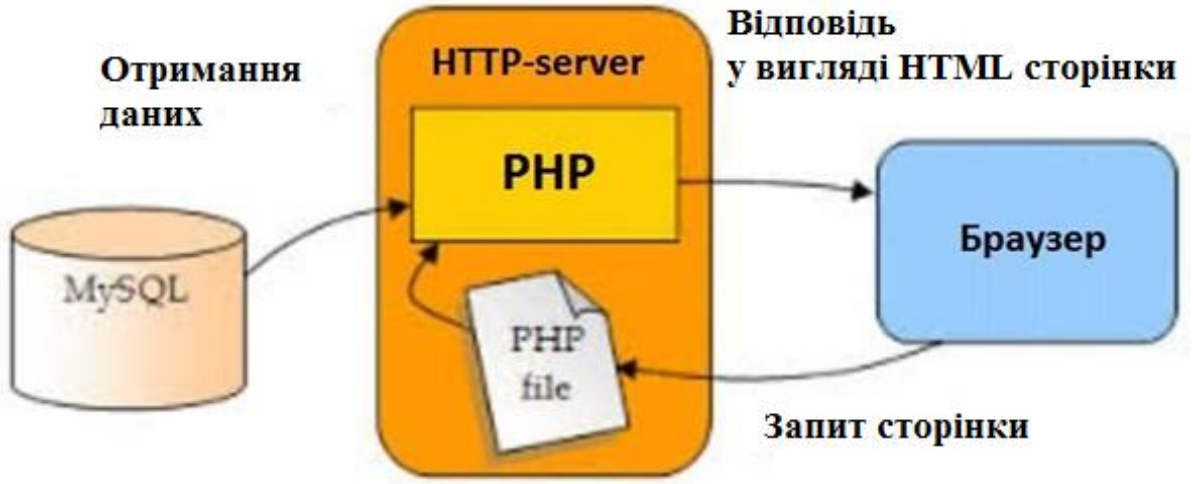

Рисунок 1.11– Схема роботи веб-сервера

Як мову програмування для написання back-end частини web-додатку інформаційної системи було обрано PHP.

PHP (передпроцесор гіпертексту) – скриптова мова програмування загального призначення, яка використовується для розробкпри роробці веб-додатків [7,13]. Використання PHP здійснюється під PHP License, що є open source ліцензією, покликаної стимулювати широке поширення вихідного коду. Ліцензія дозволяє розповсюдження матеріалів у вихідній або бінарній формі із змінами або без таких за дотримання таких умов [17]:

1. увімкнення тексту ліцензії PHP,

2. слово PHP не може бути включене в назву продукту без письмового підтвердження від PHP Group, наступне повідомлення має бути включене в будь-яку форму матеріалів, що розповсюджуються: "This product includes <http://www.php.net/software/>".

Якщо для програмного коду PHP використовується ліцензія PHP License, то керівництво PHP поширюється за ліцензією CreativeComonsAtribution3.0 License. Крім того, деякі файли можуть поширюватись під іншими ліцензіями. Для поширення таких файлів необхідно дотримуватись умов цих ліцензій.

Традиційно PHP-код виконується в php-файлах - це звичайні текстові файли, що мають розширення php, редагувати їх можна в будь-якому стандартному текстовому процесорі, які постачаються з будь-якою операційною системою як вільний і безкоштовний для використання текстовий редактор [9,13].

У процесі написання було використано середовище розробки ПЗ Geany (https://www.geany.org). Це текстовий процесор зі спеціальними функціями для розробників, такими як: автоматичне встановлення закриваючих тегів, автозавершення назв функцій, класів, змінних, підсвічування вихідного коду, підтримка великої кількості кодувань.

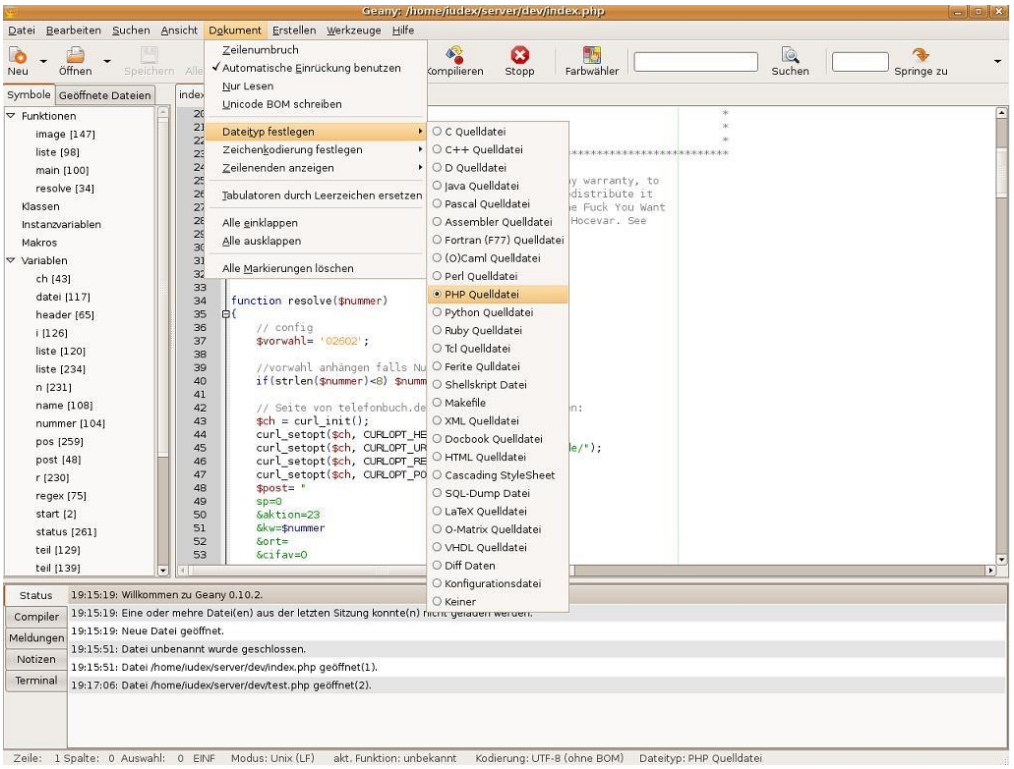

Рисунок 1.12 – Текстовий процесор Geany

Написаний PHP код був поділений на різні файли, у кожному з яких він відповідає за різні завдання. Такий поділ спрощує написання та редагування

проекту, а також дає можливість підключати лише необхідний код для вирішення поточного завдання користувача [39].

Файлова структура веб-додатку:

```
./assets/
./ckeditor/
./fancybox/
./favicon/
./fontAwesome/
./base.css
./base.js
./remPolyfill.js
./funcs.js
./normalize.css
./components/
/pages/
/safemysql/
./authorization.php
./config.php
./functions.php
./lang.php
./valute_rate.php
./images/
./templates/
./404.html
./about.html
./all.html
./authorization.html
./client_edit.html
./clients.html
./goods.html
./goods_edit.html
./logs.html
./orders.html
./providers.html
./provider_edit.html
./order_edit.html
./passchange.html
./group_edit.html
./main_right0.html
./main_right1.html
./main_right2.html
./ya_market.html
./uploads/
./.htaccess
./COPYING
```

```
./README
./index.php
./ya_market.php
```

Директорія «assets» містить допоміжні файли, такі як JavaScript скрипти та JavaScript плагіни, а також CSS таблиці каскадних стилів (додаток 1) та файли шрифтів.

• Директорія «components» містить основний PHP-код веб-програми.

• Директорія "images" містить зображення.

• Директорія templates містить HTML шаблони сторінок.

• Директорія «uploads» зберігає в собі завантажені файли користувача.

• Файли COPYING та README є обов'язковими файлами для ліцензії інформаційної системи.

• Файл ".htaccess" перетворює людино-зручну HTTP адресу в GET-запити, які в свою чергу обробляє index.php.

• Файл "valute_rate.php" здійснює оновлення курсу валют, запитуючи цю інформацію на сайті ЦБРФ і потім оновлюючи її в базі даних. Цей файл запускається автоматично у планувальнику.

• Файл "index.php" - найважливіший файл. Він підключає всі інші компоненти веб-застосунку залежно від того, яку сторінку запитав користувач (додаток 1).

• Файл «ya_market.php» – вивантаження для інтеграції ІВ з рекламним майданчиком «Яндекс.Маркет». Містить інформацію про організацію та список товарів.

Компоненти, що використовуються інформаційною системою, розроблені сторонніми розробниками:

• SafeMySql 1.0.0 - PHP клас для безпечного та зручного поводження з MySQL-запитами. Ліцензія Apache License 2.0.

