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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка компонента сайту для ведення статистики відвідувань

Виконав(ла): студент(ка) 4 курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

		Гумен Д.В.
	(підпис)	(прізвище та ініціали)
Керівник		Гром'як Р.С.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Шимчук Г.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Боднарчук І.О.
	(підпис)	(прізвище та ініціали)
Рецензент		Кареліна О.В.
	(підпис)	(прізвище та ініціали)

Тернопіль
2021

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«25» січня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Гумен Дмитро Васильович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розробка компонента сайту для ведення статистики відвідувань**

Керівник роботи **к.т.н., доц. Гром'як Р.С.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*02*» *березня* 2021 року № *4/7-171*

2. Термін подання студентом завершеної роботи 18 червня 2021 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ 1 Теоретична частина Роль пошукових систем у популярності сайту Основи CGI Практична реалізація 2.1 Аналіз принципів побудови систем аналізу відвідувань сайту 2.2 Принцип створення найпростішого аналізатора відвідувань сайту 2.3 Аналіз технічного завдання 2.4 Розробка бази даних 2.5 Розробка інформаційної системи 2.6 Робота з модулем збору статистики використання ресурсів сайту 3 Безпека життєдіяльності, основи охорони праці 3.1 Загальні вимоги законодавства з охорони праці в галузі інформаційних технологій 3.2 Розрахунок освітленості робочого місця розробника модуля
інформаційної системи для збору статистики про відвідування сайту 3.3 Аналіз небезпеки і шкідливості при розробці модуля інформаційної
системи збору статистики використання ресурсів сайту Висновок Перелік посилань Додатки

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

1. Тема. 2. Функціональна схема системи. 3. Структура модуля збору статистики. 4. Алгоритм роботи модуля збору статистики. 5. Початкова сторінка виводу статистики 6. Сторінка з даними про ОС і браузери. 7. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доц.		

7. Дата видачі завдання 25 січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.01.21-27.01.21	*Виконано*
2.	Підбір джерел по темі роботи	28.01.21 – 01.04.21	*Виконано*
3.	Оформлення першого розділу	15.04.2021	*Виконано*
4.	Оформлення другого розділу	30.04.2021	*Виконано*
5.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи охорони праці»	15.05.2021	*Виконано*
6.	Оформлення кваліфікаційної роботи	07.06.2021	*Виконано*
7.	Перевірка на плагіат	07.06.2021	*Виконано*
8.	Нормоконтроль	15.06.2021	*Виконано*
9.	Попередній захист кваліфікаційної роботи	18.06.2021	*Виконано*
10.	Захист кваліфікаційної роботи	21.06.2021	

Студент _____ Гумен Д.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гром'як Р.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка компонента сайту для ведення статистики відвідувань // Кваліфікаційна робота баклавра // Гумен Дмитро Васильович // Тернопільський національний технічний університет, факультет комп'ютерних інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // – Тернопіль, 2021 // с. – , рис. – , табл. – , слайдів. – , додат. – , бібліогр. – .

Ключові слова: відвідування сайту, модуль, інформаційна система, сценарій, php, статистика, веб-сервер.

Кваліфікаційна робота стосується розробки системи реєстрації інформації про використання ресурсів сайту за допомогою розробленого на мові PHP скрипта.

При роботі над завданням даної роботи були враховані всі фактори для зручної роботи користувача з кінцевим програмним продуктом, а також різноманітні форми звітної інформації.

ANNOTATION

Development of a site component for traffic statistics // Humen Dmytro // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group СНзс-42 // Ternopil, 2021 // p. – , fig. – , references – , posters – , applications – .

Keywords: site visit, module, information system, script, PHP, statistics, web server.

Qualification work concerns the development of a system for registering information on use of site resources using a script developed in the PHP language.

When working on the task of this work were taken into account all the factors for the user's convenience with the final software product, as well as various forms of reporting information.

ЗМІСТ

ВСТУП

На теперішній час ресурси мережі інтернет досить різноманітні та потрібні практично кожному користувачеві. Єдина проблема – пошук потрібної інформації та послуг. На сьогодні тема популярності сайту є досить актуальною. Одним із інструментів для просування сайту, підвищення його популярності є збір статистик про цей сайт, а саме, розробника може цікавити кількість відвідувань, географія користувачів, програмне забезпечення, яким вони користуються, звідки приходять на сайт тощо.