(http://phpfaq.ru/safemysql)

• PHPMailer 6.0.1 - PHP клас для створення та надсилання листів електронною поштою. Ліцензія: GNU LGPLv2.1. (https://github.com/PHPMailer/PHPMailer)

• jQuery 1.12.4 - JavaScript бібліотека для легкого доступу до будь-якого

елемента DOM, звернень до атрибутів і вмісту елементів DOM, маніпуляцій з ними. Ліцензія: MIT License. (https://jquery.com)

• fontAwesome 4.7.0 - шрифт/набір іконок з інструментарієм CSS. Ліцензії: SIL OFL 1.1, MIT License, CC BY 3.0. (http://fontawesome.io/)

• Fancybox 3 - jQuery плагін для відображення у модальних вікнах зображень, відео та іншого контенту. Ліцензія: GNU GPLv3. (http://fancybox.net/)

• css3-mediaqueries - JavaScript скрипт для підтримки CSS Media Queries у застарілих браузерах (https://github.com/livingston/css3-mediaqueries-js)

• HTML5 Shiv 3.7.3 – JavaScript скрипт для підтримки HTML5 тегів у застарілих браузерах. Ліцензії: MIT та GPL2. (https://github.com/aFarkas/html5shiv)

• remPolyfill 1.3.2 - Javascript скрипт для підтримки одиниць вимірювання rem у CSS у застарілих браузерах. Ліцензія: MIT License. (https://github.com/chuckcarpenter/REM-unit-polyfill)

• CKEditor 5 1.0.0-alpha.2 - Набір компонентів JavaScript, що дозволяє створити зручне рішення для редагування тексту. Ліцензії: GPL, LGPL, MPL. (https://github.com/ckeditor/ckeditor5)

• Normalize.css 7.0.0 - CSS-файл, для забезпеченняя кращої узгодженості між браузерами (https://github.com/necolas/normalize.css)

• Google Charts - це інструмент для побудови різних діаграм у веб-додатках, що надається за допомогою API Google. Безкоштовне використання згідно умов, Google APIs Terms of Service. (https://developers.google.com/chart/)

• Jquery UI 1.12.1 - JavaScript набір взаємодій, ефектів, віджетів та тем, побудованих на основі jQuery JavaScript Library. Ліцензія: GNU GPLv3. (http://jqueryui.com/)

Загалом PHP-код виконує такі завдання:

• Підключається до бази даних

• Здійснює запити до бази даних

• Обробляє дані надіслані користувачем

• Обробляє дані, отримані з БД

• Завантажує відповідну сторінку HTML-шаблон

• Вставляє оброблені дані в HTML-шаблон

• Надсилає готову сторінку до веб-браузера користувачам.

Наступний крок щодо створення інформаційної системи — це розробка графічного інтерфейсу для введення, виведення, об'єднання даних в інформацію та інших корисних операцій у бізнесовій діяльності торгової організації.

1.4 Розробка клієнтської частини інформаційної системи

Front-end частиною проекту є сторінки, які віддаються клієнту back-end частиною HTML, які містять HTML, CSS, JavaScript, зображення та шрифти. [32, 35, 37]

JavaScript – мультипарадигменна (об'єктно-орієнтована, імперативна, функціональна) мова програмування. У браузерах використовується як мова сценаріїв, надавай сторінці межу інтерактивності. Назва мови є товарним знаком Oracle

Корпорація, однак сама мова нікому не належить (https://developer.mozilla.org/en-US/docs/Web/JavaScript) [31,33].

Коли back-end складова веб-програми надсилає користувачеві html/css (також можливий javascript та інший код) разом із вмістом, який запросив людина, браузер користувача повинен вивести на екран готову сторінку. І тому він перетворює HTML-разметку в об'єктну модель документа (DOM), а CSS-разметку - в об'єктну модель таблиці стилів (CSSOM), тобто. створює DOM та CSSOM моделі.

```
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link href="style.css" rel="stylesheet">
    <title>Critical Path</title>
  </head>
  <body>
    <p>Hello <span>web performance</span> students!</p>
    <div><img src="awesome-photo.jpg"></div>
  </body>
</html>
```

Рисунок 1. 13 – Приклад моделі DOM

Основні етапи формування DOM:

1. Перетворення. Браузер перетворює байти з HTML-файлу на символи, ґрунтуючись на наведеному у файлі кодуванні (наприклад, UTF-8).

2. Розмітка. На основі стандарту W3C HTML5 браузер виділяє серед символів теги в кутових дужках, такі як інші. Кожен тег має своє значення і свій набір правил.

3. Створення об'єктів. За допомогою HTML-тегів браузер виділяє у документі об'єкти з певними властивостями.

4. Формування DOM. Об'єкти утворюють деревоподібну структуру, що повторює ієрархію HTML-файлу, в якому одні теги розміщуються в інші. Так, об'єкт p поміщається під body, а об'єкт body, своєю чергою, під html, тощо.

Внаслідок цього утворюється об'єктна модель документа (DOM), за допомогою якої браузер продовжує обробляти сторінку.

Всі ці дії (перетворення байтів на символи, визначення розмітки, створення об'єктів та формування DOM) браузер повинен виконувати щоразу при обробці HTML-розмітки. Цей процес займає деякий час, особливо при обробці великої кількості тегів, якими реалізується розмітка гіпертексту.

Щоб обробити таблицю каскадних стилів CSS, браузер повинен виконати ті ж дії, що і з HTML-документом: байти з CSS-файлу перетворюються на символи, символи - на теги, а теги - на об'єкти, які утворюють модель CSSOM.

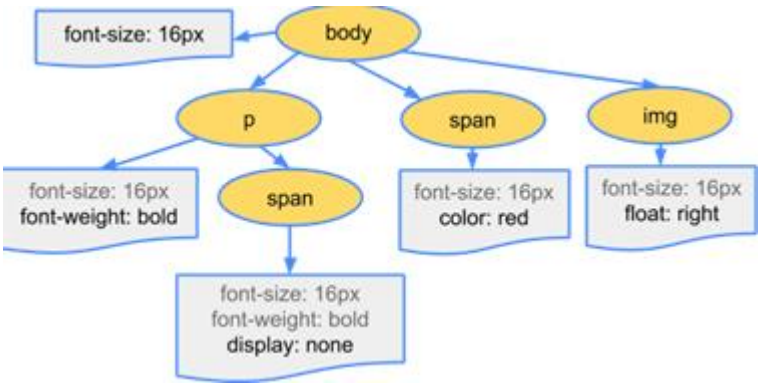

Рисунок 1.14 – Приклад CSSOM моделі

У PHP-коді інформаційної системи торгової організації написана спеціальна функція data2template. При використанні цієї функції код надсилає їй дані, які ми запросили з БД. Потім функція запитує той шаблон, який відповідає сторінці, на якій знаходиться користувач. Для кожної зі сторінок вже створено відповідний шаблон:

goods_edit.html – сторінка редагування товару; provider_edit.html – сторінка редагування постачальника; orders.html – сторінка замовлень; і т.д.

Такі ж шаблони є і для інших сторінок, які відображаються користувачем інформаційної системи.

Усередині шаблонів міститься HTML певної сторінки та мітка, що говорить PHP-коду (функції data2template), куди вставити дані в цьому HTML шаблоні. Міток може бути кілька.

Приклад шаблону для сторінки "Товари бренду Curaprox":

<h2>: <!--[brand]--></h2>

<table><thead><td>Найменування</td><td>Бренд</td><td>Ціна</td></t head><tbody><!--[table]--></ tbody></table>

У наведеному прикладі можна побачити 2 мітки для даних: <!--[brand]--> та <!--[table]-->. Замість першої функції вставить назву бренду, а замість другої сформовану в PHP-коді таблицю з товарів цього бренду.

Наведені вище шаблони не є повноцінними сторінками HTML, а містять

лише основу контенту веб-додатку. Вони відсутні такі обов'язкові теги HTML як body, head, також у них відсутня основний дизайн програми та CSS.

Так як основні елементи дизайну веб-додатки є статичними (шапка, підвал, фон), то для зручності вони виділені в окремий шаблон all.html. Таким чином, при необхідності змінити основні елементи розмітки веб-програми відпадає потреба змінювати їх у шаблонах кожної сторінки, зміни потрібні лише у вищезгаданому файлі.

Файл all.html використовується при завантаженні будь-якої сторінки програми та містить мітки: заголовок сторінки, контент сторінки, додатковий код між <head></head> тегами.

Логіка використання шаблонів виглядає так:

1. Програма отримує запит на вибрану користувачем сторінку;

2. Запитуються дані відповідно до обраної сторінки з БД;

3. Додаток обробляє дані їх подальшого відображення користувачеві (наприклад, становить таблицю);

4. Завантажується шаблон, який відповідає сторінці;

5. Згідно з мітками, додаток має в своєму розпорядженні оброблені дані в шаблоні;

6. Завантажується основний шаблон all.html, згідно з його мітками, до нього вставляються заголовок сторінки та вже оброблений шаблон сторінки;

7. Отриманий шаблон надсилається веб-браузеру користувача.

Шаблон all.html є основою та здебільшого веб-програми, в ньому підключаються скрипти JavaScript та таблиці CSS. Починається сторінка із вказівки (веб-браузеру) типу поточного документа DTD (document type definition, опис типу документа): <!DOCTYPE html> - позначає нову, 5 версію HTML.

В інтернеті тривалий час використовувалися стандарти HTML 4.01, XHTML 1.0 та XHTML 1.1. Веб-сторінки на практиці верстались з використанням змішаних особливостей, представлених різними специфікаціями, включаючи специфікації програмних продуктів, наприклад, веб-браузерів, а також загальновживаних прийомів, що склалися. HTML5 був створений як єдина мова розмітки, яка могла б

поєднувати синтаксичні норми HTML і XHTML. Він розширює, покращує та раціоналізує розмітку документів, а також додає єдиний API для складних веб-додатків.

Після оголошення типу документа йдуть основні елементи для сторінок html (рис. 19):

• html – елемент верхнього рівня в документі, для HTML;

• head - призначений для зберігання інших елементів, мета яких – допомогти браузеру працювати з даними. Між відкриваючим та закриваючим тегами head вказано кодування, мову, файли, що підключаються (CSS, JavaScript і шрифти).

• body – зберігає вміст веб-сторінвідображена у вікні браузера. Тобто. між відкриваючим та закриває тегами даного елемента знаходиться все те, що бачить користувач: «шапка» сайту, заголовки, контент сторінки та «підвал» сайту.

```
<!DOCTYPE html>
<html lang="ru">
<head> <!--голова сайта-->
    <meta charset="utf-8">
    <meta name="description" content="Создание сайтов на HTML5">
    <meta name="keywords" content="HTML5, CSS3, JavaScript">
    <title>Разработка сайтов с помощью HTML5</title>
    <link rel="stylesheet" href="style.css">
</head>
<body> <!--тело сайта-->
</body>
</html>
```

Рисунок 1.15 – Приклад структури HTMLдокументу

У додатку використано кодування UTF-8, воно вказане як у "head" HTML, так і на початку основного PHP-файлу. Крім того, бази даних також зберігають інформацію в цьому кодуванні.

UTF-8 – одне з загальноприйнятих і стандартизованих кодувань тексту, яка дозволяє зберігати символи Юнікоду, використовуючи змінну кількість байт (1-6). Кодування знайшло широке застосування в UNIX-подібних операційних системах та веб-просторі [13].

Якщо HTML вказує, де відображаються елементи веб-сторінки, то CSS-правила задають як саме. Самі правила також містяться в основному шаблоні та прописані у спеціальному HTML-тегу <style type=”text/css”>.

Перед тим, як описувати стилі HTML-елементів у CSS був доданий normalize.css — це набір стилів, який забезпечує для HTML-елементів кращу кросбраузерність у стандартних стилях. Проект є продуктом сотень годинника глибокого дослідження відмінностей між початковими стилями браузера. Це дослідження провели Ніколас Галахер, @necolas та Джонатан Ніл, @jon_neal. Коли елемент має різні типові стилі в різних браузерах, normalize.css там, де це можливо, прагне зробити ці стилі сумісними і відповідними сучасним стандартам.

Цілі normalize.css:

• зберігати корисні налаштування браузера, а не прати їх;

• нормалізувати стилі для широкого кола HTML-елементів;

• коригувати помилки та основні невідповідності браузера;

• удосконалювати юзабіліті непомітними покращеннями;

• пояснювати код, використовуючи коментарі та детальну документацію. Normalize.css вільно розповсюджується під ліцензією MIT.

За допомогою CSS у додатку задані властивості різних елементів дизайну, наприклад, розмір, колір, сімейство, варіант написання шрифту (рис. 20) як для всієї сторінки, так і окремо для деяких елементів, наприклад, заголовків. Задано загальний фон сторінки та фон контенту, обрамлення таблиць та відступи у їхніх комірках.

```
body {
    background-color: #d0e4fe;
}

h1 {
    color: orange;
    text-align: center;
}

p {
    font-family: "Times New Roman";
    font-size: 20px;
}
```

Рисунок 1.16 – Приклад стилів CSS для HTML-елементів

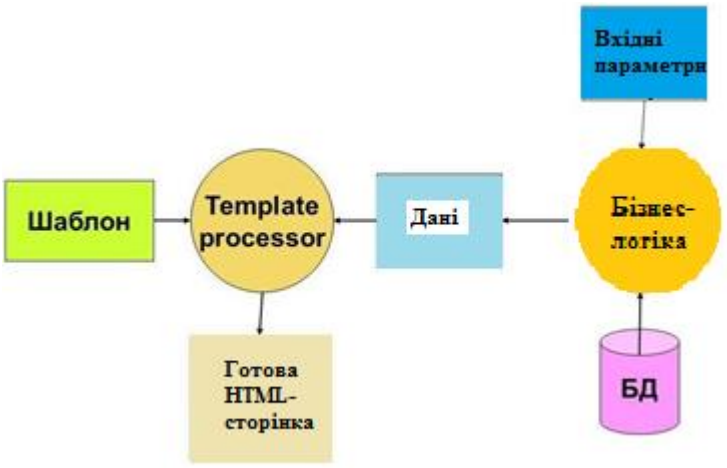

Рисунок 1.17 – Діаграма роботи веб-додатку

Розробивши back-end та front-end частини веб-додатку інформаційної системи – отримано необхідний для функціонування інформаційної системи функціонал. Працюючи як клієнь-серверний додаток, система позбавляє кінцевого користувача необхідності установки додаткового програмного забезпечення, оскільки основні процеси веб-додатки відбуваються на стороні сервера.

2. ТЕСТУВАННЯ ТА ОГЛЯД ІНФОРМАЦІЙНОЇ СИСТЕМИ

Особливості тестування, орієнтованого на клієнт-серверну взаємодію веб-додатку, передбачають наявність навичок знаходити дефекти як на стороні клієнта (веб-браузер), так і сервері. Важливо врахувати дефекти, що виникають при взаємодії сервера та клієнта, такі дефекти часто пов'язані з проблемами бізнес-логіки (функціональні дефекти) інформаційної системи. Функціональне тестування спрямоване на те, щоб кожна функція веб-сайту працювала відповідно до вимог специфікації. Тестування функціональності веб-сайту показує, що робить система. Саме тому важливо спочатку проаналізувати аспекти інформаційної системи, що розробляється, і провести проектування, такий підхід спочатку дозволить уникнути даних помилок.

Для знаходження помилок у коді розробленого веб-додатку були використані стандартні засоби та методики. Для PHP-коду це висновок помилок при виконанні скрипта. Таке тестування проводилося у процесі написання коду та перевірки його функціонування. При значних, критичних помилках PHP-код не виконується і виводить відповідне повідомлення, вказавши проблемну ділянку коду (рис. 21.).

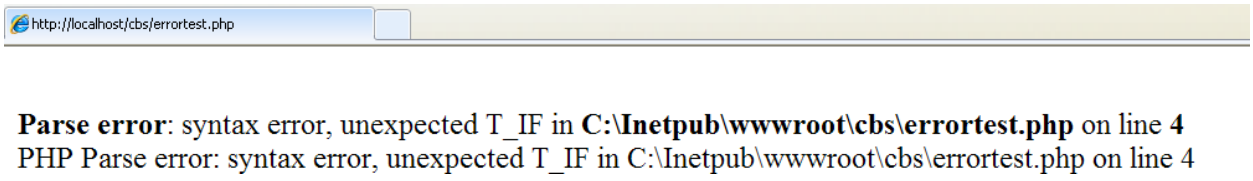

Рисунок 2.1 – Виведення PHP-помилки у браузері

У PHP є широкий функціонал з налаштування виведення та обробки таких помилок – їх можна виводити у клієнтський браузер, спеціальний текстовий файл, записувати в базу даних, нікуди не виводити. PHP надає функціонал для створення власних обробок помилок, завдяки яким їх можна доповнити помилками зі своєю класифікацією (наприклад при помилках в бізнес-логіці). Стандартно помилки в PHP класифікуються так:

• фатальні (непереборні, при виникненні таких скрипт припиняє роботу);

• не фатальні (усунуті, скрипт не припиняє роботу);

• змішані (фатальні, але тільки якщо не оброблені функцією, визначеною користувачем);

• попередження (не є помилками, однак, вказують на недоліки у коді, що виконується).

Існує і рідше використовуваний спосіб перевірки PHP коду на наявність помилок - це стороннє програмне забезпечення, що перевіряє завантажений в них код або онлайн сервіси, як правило, вони перевіряють тільки синтаксис, не виконуючи його.

У PHP є широкий функціонал з налаштування виведення та обробки таких помилок – їх можна виводити у клієнтський браузер, спеціальний текстовий файл, записувати в базу даних, нікуди не виводити. PHP надає функціонал для створення власних обробок помилок, завдяки яким їх можна доповнити помилками зі своєю класифікацією (наприклад при помилках в бізнес-логіці). Стандартно помилки в PHP класифікуються так:

• фатальні (непереборні, при виникненні таких скрипт припиняє роботу);

• не фатальні (усунуті, скрипт не припиняє роботу);

• змішані (фатальні, але тільки якщо не оброблені функцією, визначеною користувачем);

• попередження (не є помилками, однак, вказують на недоліки у коді, що виконується).

Існує і рідше використовуваний спосіб перевірки PHP коду на наявність помилок - це стороннє програмне забезпечення, що перевіряє завантажений в них код або онлайн сервіси, як правило, вони перевіряють тільки синтаксис, не виконуючи його.

Тестування помилок на стороні клієнта (у веб-браузері) багато в чому здійснювалося завдяки наданому в браузерах інструментарію для веб-розробників.