Ресурси Інтернету перетворились на незамінний інструмент для повсякденної роботи людей багатьох професій. За оцінкою експертів об'єм даних, які поширюються каналами Інтернету, подвоюється кожні шість місяців. Природно, що без систем пошуку вони взагалі не були знайдені. Виникла необхідність створення для швидкого і надійного необхідних відомостей.

Найпопулярнішим способом пошуку в Інтернеті є застосування сервісів пошуку(пошукових систем). Це портал, що здійснює пошук та сортування інформації в Інтернеті. Користувачі вибирають пошукову систему на основі декількох основних параметрів. По перше, чи знайшов користувач шукані відомості. В разі неуспіху, як він змінював пошукові запити, щоб отримати позитивний результат пошуку. Чи були знайдені дані найактуальнігшими та як були представлені результати. Важливим є також час виконання пошукового запиту та форма подання результатів, в тому числі і порядок виводу результатів, що задовільняють пошуковим критеріям.

З іншого боку розробники сайтів повинні певним чином оптимізувати розроблений сайт, щоби він був популярним серед користувачів, щоби його індексували пошукові машини. Відповідь на ці питання можна отримати, використавши систему аналізу статистичних даних відвідувань сайту. Саме таку систему, як засіб адміністрування сайту, належить розробити у цьому проекті.

Отже, додатковий модуль, котрий збиратиме у базі даних інформацію про використання ресурсів сайту його відвідувачами, допоможе на основі аналізу

цих даних внести корективи у структуру сайту, його оформлення, його змістовне наповнення та технічні особливості реалізації. До них можна віднести розміри сторінок, використання різноманітних мультимедійних елементів, орієнтація на роздільну здатність монітора відвідувача сайту і т.п. Вбудувавши такий модуль збору статистики у сторінки сайту, така інформація буде збиратись і записуватись у базу даних, а потім – використовуватись для розробки цілеспрямованих заходів розкрутки веб-ресурсу у потрібному напрямку.

1 ТЕОРЕТИЧНА ЧАСТИИНА

1.1 Роль пошукових систем у популярності сайту

Можна створити найкращий і найцікавіший Web-сайт, але якщо ніхто про нього не дізнається, то усі зусилля виявляться даремними. Створити сайт і не "розкрутити" його – це все одно, що зняти кінофільм і не показати його глядачам. Під "розкручуванням" або, як ще говорять, "просуванням" сайту розуміється цілий комплекс різноманітних заходів по збільшенню популярності сайту, його відвідуваності і популярності.

Завдання "розкручування" сайту полягає не лише і не стільки в створенні іміджу компанії, скільки в залученні на сайт відвідувачів. Для цього використовується декілька способів. Перший спосіб називається "пасивною" рекламою. Це та ситуація, коли зацікавлений споживач сам шукає потрібний йому товар. "Пасивна" реклама зводиться до реєстрації в пошукових системах, каталогах і рейтингах, обміну посиланнями. Іноді вона здатна принести достатню кількість відвідувачів.

З іншого боку, якщо ви зацікавлені не у кількості відвідувачів, а в їх здатності скористатися вашими, можливо, платними послугами, то ці відвідувачі представляють для вас особливий інтерес: вони шукали саме вас і цілком можуть витратити на вас свої гроші, якщо вони, звичайно, є.

Другий спосіб називається "активною" рекламою. До неї відносяться усі рекламні ролики, що показуються на телебаченні, рекламні тексти на радіо, а в Інтернеті – банери – картинки, клацання мишею на яких переводить відвідувача на сайт рекламодавця. Саме у цей спосіб реклами вкладається найбільша частина грошей, витрачених на рекламу проекту. Цей спосіб має на увазі пошук споживача товаром, а не товару споживачем, як у разі "пасивної" реклами.

Мережевий аспект цього методу полягає в залученні відвідувача на сайт, далі ж завдання полягає в тому, щоб не дати відвідувачеві піти і змусити його щось купити, якщо сайт комерційний. Створивши свій сайт, ви завжди можете розраховувати на його хоч би часткову окупність. Підкреслимо – завжди. Будь-

який сайт в змозі отримати гроші за рекламу на своїх сторінках. Але не слід думати, що будь-який сайт може заробити багато грошей. Навпаки – тільки деякі сайти здатні повністю окупати витрати на свій зміст. Для того, щоб сайт окупався, обов'язкове виконання декількох умов, одним з яких є хороша відвідуваність. А для хорошої відвідуваності потрібний цікавий зміст і правильне оформлення. Ніяке "розкручування" не гарантує хорошої окупності поганому ресурсу. Неможливо заробити скільки-небудь велику суму, не зробивши якісний ресурс і не вкладаючи в нього свою працю. Окупити сайт можна двома способами. Перший полягає в наданні яких-небудь платних послуг. Для цього не потрібна велика відвідуваність, важлива платоспроможність відвідувачів і їх зацікавленість. Другий спосіб – розміщення реклами – навпроти, вимагає великої відвідуваності. Помістити рекламу на своєму сайті можна, або самостійно знайшовши собі рекламодавця, або скориставшись послугами "спонсорів". Кращий варіант – перший: створити цікавий, відвідуваний ресурс на популярну тематику і знайти собі рекламодавця, готового працювати саме з вами. Це найбільш трудомісткий, та зате найбільший дохід, що приносить, спосіб. Якщо ж ви хочете скористатися менш трудомістким методом, то в Інтернеті ви знайдете безліч посилань на сайти рекламодавців-спонсорів. Природно, цей метод також має на увазі хороше оформлення сторінок і цікаву тематику.

Ймовірно твердження, що нині основним засобом знаходження інформації в Інтернеті являються пошукові машини, є досить достовірним. Можливо, ви погодитеся також і з тим, що найпоширеніша схема роботи по пошуку інформації в Інтернеті – це знаходження за допомогою пошукових машин десятка посилань з наступним розгортанням теми по суміжних посиланнях.

Щодня до пошукових машин звертається декілька десятків мільйонів чоловік, які шукають інформацію, що цікавить їх. Можливо, ця інформація знаходиться на вашому сайті. Декілька російських пошукових машин обробляють в день більше ста тисяч запитів і є одними з самих відвідуваних сайтів в російськомовному Інтернеті.

Природно, дуже важливо, щоб ваш сайт був доступний пошуковим машинам. Для деяких сайтів трафік (потік відвідувачів) від пошукових машин

може досягати сорока і більше відсотків, проте твердо розраховувати на це все ж не доводиться. Причому, на відміну від каталогів і рейтингових систем, при використанні пошукових машин люди потрапляють безпосередньо на матеріал, що цікавить їх, який далеко не завжди є головною сторінкою сайту.

Очевидно, що взаємодія з пошуковими машинами є одним з найефективніших способів збільшення відвідуваності ресурсу. Нехтувати їм ні в якому разі не слідує, більше того, звернете особливу увагу на оптимізацію вашого сайту для пошукових машин. Основа пошукової машини – робот або, як його ще називають, павук – оцінює текст вашої сторінки і визначає його відповідність (релевантність) введеному запиту.

Одним з найважливіших параметрів релевантності документу є частота вжитку ключового слова в тексті. Отже, ваше завдання – знайти ключові слова, в максимальній мірі відповідні тематиці сайту і оптимізувати сторінки саме під ці ключові слова. Ключові слова, тобто ті слова, які найбільш характерні для описуваного предмета, є основним критерієм, який використовується для пошуку різними системами.

Тому перед тим, як подавати запит на включення вашого сайту в базу даних пошукового робота, уважно прочитайте і проаналізуйте текст на головній сторінці сайту. Визначите, яке слово або вираження трапляється на ній найчастіше. Чи характерно це слово для тематики сайту або це розмовний оборот? Як часто зустрічається цей вираз?

Коли ви проаналізуєте текст на сторінці у такий спосіб, у вас утворюється список з декількох ключових слів, які досить точно і яскраво характеризують ваш ресурс. Варто зберегти ці слова в окремому текстовому файлі. Після цього треба переробити текст на сторінці (йдеться тільки про ту сторінку, яку потрібно рекламувати) так, щоб вибрані ключові слова там не лише траплялися, але траплялись регулярно (в середньому не менше 5-10 разів).

У цьому ключі потрібно переробити не лише сам текст, але і назву сторінки, яка відображується в заголовку браузеру, підписи до малюнків і так далі. Очевидно, що оптимізувати сторінку під велику кількість ключових слів

украй складно. Тому постарайтеся обійтися двома-трьома словами. Для цього виберіть декілька "найяскравіших" слів і підлаштовуйте сторінку саме під них.