Рисунок 2.2 – Інструменти веб-розробників у Mozilla Firefox

Інструменти для веб-розробників - це комплекс засобів, що представляє собою спеціальну панель і включає такі засоби як [25]:

• Інспектор – дозволяє редагувати DOM-модель у реальному часі (всі зміни відразу відображаються на сторінці);

• Консоль – незамінний інструмент, що показує помилки у виконанні JavaScript, CSS, помилки під час завантаження файлів тощо;

• Запити (мережа) – показує всі http запити, їх заголовки, відповіді, зміст. Відобразить відповідну інформацію, якщо не вдалося завантажити зображення з сервера;

• Сховище – показує та дозволяє редагувати дані, що містяться в різних сховищах клієнта, таких як Cookie, Indexed DB, сесійне сховище, сховище кешу.

• Стилі – подібний до інспектора, але орієнтований на CSS;

• Пам'ять – вказує на виконання та завантаження чого витрачається відповідний ресурс клієнта;

• Профайлер – потужний інструмент для глибокого аналізу процесів, що відбуваються при завантаженні сторінки, побудові DOM, застосуванні стилів та багато іншого;

• Адаптивний дизайн – допомагає розробляти та перевіряти сторінку на правильне відображення в екранах різного розміру, що особливо актуально в даний час, коли сайти однаково користуються як з мобільних пристроїв, так і з

десктопних.

Також залежно від браузера та його версії є й інші інструменти, такі як лінійка, скріншот сторінки, піпетка, редактор JavaScript та інші.

Написання коду, що виконується, не завжди є запорукою його ефективного функціонування та отримання однакового результату в різних браузерах. Така проблема сумісності багато в чому виключається під час написання коду з дотриманням відповідних стандартів і повністю виключається при тестуванні додатковими перевірками.

Для цього W3C (Консорціум Всесвітньої павутини) створив автоматизовані тести сумісності для CSS, HTML і JavaScript. Ці тести проводяться як на окремих ділянках коду, так і на готових веб-додатках доступних у глобальній мережі. За результатами перевірки виводяться рекомендації щодо виправлення. Такі перевірки доповнюють багато IT-гігантів (таких як Google та Яндекс) зацікавлені у зростанні якості інтернет ресурсів [11]. Найпопулярнішим тестом у корпорації Google [1,11] є тест на швидкість завантаження веб-сторінок і наявність властивості адаптивності у веб-інтерфейсу для відображення на різних пристроях [24,29,36].

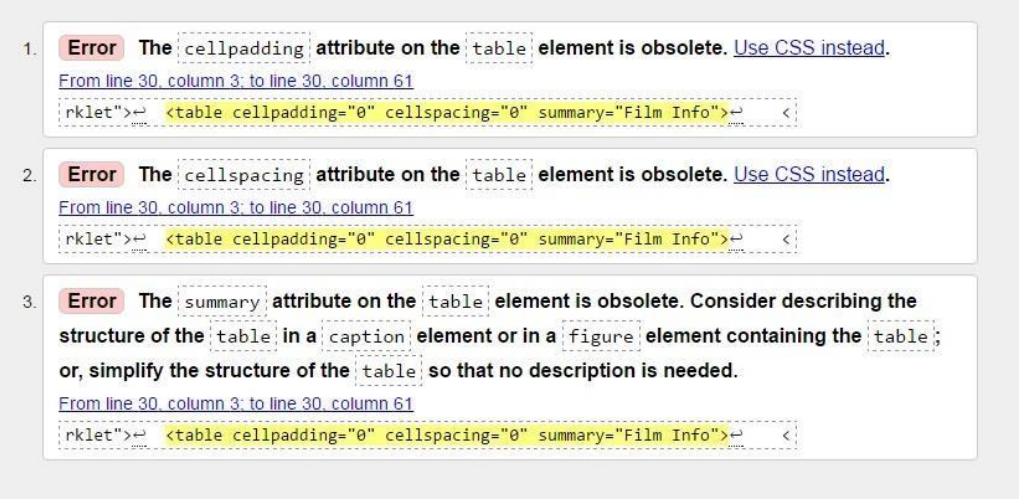

Рисунок 2.3 – Помилки HTML показані валідатором коду від W3C

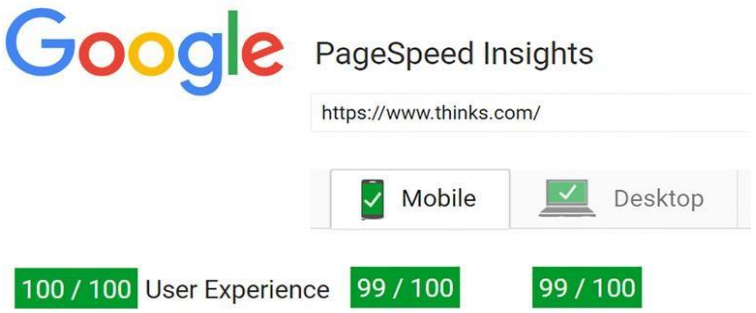

Рисунок 2.4 – Google PageSpeed Insights – перевірка на швидкість завантаження сайту

Окрім стандартних та автоматизованих засобів та методик знаходження дефектів, кожен веб-розробник з появою досвіду починає складати свою стратегію та свої «чек аркуші» для перевірки інформаційних систем та їх складових. Такі перевірки можуть бути вузькоспрямованими і складатися безпосередньо для даної системи. час проектування або мати список для перевірки частих помилок у будь-якій інформаційній системі.

Чек-лист для тестування веб-інтерфейсу користувача:

• Тестування посилань

o Вихідні посилання

o Коректність внутрішніх посилань

o Відсутність посилань, що ведуть до однієї сторінки

o Чи є сторінки, на які не вказано посилання

o Відсутність непрацюючих посилань

• Тестування форм для всіх сторінок

o Дійсність вхідних даних

o Допустимі значення для поля даних

o Неприпустимі вхідні значення для поля даних

o Параметри форм, у яких можливе видалення чи будь-яка інша модифікація даних.

• Тестування cookies

o Перевірте сайт із вимкненими cookies

o Перевірте сайт із включеними cookies

о Переконайтеся, що файли cookie зашифровані перед записом на комп'ютер користувача

о Перевірте аспекти безпеки під час видалення файлів cookies.

о Якщо cookies мають тривалість дії, слід перевірити, чи активні вони у зазначений період часу.

• Переконайтеся, що веб-сайт доступний для пошукових машин.

Багато в чому тестування полегшує уніфіковане написання коду та повторне використання компонентів. У середовищі веб-інтерфейсів у цьому сприяє розробка веб-застосування, а при написанні PHP-коду стандарти PSR. Важливим є підготовка до створення бази даних, що передбачає її проектування шляхом складання інфологічної та даталогічної моделей.

Обстежуючи область тестування веб-застосунку не можна не згадати маркетингові дослідження. Такі дослідження повинні проводитися фахівцями компетентними у сфері маркетингу, однак, за раніше описаними методологіями веб-розробки, кожен учасник команди (або фахівець, який одноосібно розробляє ІС) повинен мати розуміння та частку компетенції на кожному щаблі розробки та проектування інформаційної системи.

Найбільш універсальним та поширеним методом у маркетингових дослідженнях є A/B тестування. Мета даного тестування – це покращення цільового показника. Це досягається шляхом порівняння підконтрольної групи із тест-групами, в яких змінено один чи кілька показників. Такий метод часто використовується у веб-додатках для покращення показників дизайну, з типових прикладів можна виділити: вплив схеми кольорів, розміщення та розміри елементів веб-інтерфейсу сайту з метою збільшити важливий цільовий показник – конверсія. У такому прикладі A/B тестування показує різним користувачам різний варіант виконання дизайну, а з накопичення необхідної кількості показів, відбувається порівняння числових показників досягнення мети, з чого вибирається найбільш підходящий варіант дизайну. Використання в комплексі перерахованих вище методів та інструментів для тестування помилок у коді, логіці та процесах взаємодії інформаційних систем, заснованих на веб-додатках, дозволило не тільки

виключити виникнення помилок, але й підвищити робочі характеристики та важливі показники на більш якісний рівень.

Мінімальні вимоги до апаратно-програмного забезпечення:

• Для сервера з 20 підключеними користувачами:

- o Процесор - Intel Core i3-2xxx, або аналогічний
- o ОП - 1 гб
- o Жорсткий диск - 1 гб
- o Інтернет з'єднання - 8 Мбіт
- o Виділена IP-адреса
- o PHP версії >= 5.6
- o MySQL server версії >= 5.7
- o MySQL client версії >= 5.7
- o Apache >= 2.4

• Для користувача:

- o Оперативна пам'ять - 1 гб
- o Платформа - 32 або 64 розрядна
- o Операційна система - з графічним інтерфейсом
- o Інтернет з'єднання - 2 мбіт
- o Браузер — підтримувана розробником версія, підтримка JavaScript

Питання ліцензування є важливою складовою проектування та розробки інформаційних систем та програмних продуктів, яка часто опускається при створенні інформаційних систем приватними розробниками. У цій галузі головним чином фігурує поняття автор, який може мати виняткове право (майнове право використання і розпорядження твором) на розроблену інформаційну систему, і немайновим правом (право ім'я, декларація про недоторканність твори та інше). Автором є фізична особа, творча діяльність якої призвела до створення твори, також можливе співавторство. Правовласником ж, може бути сам автор, або організація, яка отримала виняткові права від автора за договором відчуження, або, якщо твір створювалося за службовим завданням. Також виняткове право можна отримати у спадок. Проте, немайнові права залишаються за автором назавжди [27].