Внаслідок особливостей пошукових машин має сенс звернути особливу увагу на перший кілобайт тексту сторінки і працювати цілеспрямовано саме з ним. Для того, щоб пошуковий робот міг знайти і зрозуміти ваші ключові слова, треба оформити їх в спеціальну конструкцію, звану мета-тегами. Це необов'язкові атрибути Web-документа, які містять опис сторінки, ключові слова до неї, деяку інформацію про автора, команди, що управляють, для пошукових роботів і зачую службову інформацію, не призначену для усіх відвідувачів. Мета-теги поміщаються на початку HTML-коду між тегами <head> і </head> і виглядають , як в лістингу 1.1.

Лістинг 1.1 – Приклад мета-тегів

```
<head>
<title>заголовок</title>
<meta http-equiv="author" contents" Автор сторінки">
<meta http-equiv= "KEYWORDS" content="Ключові слова через
 кому">
<meta http-equiv="DESCRIPTION" content="Опис сторінки">
</head>
```

Мета-теги необхідно вставити в код кожної сторінки вашого сайту. Тобто кожна сторінка повинна містити свої власні ключові слова і свій власний опис, які їй найбільш відповідають. Це досить велика робота, але необхідна. Від неї, зокрема, залежить кількість відвідувачів, яка прийде на ваш сайт.

Усе сучасні HTML-редактори, незалежно від того, чи працюєте ви безпосередньо з кодом або у візуальному редакторі, дозволяють вставляти мета-теги в автоматичному режимі. Досить вказати у відповідному майстрові або діалозі опис сторінки разом з її ключовими словами, і мета-теги будуть додані. Повторимо, що кожна сторінка вашого сайту, окрім власної назви, матиме і власні, найбільш їй відповідні, ключові слова, і власний опис.

Звичайно, частина ключових слів, загальних для усього ресурсу, має бути присутньою в мета-тегах на усіх сторінках сайту, але там можуть знаходитися також і свої, унікальні для кожної сторінки слова. Це і природно, якщо враховувати, що кожній сторінці вашого сайту властиво своє, унікальний зміст і саме його, можливо, і потрібно буде знайти користувачеві через пошукову машину. Тому унікальні особливості кожної сторінки корисно відбити в мета-тегах.

Дуже важливо, щоб ключові слова були не просто описом теми, але і відповідали основному тексту сторінки, повторювалися у ньому. Інакше усі ваші зусилля пропадуть дарма, тому що сторінка, яка містить шукані ключові слова тільки в мета-тегах, швидше за все буде визнана роботом нерелевантною або слабо релевантною, що загалом одне і те ж. Тому потурбуйтеся про те, щоб включити в основний текст вашої сторінки ключові слова, причому чим частіше вони там вживатимуться, тим краще.

Не менш важливий опис титульної сторінки, оскільки він повинен відображувати увесь сайт: усі його особливості, напрями роботи, цікаві місця – або, простіше кажучи, його унікальність і значущість. Написавши хороший опис титульної сторінки, опис кожної окремої сторінки скласти дещо простіше – треба узяти загальний опис і звузити його до теми конкретної сторінки.

Вдалим рішенням можна вважати включення в мета-теги внутрішніх сторінок опису і ключових слів, складених для усього сайту. Довжина кожного поля content в мета-тегах обмежена 1024 символами, включаючи пропуски і розділові знаки. Цього більш ніж достатньо, щоб написати продуманий, розгорнутий опис.

Проте слід пам'ятати, що мета-теги включаються в код сторінки і збільшення їх довжини веде до збільшення об'єму сторінки, а, отже, збільшує час завантаження. Тому не слід захоплюватися. До речі, в цілях економії місця можна, і це має відомий сенс, поміщати в мета-теги внутрішніх сторінок скорочену версію опису усього ресурсу.

Величезне значення має також правильно складений заголовок веб-сторінки. Саме він відображується в першому рядку вікна браузеру, а потім – в закладках користувачів. Його показуватимуть пошукові машини в результатах пошуку, на нього в першу чергу звернуть увагу оглядачі мережевих ресурсів. Тому заголовок слід вибирати дуже ретельно і уважно. Заголовок поміщається в спеціальній конструкції між відкриваючим тегом <title> і закриваючим тегом </title> і може містити 256 символів.

На самій сторінці заголовок не відображується, якщо тільки ви його не вставите окремо. Не дивлячись на те, що у розпорядженні веб-дизайнера є 256 символів, робити назву завдовжки більше ніж 60-80 знаків неварто, оскільки ширина вікна браузерів і закладки не вміщують довгих рядків, обрізуючи їх. А заголовок, який "обрізаний", виглядає не самим кращим чином. Назву сторінки слід починати зі значущого слова: назви продукту, торговельної марки, послуги або тематики.

Зверніть увагу на те, щоб назва внутрішніх сторінок містила також і назву самого сайту, підкресливши таким чином їх приналежність. Це стає особливо важливим, коли сторінок багато, а розкид тем великий. Річ у тому, що з пошукових машин відвідувач часто потрапляє на внутрішні сторінки сайту, минувши титульну, а, отже, він не знатиме ні назви вузла, ні його адреси. Після того, як сторінки забезпечені мета-тегами з описами і ключовими словами, вони готові до реєстрації в пошукових машинах. Пошукова машина складається з двох частин: робота-павука і індексу.

Павук постійно обстежує Інтернет і усі зустрінуті ресурси переглядає і заносить в спеціальну базу даних – індекс. Таким чином, пошук за запитом виконується не по реальних сайтах, а по записах в індексі.

Теоретично, якщо існує сайт, то коли-небудь павук його знайде. Але коли це станеться – не знає ніхто. Тому існує можливість запропонувати пошуковому роботові проіндексувати розроблений сайт, повідомивши йому адресу. Заповнюючи спеціальну форму, ви записуєтеся в "чергу" на індексацію і можете

сподіватися, що через декілька днів ваша адреса вже буде в базі даних пошукової машини.

На яких пошукових машинах і як слід реєструватися? Багато хто вважає, що треба послати свої координати в бази даних максимальної кількості доступних машин. На сьогодні існує немало пошукових машин в мережі Інтернет. Але майже всі (до 98% запитів) обробляється дуже малою їх кількістю. Інші або не розкручені до загальновідомого стану, або локалізовані по мові, а, отже, позбавлені сенсу, якщо, звичайно, вони не написані українською чи англійською мовами. У результаті робимо висновок, що реальною цінністю є не більше десятка пошукових роботів. Серед українських заслуговує на увагу один пошуковий ресурс – meta.ua. Його використання має зміст, коли потрібно проводити пошук по українських сайтах.

Враховуючи той факт, що в Інтернеті усе дуже швидко міняється, слід мати на увазі, що вказані нижче адреси з часом можуть стати недоступними. Серед зарубіжних пошукових машин практично усі провідні пошукові роботи навчилися індексувати документи різними мовами. Такі гіганти, як AltaVista, Infoseek, HotBot, GoTo.com. Google і інші, пропускають через себе декілька мільйонів користувачів щоденно, тобто значно більше, ніж українські.

Тому, не дивлячись на те, що велика частина користувачів цих машин володіє тільки англійською мовою, реєструватися в них все одно треба, оскільки це дає реальних відвідувачів. Більше того, при вдалому збігу обставин ви можете отримати до декількох тисяч чоловік в день від будь-якої з цих машин. Але і удача для цього має бути винятковою. Перш ніж реєструватися в англомовній пошуковій машині, перекладіть опис і ключові слова англійською мовою, можливо, продублюйте їх в мета-тегах. У формі реєстрації на сайті пошукової машини краще вводити або англійський текст, або разом з локалізованим варіантом переклад на англійську.

Пошукова машина AltaVista (http://www.altavista.com/), мабуть, найбільш відома і найстаріша з нинішніх лідерів пошукових машин, хоча і не найзручніша. Здатна шукати майже на тридцяти мовах і перекладати з однієї на іншу "на

льоту", правда, не на всі. Дуже швидко індексує сторінки, зазвичай протягом двох-трьох днів. Форма для реєстрації знаходиться за адресою: http://www.altavista.com/addurl.