Як правовий інструмент, який визначає та поширює програмне забезпечення, виступають ліцензії на програмне забезпечення. У рамках закону відсутність ліцензії на програмне забезпечення забороняє таке використовувати іншим особам, крім автора. Ліцензії на ПЗ поділяються на вільні та невільні (пропрієтарні). Тому проектуючи інформаційну систему, необхідно заздалегідь перейматися питанням ліцензування майбутнього твору. Для ліцензування даної інформаційної системи було вирішено використовувати Open-Source.

Таким чином, в цілому розроблена система відповідає критеріям відкритих систем, код веб-додатку (основного компонента системи) написаний з використанням сучасних стандартів та методологій. Інформаційна система пройшла тестування, налагодження та ліцензування як вільно розповсюджуване програмне забезпечення.

3. БЕЗПЕКА ЖИТТЄДІЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Характеристика життєдіяльності людини у системі «людина-машина – середовище існування».

У теперішній час, світ машин та різноманітних механізмів все більше набуває популярності. Він буквально щодня супроводжує людину. Суспільство стає все залежнішим від цього. Сучасна, складна техніка-механізмів машин, включаючи електронно-обчислювальних. Вони тепер очолюють життєво необхідні системи: військові об’єкти, інформаційні системи, наденергоємні- атомні електростанції та штучні космічні об’єкти.

Машина – це те, за допомогою чого виконуються різні операції. Наприклад операції для зміни місця об’єктів, трансформація енергії, матеріалів, збереження та систематизація інформації [27]. Завдяки цьому інтелектальний та фізичний рівень людини зростає. Існує три основних типи машин:

– роботи;

– енергетичні;

– інформаційні.

Машино-двигуни – це пристрої за допомогою яких енергія будь якого виду перетворюється в механічну. Електродвигуни та двигуни внутрішнього згоряння, турбіни, теплові машини. Група машин поділяється на технологічні та транспортні. Технологічні – металообробні верстати, поліграфічне устаткування, прокатні стани. Транспортні – літаки, автомобілі,конвєйєри, підіймачі, водогони.

Інформаційні машини – самі складні пристрої, розраховані на переробку, одержання та збереження інформації. ЕОМ, механічні інтегратори, арифмометри, персональні комп’ютери. Хоча в повному розумінні слова електронно обчислювальні машини не є машиною, тому що в ній механічні пристрої мають допоміжне значення [28].

Слід зазначити , що більшість сучасних машин недосконалі і тому вони, витвори людини, забруднюють довкілля, змінюють його в негативний бік. Дано рердження звісно схиляється здебільшого до енергетичних або ж роботичних

машин, але все ж тему кваліфікафікаційної роботи дана проблема зачіпає в повній мірі. До прикладу можна навести структурну роботу та архітектуру веб-сайтів, що попередньо розглядалось в розділі другому розділі, а саме в пункті про моделювання архітектури та структури інтернет інтернет-мазазину. В даній частині кваліфікаційної роботи сказано, що клієнт-серверна аріхітектура здебільшого розділена на три ланки, одна з яких серверна частина, що містить в собі всю необхідну інформацію для роботи сервісів, інтернет-магазинів та інших різних платформ, що містять в собі виликий обсях даних. Особливо під цю категорію підпадають онлан-сервіси, які спеціалізуються саме на розміщенні користувацьких даних. Дані веб-сайти потребують наявності значних накопичувальних ресурсів, а ці ресурси як привило – це спільні або приватні мережеві ферми.

Такі ферми споживають дуже багато електроенергії та потребують значної кількості комплектуючих, що у свою чергу дуже погано впливає на екологію. Саме тому розробники зі всього світу працюють над алгоритмами компресії файлів для того щоб зменшити ресурс використання накопичувальних пристроїв. Але все ж слід підкреслити, що ідеальних механізмів небуває, тому навіть найсучасніші на найоптимізованіші програми не зможуть відмінити антропогенний вплив інформаційних машин. З цього можна зробити висновок, що неможливо назвати машину, механізм антропогенного походження, який був би ідеальним і не шкодив би здоров'ю людини, не забруднював би середовище життя.

3.2 Психофізіологічне розвантаження для працівників

У сучасному світі все більше набуває затребуваності професія програміста. До прикладу напрямок веб-дизайн, створення веб-сайтів. Це не тільки програмування та розробка дизайну. Сам процес містить в собі детальний аналіз проекту, співпраця з замовником, пошук рішень для отримання поставлених цілей проекту.

Враховуючи тривалий час роботи за комп’ютером для працівників необхідно забезпечити психофізіологічне розвантаження, що відбувається в спеціально облаштованих для цього приміщеннях (кімната психологічного розвантаження).

Аутогенне тренування – це метод який будується на свідомому застосуванні сукупності пов’язаних між собою технік психічної саморегуляції та виконанні простих фізичних вправ, додаючи словесне самонавіювання [29]. Саме це тренування пропонується для занять з психофізіологічного розвантаження працівників. Також, кімнати для тренувань обладнують із спеціальним інтер’єром та яскравим оформленням. Сеанс поділено на три етапи, що відповідають стадіям відновлювального процесу.

Перший етап – абстрагування. Працівники повинні розслабитись та абстрагуватись від робочої обстановки, що служить етапу залишкового збудження. Грає спокійна, повільна та мелодійна музика. Можливо звуки моря, шум лісу чи пташиний спів. Зайнявши зручну позу, працівники звикають, адаптуються та психологічно настроюються до наступних етапів.

Другий – затишок (заспокоєння). Цей етап відповідає за регенеративне гальмування. На екрані телевізора чи проектора відбувається показ фотографій чи коротких відео з зображенням квітів, степів, лісів, ставків, пустель чи гладких поверхонь. В цей ж час у навушниках звучить заспокійлива музика, на фоні якої тричі повільно повторюється формула аутогенного тренування:

– « я повністю розслаблений, я спокійний»;

– « моє дихання спокійне, воно рівне»;

– « моє тіло важке, гаряче, повністю розслаблене. У мене легка голова, холодний лоб».

Зелене світло служить функціональним освітленням. Протягом усього сеансу воно поступово знижується. На кінці екран та світло зовсім вимикається на декілька хвилин.

Третій – активізація. Третій період відповідає етапу підвищеної збудженості. Спочатку світло вимкнене. За деякий час на екрані чи моніторі з’являється яскрава пляма червоного кольору. Яскравість та розміри якої поступово збільшуються. На

кінці сеансу грає весела та бадьора музика. Тричі повторюються формули аутогенного тренування. Формули включають в себе глибоке вдихання та глибоке довге видихання:

– « я бадьорий, повний енергії, у мене хороший настрій»;

– « я веселий, свіжий та готовий діяти».

За потреби та бажання під час музичної програми пропонується вимовляти окремі фрази. Наприклад, навіювання гарного настрою, відпочинку, сили чи бадьорості.

Заняття з психофізіологічних розвантажень можуть бути проведені за єдиною програмою та складатись із двох етапів. Кожен із них по 5 хвилин:

– перший- абсолютне розслаблення;

– другий- активізація працездатності.

Після занять психофізіологічного розвантаження зменшується втома та з’являються нові сили, бадьорість та гарний настрій, що допомагає веб-розробникам продовжувати працювати над своїми задачами.

Варто звернути увагу, якими методами психофізіологічного розвантаження сучасні ІТ компанії користуються. До прикладу можна навести компанію Google, яка підтримує найпопулярнішу пошукову систему, займається розробкою програмного забезпечення та володіє найпопулярнішими сервісами, які використовують люди в всіх куточках світу. Люди, які працюють в цій компанії здебільшого розробники програмного забезпечення, зокрема веб-сервісів. Вони дійсно працюють над тим щоб робити життя своїх користувачів краще. Але інша сторона медалі – це психологічне вигорання працівників, адже ця робота вимагаю значну кількість розумових ресурсів, що власне викликає даний наслідок. Для вирішення даної проблеми компанія вжила чимало заходів. До прикладу можна навести:

– безплатна їжа для всіх працівників, що сприяє повноцінному харчуванню та економія часу, який би був витрачений на пошук та замовлення;

– зони відпочинку де працівники можуть відпочити та зайнятись психофізіологічним розвантаженням;

– безплатні спортзали, басейни та все необхідне для заняття спортом, що теж сприяє психофізіологічному розвантаженню;

– можливість працювати в домашніх умовах, у випадках якщо працівникам комфортніше працювати саме так.

Саме такий підхід дозволяє Google залучати найкращих. Та оприємувати від працівників набагато більш позитивні відгуки про досвід роботи в прорівнянні з іншими світовими компаніями. Виходячи з даної інформації було зроблено висновок, що рівень психофізичного розвантаження для працівників на дуже високому рівні.