Пошукова машина Infoseek (http://www.infoseek.com/) – ймовірно, найбільша і найзручніша на сьогодні пошукова машина. Так само як і AltaVista здатна шукати на багатьох мовах, і навіть по базах MP3-файлів і картинках (по полю alt). Дуже добре шукає по запитах з цілих фраз, точно визначає при цьому релевантні документи. Дуже швидко індексує сторінки. Вважається на даний момент одній з найзручніших і "акуратніших" пошукових машин. Пошукова машина HotBot (http://www.hotbot.com/) уміє шукати на декількох мовах, у тому числі і на росіянинові, не шукає словоформи, повільно реєструє. Ефективність цієї машини непередбачувана, оскільки однакова, здавалося б, початкова інформація для Infoseek і HotBot означає принципово різні речі. Положення лідера говорить про те, що реєструвати свій сайт в базі цієї машини все ж треба, тим більше що база HotBot використовується порталом Infoseek і декількома популярними каталогами. Форма для реєстрації знаходиться за адресою: http://hotbot.lycos.com/addurl.asp.

На сьогоднішній день беззаперечним світовим лідером серед пошукових систем є Google. Ця система індексує контент в Інтернеті, має каталог, окремо може здійснювати пошук зображень тощо. Крім того, користувачам надається цілий ряд безкоштовних сервісів, котрі не мають безпосереднього відношення до пошуку даних в мережі Інтернет.

1.2 Основи CGI

Даний розділ орписує програмні інструменти відправки інфомації із сторінок сайту на сервер та обробку цих даних з використанням мови PHP. На цьому прикладі можна детально розглянути основні принципи клієнт-серверних технологій. Проаналізуємо також, як працює HTML-форма і як можна

відправити з неї на сервер введені дані. У зв'язку з цим розглянемо методи Post та Get.

PHP – це інтерпретована мова програмування з виконання скриптів на серверній стороні. Опишемо трохи детальніше про поняття сервера.

Сервер – це програмний процес, що виконується з метою обробки запиів від інших програм, котрі називаються клієнтами. Очевидно, що така програма встановлена на певному комп'ютері, який, як правило, також називають сервером.

Взаємодія між клієнтом і сервером починається ініціює клієнт, який запитує тип послуги в сервера, встановлює сеанс підключення, отримує вчідповідні до запиту результати і надсилає серверу повідомлення про те, що сеанс зв'язку можна завершити. Типово один сервер обслуговує запити більше одного клієнта. Тому комп'ютер-сервер повинен володіти достітніми обсилювальними потужностями та, крім того, гарантувати необхідний рівень захисту даних. Можливі такожситуації, коли серверне та програмне забезпечення фізично встановлені та киконуються не тільки в мережі, але і не окремому одному і тому ж комп'ютерові.

Опишемо основні типи серверів, як апаратно-програмних систем.

- Відеосервер апаратно сконфігурований для обробки мультимедія та володіти достатнім обсягом оперативної пам'яті і відеоадаптером.
- Пошуковий сервервиконує запити для знаходження даних в мережі Internet.
- Поштовий сервер виконує роль електронного "поштового віддліення" і здійснює пересилку повідомлень електронної пошти.
- Сервер WWW (веб-сервер) опрацьову HTTP-запити.
- Сервер баз даних надає доступ до баз даних на основі запитів мовою для відповідної моделі даних.
- Сервер захисту реалізує додаткові механізми автентифікації та авторизації.

– Сервер застосувань призначений для реалізації бізнес-логіки на серверній стороні інформаційної системи.

– Сервер віддаленого доступу надає доступ клієнтам до інформації по каналах мережі на основі протоколів віддаленого доступу.

– Файловий сервер реалізує файлове сховище на основі централізованої чи розподіленої системи зберігання даних.

Відповідно до теми кваліфікаційної роботи нас цікавить сервер веб. По-перше, такий сервер містить колекцію інформаційних ресурсів на мові розмітки. Доступ до ресурсів користувачі отримують через протокол HTTP. Клієнтським програмни забезпеченням в цьому разі є інтернет-браузер. В процесі виконання запитів сервер може надсилати запити іншими серверам.

За останні роки веб-серверами, що набули найбільшого поширення, є два сервери. Перший з них – це Apache, сервер, що розповсюджується на умовах відкри тої ліцензії GPL та координується Apache Software Foundation – організації, що працює на засадах співпраці вільних розробників. Другим сервером є програмний продукт, що по факту складається з багатьох застосунків під загальною назвою Internet Information Server (IIS) від Microsoft.

Таблиця 1.1 – Розробники web-серверів

Сервер	Лютий 2014	%	Березень 2014	%	Зміна
Microsoft	9848991	20,88	10199765	22,02	1,14
Apache	31703 990	67,25	34289582	67,21	-0,04
SUNONE	1657295	3,56	1652585	3,48	-0,08

Крім того, можна згадати менш розповсюджені аналогічні продукти, наприклад, SunOne від Sun Microsystems. У таблиці 1.1 показана досить давня статистика по використанню різних веб-серверів.