ВИСНОВОК

Аналіз потреб комерційних організацій допоміг виявити недоліки пропозицій на ринку програмного забезпечення. Було вирішено створити інформаційну систему вирішальну потребу організації торгівлі в організації зберігання, обробки та наповнення інформації з метою покращення ефективності бізнес-процесів. При проектуванні та створенні були вивчені та застосовані сучасні стандарти, що дозволило відповідати кінцевому продукту актуальним вимогам та мати можливість розширюватись та доповнюватися новими функціями у майбутньому. Вибір реалізації та компонентів інформаційної системи проводився з розрахунком створити систему, в якій вирішено повністю чи частково недоліки існуючих пропозицій ринку ПЗ.

Створена інформаційна система враховує бюджетні можливості організацій з низьким фінансуванням інформатизації - це виявляється у низьких апаратних вимогах та використанні лише вільно-розповсюджуваного програмного забезпечення. При необхідності апаратну складову можна модифікувати з метою збільшення ресурсів для більшої кількості користувачів, при цьому внесення змін до програмних компонентів ІС не потрібно. ПЗ є кросплатформеним, тому може встановлюватися на серверному обладнанні з різними операційними системами. Такі показники значно полегшують використання інформаційної системи в організацію. Аналогічні переваги є на стороні користувача. Реалізація інтерфейсу за допомогою веб-програми дозволяє працювати з системою на будь-якому персональному комп'ютері з графічним інтерфейсом, інтернет-браузером та активним інтернет-з'єднанням. Такий підхід позбавляє кінцевого користувача від необхідності встановлювати додаткове програмне забезпечення та дає доступ до інформаційної системи повсюдно.

Веб-додаток, що відповідає за інтерфейс та алгоритми обробки даних, є гнучким і легко настроюваним компонентом системи. Завдяки відкритому вихідному коду будь-який розробник при необхідності може модифікувати веб-

додаток під специфічні потреби організації. Реалізація з урахуванням сучасних стандартів та методологій дозволить здійснити такі модифікації, змінивши лише необхідний мінімум компонентів, а також швидко розібратися в алгоритмах та процесах застосування стороннім розробникам.

Цілі та завдання досягнуто. Створена інформаційна система надає можливість організації торгівлі вирішити низку проблем зі зберіганням і наповненням бази даних, автоматизувати обробку даних, скоротити час та ресурси для аналізу та аналітики, краще координувати дії співробітників і стежити за показниками торговельної діяльності в реальному часі. Впровадження такої інформаційної системи спростить досягнення цілей організації торгівлі, відкриє нові можливості та підвищить показники ефективності комерційної діяльності.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Husni M. Improved Information Retrieval Performance on SQL Database Using Data Adapter / M. Husni. – Madrid: Grupo Planeta, 2018. – 127 c.

2. – (Materials Science and Engineering).

3. Noaman A. eCloudDB: A Unified API for Secure SQL and NoSQL Cloud Databases. / A. Noaman, E. Noaman, A. Noaman. – Washington: Wiley, 2019. – 124 c. – (Proceedings of the 2019 3rd International Conference on Cloud and Big Data Computing).

4. Mukherjee S. Indexes in Microsoft SQL Server / Sourav Mukherjee. – New-York: HarperCollins, 2019. – 53 c.

5. Efficiency test of Microsoft SQL Server 2016 / T.Balla, T. Radvanyi,

6. S. Kiraly, R. Kiraly. –Amsterdam: Wolters Kluwer, 2018. – 192 c.

7. Dejan S. SQL Server 2016 Developer's Guide / S. Dejan, M. Radivojevic, W. Durkin. – Zagreb: Cornelsen, 2017. – 77 c.

8. William A. SQL server 2017 administration inside out / Assaf William. – Washington: RELX Group, 2018. – 36 c.

9. Mukherjee S. Popular SQL server database encryption choices / Sourav Mukherjee. – New-York: HarperCollins, 2019. – 37 c.

10. Переваги і недоліки інтернет магазину для ведення бізнесу [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://officem.com.ua/uk/poleznaja-informatsija/preimuschestva-i-nedostatki-internet-magazina-dlja-vedenija-biznesa.

11. Етапи створення інтернет-магазину [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://essuir.sumdu.edu.ua/bitstream-download/123456789/38235/1/Ivanova.pdf.

12. Тестування програмного забезпечення [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.

13. Laravel — лідер серед PHP фреймворків, схвалений розробниками [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: http://savelink.org.ua/laravel-lider-sered-php-frejmvorkiv-shvalenij-rozrobnikami/.

14. Синтаксичний цукор [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Синтаксичний_цукор.

15. MySQL/Посібник для початківців [Електронний ресурс]. – Режим доступу до ресурсу: https://wiki.gentoo.org/wiki/MySQL/Startup_Guide/ua.

16. Основні поняття реляційних БД: нормалізація, зв’язок та ключі [Електронний ресурс]. – 2011. – Режим доступу до ресурсу:

17. https://bondarenko.dn.ua/osnovni-ponyattya-relyatsijnih-bd-normalizatsiya-zv-yazok-ta-klyuchi/.

18. Що таке архітектура програмного забезпечення [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://soandso.biz/blog/software-engineering/arhitektura-programnogo-zabezpechennya.html.

19. Змерзлий І. Клієнт-серверна архітектура та ролі серверів. [Електронний ресурс] / Іван Змерзлий. – 2017. – Режим доступу до ресурсу: https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-роль-серверів-9893d8048229.

20. HTTP [Електронний ресурс] // 2022 – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTP.

21. Структурне програмування [Електронний ресурс] – Режим доступу до ресурсу: https://ua5.org/osnprog/200-strukturne-programuvannja.html.

22. HTTP Session [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/9.x/session.

23. Різниця між cookie та сесією [Електронний ресурс] – Режим доступу до ресурсу: https://myrusakov.ru/php-cookie-session.html.

24. Проектування інтерфейсу для сайту та онлайн сервісу [Електронний ресурс] – Режим доступу до ресурсу: https://dlogic.com.ua/ua/service/proektuvannya-interfeisiv/.

25. Кольори для дизайну сайту та їх вплив на користувачів [Електронний

ресурс] – Режим доступу до ресурсу: https://www.itk-agency.com/kolori-dlya-dizajnu-sajtu-ta-yih-vpliv-na-koristuvachiv/.

26. Структура веб-сайтів. Етапи створення сайту. Засоби автоматизованої розробки веб-сайтів, редактор веб-сайтів з графічним інтерфейсом. Хостинг.

27. phpMyAdmin [Електронний ресурс] – Режим доступу до ресурсу: https://docs.phpmyadmin.net/uk/latest/intro.html.

28. Chmel M. SQL Server 2017 Administrator's Guide: One stop solution for DBAs to monitor, manage, and maintain enterprise databases / M. Chmel, V. Muzny. – Warsaw: Klett, 2017. – 52 с.

29. Желібо Є. П. Безпека життєдіяльності / Є. П. Желібо, В. В. Зацарний. – Київ: Каравела, 2008. – 344 с.

30. Mukherjee S. SQL Server Development Best Practices / Sourav Mukherjee. – New-York: HarperCollins, 2019. – 41 с.

31. Johnson B. Using Git in Visual Studio 2019 / Bruce Johnson. – New- York: Apress, 2020. – 141 с.

32. Посвістак В.С., Демківська Т.І. Клієнт-серверна архітектура та її використання при розробці програмного забезпечення. / Інформаційні технології в науці, виробництві та підприємництві – 2020. – 14 с.

33. Висоцький А.В. Поняття клієнт-серверної архітектури. / Тези доп. міжнародної науково-практичної інтернет-конференції. – Дніпро, 2020. – 8 с.

34. Михайленко А.М. Клієнт-серверна система управління задачами./ КПІ ім. Ігоря Сікорського. – Київ, 2019. – 7 с.

35. Войтко В., Денисюк П.. Особливості розробки серверних додатків клієнт-серверної архітектури. – 2017. – 44 с.

36. Божок Р.Ю. Клієнт–серверна система підтримки учбового процесу. / КПІ ім. Ігоря Сікорського. – Київ, 2020. – 55 с.

37. Geary Nigel. The client-server architecture in a mixed database environment. / Data Distribution: Managing the Environment. – Routledge, 2018.– 77 с.

38. Langa S. Management aspects of client/server computing. – 2019. –13 с.

39. Kumar Santosh. A Review on Client-Server based applications and research

opportunity. / International Journal of Recent Scientific Research. – 2019.

40. – 24 с.

41. Madalina Eleonora. Providing Security for Client-Server Applications.

42. – 2019. – 8 с.

43. Rabin Steven. Developing Workstation-Based Client/Server Applications. – Auerbach Publications, 2018. – 44 с.

44. Sahoo Niranjan, Rajashree Shukla. Use of Client Monitoring Client- Server Data Security and Health Management. – 2019. – 25 с.

45. Romanov E.L., Troshina G.V., Menzhulin S.A. Client-server application framework based on an object-oriented network model. – IOP Publishing, 2020. – 35 с.

46. Радченко Д. Ю., Колодний В.В.. Технологія та моделі архітектури клієнт-серверної взаємодії. – ВНТУ, 2019. – 17 с.

47. Висоцький А.В. Поняття клієнт-серверної архітектури. – Дніпро, 2020. – 98 с.

48. Снопок А.І. Багатопотоковий клієнт-серверний додаток. / КПІ ім. Ігоря Сікорського. – Київ, 2020. – 29 с.

49. Sharp John. Microsoft visual C# step by step. – Microsoft Press, 2018.– 33 с.

50. Bouna Praveenkumar. Visual Studio Code for C# Developers. – 2022.– 11 с.

51. Ilic Milos. The difference between ADO .NET and Entity Framework in software. – 2019. – 14 с.

52. Strauss D. Working with Visual Studio 2019. Getting Started with Visual Studio 2019. / Dirk Strauss. – New-York: Apress, 2020. – 201-202 с.

53. Snell P. Microsoft Visual Studio 2015 / P. Snell, L. Snell, M. Snell. – New-York: HarperCollins, 2019. – 109 с.

54. Bhandari P. Consuming Microsoft Cognitive APIs. / P. Bhandari, N. Bhandari, A. Bhandari. – Berkeley: Californial, 2018. – 116 с.

55. Development of Dissolved Gas Analysis Analyzing Program using Visual Studio Program. / P.Kunagonniyomrattana, P. Kunagonniyomrattana, T. Kunagonniyomrattana, C. Supakit. – Tokyo: Shuesisha, 2019. – 178 с.

56. Buananno E. Functional programming in C#. Manning / Enrico Buananno. –

San-Jose: HarperCollins, 2018. – 111 с.

57. Kunal C. Mastering Visual Studio 2017 / Clowdhury Kunal. – Los- Angeles: RELX Group, 2017. – 69 с

58. Sharp J. Microsoft Visual C# 2013 Step by Step. Pearson Education / John Sharp. – Boston: McGraw, 2017. – 41 с.

59. Schrotenboer S. Illustrated C# 7: The C# Language Presented Clearly, Concisely, and Visually / S. Schrotenboer, D. Schrotenboer, C. Schrotenboer. – New-York: Apress, 2018. – 108 с.

60. Guerin B. ASP. NET con C# en Visual Studio 2017: diseño y desarrollo de aplicaciones Web / Brice-Arnaud Guerin. – Madrid: Grupo Santillana, 2018. – 50 с.

61. Watts G. Developing a Declarative Analysis Language: LINQToROOT. / Gordon Watts. – Washington: Apress, 2019. – 21 с. – (EPJ Web of Conferences).

62. Muller, Nathan J. Client-server Architecture and Implementation. - Auberbach Publication, 2020. – 186 с.

63. Методичні вказівки до виконання дипломної роботи освітнього рівня ―бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

64. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

65. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, Я.І. Кінах, Г.Б. Цуприк - Тернопіль: ТНТУ, 2017-38 с

66. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Інформаційні технології видобутку

даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

67. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с.

68. Нога В.В., Цуприк Г.Б. Інформаційні технології при створенні автоматизованих систем для задач збору та збереження інформації. Збірник тез доповідей VII Міжнародної науково-технічної конференції молодих учених та студентів. Актуальні задачі сучасних технологій – Тернопіль 28-29 листопада 2018. — Т. : ТНТУ, 2018. — Том 2. — С. 131.

69. І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз). Навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів. Конспект лекцій : Навчальний посібник / І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк – Тернопіль : ТНТУ , 2017. – 64 с.

70. Якість та тестування ПЗ. URL: https://dl.tntu.edu.ua/index.php

71. Java - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java.

72. Firebase - Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Firebase (дата звернення: 30.05.2023).

73. Лабораторна робота №3 [Електронний ресурс] – Режим доступу до ресурсу: https://dl.tntu.edu.ua/get.php/lab20172018/lab3.pdf.

74. Firebase Test Lab [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/test-lab.

75. What is automation testing? URL: https://www.globalapptesting.com/blog/what-is-automation-testing.

ДОДАТКИ

ДОДАТОК А

"./index.php" - головний файл веб-додатку.

```
<?php
/* Стандартний сервер може не відображати помилки виконання php, наступні 2 директиви локально (при виконанні цього скрипта і коду підключеного цим скриптом) задають зворотні налаштування. При необхідності відображення помилок слід зробити їх діючими (розкомментувати) - це може стати в нагоді при налагодженні та тестуванні веб-додатку. */
ini_set("display_errors",1);
error_reporting(E_ALL); /* Якого рівня помилки показувати */


/* Вказуємо створити сесію, якщо вона ще не створена (за нею ідентифікується браузер клієнта) */
session_start();


/* Заголовок вказує у якому кодуванні працюємо. Вказано кодування UTF-8/*/
header('Content-Type: text/html; charset=utf-8');


/* Задаємо часовий пояс, в якому ми працюємо */
date_default_timezone_set('Asia/Yekaterinburg');

/* Підключається php-код із файлу safemysql, він містить клас safemysql для роботи з базою даних. */ include_once('components/safemysql/safemysql.class.php');

/* Підключається php-код із файлу functions, він містить основні PHP функції системи. */

include_once('components/functions.php');


/* Підключається php-код із файлу config, він містить основні налаштування системи. */
include_once('components/config.php');


/* Підключається php-код із файлу lang, він містить усі мовні константи.
*/ include_once('components/lang.php');

/* Створюю об'єкт $db з екземпляром класу safemysql (при створенні клас підключається до БД, тому передаю параметри підключення: користувач БД, пароль, назва бази, хост) */
$db = новий SafeMySQL(['user' => CNF_MYSQL_USER, 'pass' =>
CNF_MYSQL_PASSWORD, 'db' => CNF_MYSQL_DB, 'host' => CNF_MYSQL_HOST]);

/* Визначається головна змінна в якій зберігатимуться всі службові дані */

/* Вивантажуються дані про торговельну базу з СУБД */
$isys_config = $db->getRow("SELECT * FROM `config`");
```

```
/* Підключається php-код із файлу authorization, він здійснює авторизацію користувача */
include_once('components/authorization.php');


/* Записуємо get-параметри ЧПУ */
$isys_url = [0 => $_GET['path1'], 1 => $_GET['path2'], 2 => $_GET['path3']];

/* Запитуємо дані про запитувану сторінку із СУБД */
$isys_page = $_GET['path1'];


/* Якщо жодна сторінка не запрошена, то завантажуємо головну */ if($isys_url[0] == null) {
$isys_url[0] = 'main';
}


$isys_page = $db->getRow("SELECT * FROM `pages` WHERE `link`=?s AND
`rights` LIKE '%?p,%' AND `del` IS NULL LIMIT 1", $isys_url[0], $isys_user['rights']);

/* Якщо така сторінка не існує, то показуємо помилку 404 */ if($isys_page == false) {
$isys_page = '404';
$isys_page = $db->getRow("SELECT * FROM `pages` WHERE `link`=?s AND `rights`LIKE
'%?p,%' AND `del` IS NULL LIMIT 1",$isys_page,$isys_user [ 'rights']);
}


/* Завантажуємо файл з PHP-кодом даної сторінки */
include_once('components/pages/'.$isys_page['file'].'.php');

/* Додаткові дані для виведення в головний шаблон системи */
include_once('components/template_functions.php');

/* Вивантажуємо дані сторінки в головний шаблон системи */ echo data2template('all',
[
'page_content' => $isys_page['content'],

'page_title' => $isys_page['title'], 'page_head' => $isys_page['head'], 'page_foot' =>
$isys_page['foot'],
'system_name' => $isys_config['system_name'], 'system_version' =>
$isys_config['system_version'],
'system_version_date' =>
time2date($isys_config['system_version_date'],8),
'org_name' => $isys_config['org_name'], 'user_name' =>
$isys_user['name'],
'user_post' => $isys_user['post'],
'page_add_taskCount' => $isys_page['add_taskCount'], 'page_add_orderCount' =>
$isys_page['add_orderCount']
]
);
?>
```

"./templates/all.html" - головний HTML шаблон веб-додатку.

```
<!DOCTYPE html>
<html lang="ru-RU">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<meta name="viewport" content="width=1200, maximum-

scale=1.0">


<title><!--[page_title]--></title>
<link rel="apple-touch-icon" sizes="180x180"

href="/assets/favicon/apple-touch-icon.png">
<link rel="icon" type="image/png" sizes="32x32" href="/assets/favicon/favicon-32x32.png">
<link rel="icon" type="image/png" sizes="16x16" href="/assets/favicon/favicon-16x16.png">

<link rel="manifest" href="/assets/favicon/manifest.json">
<link rel="mask-icon" href="/assets/favicon/safari-pinned-tab.svg" color="#5bbad5">
<link rel="shortcut icon" href="/assets/favicon/favicon.ico">
<meta name="apple-mobile-web-app-title" content="Торгівельна база">
<meta name="application-name" content="Торгівельна база">
<meta name="msapplication-config" content="/assets/favicon/browserconfig.xml">
<meta name="theme-color" content="#ffffff">
<script src=https://yastatic.net/jquery/1.12.4/jquery.min.js></script>
<script>window.jQuery || document.write('<script
src=/assets/jquery.min.js><\/script>')</script>
<script src=/assets/funcs.js?1></script>
<link href="https://fonts.googleapis.com/css?family=Roboto+Condensed:400&amp;subs
et=cyrillic" rel="stylesheet"/>
<link rel="stylesheet" href="/assets/base.css?13"/>
<link

<link rel="stylesheet" href="/assets/fontAwesome/font- awesome.min.css"/>
<!--[page_head]-->
</head>
<body>
<div class="width-wrapper">
<header class="cf">
<div class="header_top cf">
<div class="header_menu">
<a href="/config/">Адміністрування</a>    
<a href="/passchange/">Змінити
пароль</a>    



база</div> [org_name]--></div>

-[user_post]-->)</div>
```

```
<a href="/exit/">Вихід</a>
</div>
<div class="header_title">
<div class="header_title_is">Торгівельна


<div class="header_title_org"><!--


</div>
<div class="header_user"><!--[user_name]--> (<!-


<div class="header_info">
Продажі: <!--[page_add_orderCount]-->

    Завдання: <!--[page_add_taskCount]-->
</div>
</div>
<ul class="header_pages">
<li><a href="/"><div>Головна</div></a></li><li><a
href="/orders/"><div>Замовлення</div></ a></li><li><a
href="/clients/"><div>Клієнти</div></a></li><li><a href="/providers/"><div
>Постачальники</div></a></li><li><a href="/goods/"><div>Товари</div></a></li><li><a
href= "/tasks/"><div>Завдання</div></a><li><a
href="/reports/"><div>Статистика</div></a></li></li >
</ul>
</header>
<h1><!--[page_title]--></h1>
<div class="content cf"><!--[page_content]--></div>

<footer class="cf"><a href="/about/">Система управління торговою базою v.<!--
[system_version]--> від <!--[system_version_date]--
></a></footer>
</div>
<div id="modalNotice"></div>


<link href="https://fonts.googleapis.com/css?family=Open+Sans:400,600,700&amp;sub
set=cyrillic" rel="stylesheet"/>
<script defer async src=/assets/base.js?5></script>
<script defer async src=/assets/remPolyfill.js></script>
<!--[page_foot]-->
<!--[if lt IE 9]><script src=/assets/html5shiv.min.js></script><script src=/assets/css3-
mediaqueries.js></script><![endif] -->
</body>
</html>
```

«./components/functions.php» - файл, що містить користувацькі PHP-функції.

```
<?php
/* ######################################## Вставка даних у шаблон */
```

```
function data2template($template, $patterns = []) {
/* Зчитуємо із запитаного шаблону html-код */
$template = file_get_contents('templates/'.$template.'.html');
/* Якщо отримано масив даних для вставки в шаблон, то запускаємо цикл */
if(is_array($patterns) and $patterns != []) { foreach($patterns as $key => $value) {

/* Вставляємо в шаблон дані за ключовими словами, заздалегідь прописаними в
шаблоні */
$template = str_replace('<!--['.$key.']-->',$value,$template);
}
}
/* trim - очищає початок і кінець змінної від зайвих прогалин */ return trim($template);
}


/* ###################################### Перетворення
unix_timestamp у зрозумілий людині формат */ function time2date($timestamp, $variant) {
global $isys_lang;
$month = & $isys_lang['time2date']['month'];
$month2 = & $isys_lang['time2date']['month2'];
$week = & $isys_lang['time2date']['week'];


switch($variant) {
$d=date("j", $timestamp); $m=$month[date('n',$timestamp)];
$time=date("H:i", $timestamp); return $d.' '.$m.' у '.$time; break;
//1 липня 2014
case 2:
$d=date('j',$timestamp); $m=$month[date('n',$timestamp)];
$y=date('Y',$timestamp);return $d.' '.$m.' '.$y; break;

case 3:
$d=date('j',$timestamp); $m=$month[date('n',$timestamp)];
$y=date('Y',$timestamp); $time=date('H:i', $timestamp); return $d.' '.$m.' '.$y.' у '.$time;
break;
//1 липня case 4:
$d=date('j',$timestamp); $m=$month[date('n',$timestamp)];
return $d.' '.$ m;
break;
case 5:
$d=date('j',$timestamp); $m=$month[date('n',$timestamp)];
$w=$week[date('N',$timestamp)]; return $w.', '.$d.' '.$ m; break;
case 6:
$m=$month2[date('n',$timestamp)]; $y=date('Y',$timestamp);
return $m.' '.$y;
break;
//23:55
case 7:
return date('H:i',$timestamp);
break;
case 8:
```

```
return date('d.m.Y',$timestamp);
break;
```

"./assets/base.css" - файл містить CSS правила.

```
@charset "utf-8";
.cf:before,.cf:after {content:" ";display:table;}.cf:after {clear:both;}.cf {*zoom:1;}


html {font:normal normal 16px 'Open Sans',sans-serif;} body {background:#f0f0f0;}
.clear{clear:both;}
a {transition:all linear 0.3s;color:#840000;} a:hover {color:#E42106 !important;}
blockquote {font:italic normal 0.9rem 'Verdana',sans-serif;color:#333;}
.bold {font-weight:600;color:#222;}
.underline {text-decoration:underline;}
.hidden {display:none !important;}


input.placeholder,textarea.placeholder,select.placeholder {color:#808080
!important;}
::-webkit-input-placeholder{color:#808080;}::-moz-placeholder{color:#808080;}:- moz-
placeholder{color:#808080;}:-ms-input-placeholder{color: #808080;} input[placeholder]{text-
overflow:ellipsis;}input::-moz-placeholder{text- overflow:ellipsis;}input:-moz-placeholder{text-
overflow:ellipsis;}input:-ms- input-placeholder {text-overflow:ellipsis;}
:focus::-webkit-input-placeholder{color:#969696}:focus::-moz-
placeholder{color:#969696}:focus:-moz-placeholder{color:#969696}:focus:-ms- input -
placeholder{color:#969696;}

ul.defUl-bull li {list-style:disc outside;} ul.defUl-num li {list-style:decimal outside;} ul.defUl-
square li {list-style:square outside;} ul.defUl-circle li {list-style:circle outside;}
ul.defUl-bull,ul.defUl-num,ul.defUl-square,ul.defUl-circle {margin-bottom:10px;}

ul.defUl-bull li,ul.defUl-num li,ul.defUl-square li,ul.defUl-circle li {margin-left:18px;}

.defForm {width:500px;box-sizing:border-box;border: solid 1px #e6e6e6;padding:10px 20px
0;margin-bottom:10px;}
.defFormSave {float:right;margin: 10px 0px -15px 0;background: #49CC3D;border:
none;padding: 5px 20px;font-weight: 600;font-size: 0.95rem;transition: 0.2s fff;}
.defFormSave:hover {cursor:pointer;background: #39BA2D;}
.defInput {width:436px;border:solid 1px #f7f7f7;background-color:#f7f7f7;font- size:
0.95rem;padding: 3px 10px;margin-top:3px;}
.defInput:focus {border-color:#000;background-color:#fff;}
.defSelect {width:458px;border:solid 1px #f7f7f7;background-color:#f7f7f7;font- size:
0.95rem;padding: 3px 10px;margin-top:3px;}
.defSelect:focus {border-color:#000;background-color:#fff;}
.defTextarea {width:458px;border:solid 1px #000;font-size: 0.95rem;padding: 3px
10px;margin-top:3px;}
.defRadio {font-size: 0.95rem;padding: 3px 10px;margin-top:3px;margin- right:2px;}
.defInputWrap {margin-bottom:15px;}
.defInputLabel {font-size:0.9rem;font-weight:600;margin-left:3px;}
```

```
.defButton {margin: 20px 15px 15px 0;background: #fff;border: solid 2px #414aff;padding: 4px 20px;font-weight: 600;font-size:0.95rem;transition:0.2s;}
.defButton:hover {cursor:pointer;background:#414aff;color:#fff;}


.defTable {width:100%;box-sizing:border-box;}
.defTable caption {font-size:1.05rem;font-weight:600;margin:15px 0 10px 0;color:#111;text-align:left;}
.defTable td {padding:2px 10px;border:solid 1px #dadada;font-size:0.95rem;}

.defTable thead td {font-weight:600;border-bottom:solid 1px #b1b1b1;}
.defTable tbody tr:nth-child(2n) {background:#f8f8f8;}
.defTable tbody tr:hover {background:#ffe99b;}


.width-wrapper {width:1200px;margin:0 auto;}


header {height:84px;background-color:#fff;box-shadow:0 0 1px 1px #e6e6e6;margin-top:15px;}
.header_top {height:44px;border-bottom:solid 1px #dad8d8;position:relative;}
.header_title {float:left;padding:0 15px 0;font-weight: 600;margin-top:4px;}
.header_title_is {font-size:0.85rem;}
.header_title_org {font-size:0.8rem;}
.header_user {float:left;margin: 12px 0 0 40px;font-size:0.8rem;}
.header_info {margin:11px 40px 0 40px;font-size:0.9rem;float:left;}
.header_menu {background:#fff;margin:9px 20px 0 0;right:0;position:absolute;transition:all ease-out 120ms;}
.header_menu a {font-size:0.8rem;color:#7A0F04 !important;}


.header_pages {font-size:0.95rem;}
.header_pages a {color: #000 !important;}
.header_pages li.current a {color: #C51B00 !important;}
.header_pages li {display:inline-block;padding:0;margin:0;width:171px;}
.header_pages li div {text-align:center;width:170px;box-sizing:border- box;padding:8px 0 9px;border-right:solid 1px #e6e4e4;}
.header_pages li div:hover {background:#f7f7f7;}
.header_pages li:last-child div {border-right:none;width:174px;}
```

ДОДАТОК Б