Як видно, навіть на досить давній час сервер Apache був лідером. Зберіг він ці позиції і нині. А на другому місці по поширенню перебуває IIS від

Microsoft. А тепер розглянемо особливості протоколу HTTP для реалізації нашого сайту.

1.2.1 Способи передачі даних на сервер через протокол HTTP

Протокол передачі гіпертексту (HyperText Transfer Protocol, HTTP) – протокол прикладного рівня, розроблений для здійсненя обміну гіпертекстовими документами в Internet.

HTTP володіє набором методів для побудови запитів, що відправляються на веб-сервер. Для вказівки точно ресурсу, до якого повинен бути застосований конкретнрий метод, застосовується URI – Universal Resource Identifier у вигляді адреси, за якою можна отримати ресурсс – Universal Resource Locator, URL. Повідомлення в мкежах протоколу HTTP передаються у вигляді тексту, що містить структуровані дані.

HTTP служить для комунікацій між програмами користувача а також між проміжними програманими шлюзами, через які надається доступ до інших сервісів та Internet-протоколів: SMTP (електронної пошти), FTP (протокол передачі файлів), тощо.

Протокол працює по ппринципу запит/відповідь. Програма –клієнт надсилає запит на сервер, який дає відповідь.

У запиті вказані настіпні дані:

- метод доступу;
- адреса URI;
- версія протоколу;
- саме повідомлення з даними про клієнта, його ОС, браузер. Наявність змістовної частини не є обов'язковою для всіх питів запитів.

Відповідь сервера містить:

- стрічка стану з версією протоколу та код повернення від сервера (успіх – 200 або помилка 400, 500);
- повідомлення, в яке входить дані від сервера і тіло повідомлення.

На практиці з'єднання, як правило, ініціює клієнт, а сервер після відправки відповіді закриває з'єднання без "запам'ятовування" будь-якої інформації про сеанс зв'язку, що відбувся. Це потребує вирішення ждля контролю сесій на клієнській стороні.

Розглянемо детальніше, як запити відправляються на сервер. Клієнт посилає серверу запит у повній або скороченій формі. Простий запит складається з полів методу доступу та адреси. Це можна записати наступним чином:

```
<Простий-Запит> := <Метод><символ пробіл><Затребуваний-URI><символ
нового рядка>
```

В якості методу можуть бути будь-який з наступних: GET, POST, HEAD, PUT, DELETE. В якості URI використовується найчастіше URL ресурс потрібного документу.

Прикладом простого запиту є наступний. Варто звернути увагу, що всі запити формує браузер на основі даних, введених у стрічку адреси та, при наявності, полів форми: `GET http://sitesample.info/`

GET – це метод, який має бути використаний до потрібного ресурсу, а http://sitesample.info/ – це URL-адреас ресурсу.

Повний запит містить заголовок, загальний заголовок чи заголовок змісту і, при потребі, тіло запиту. Загальний вид повного запиту можна побачити на лістингу 1.2.

Лістинг 1.2 –Вид повного запиту протоколу HTTP

```
<Повний запит> := <Рядок Стану>
        (<Загальний заголовок>|
        <Заголовок запиту>|
        <Заголовок змісту>)
  <символ нового рядка>
  [<зміст запиту>]
```

Квадратні дужки містять опційні елементи заголовка, через вертикальну риску вказані альтернативні варіанти. <Рядок стану> фактично є текстовою стрічкою, котра складається з декількох частин, а саме, з вказання методу запиту та URI ресурс. Також стандарт протоколу передбачає вказання у цій стрічні використовувану версію протоколу. Наприклад, для звертання до сторонньої програми обробки можливо скасти і виконати наступний запит:

```
POST http://phpbook.info/cgi-bin/test HTTP/1.0
```

В даному випадку використовується протокол HTTP версії 1.0 та метод передачі даних POST. Далі розглянемо методи відправки запитів до веб-сервера по протоколу HTTP.

1.2.2 Методи передачі даних на сервер

Специфікація HTTP визначає ці методи - GET, POST, PUT, HEAD, DELETE, OPTIONS та кілька інших.

Простий HTTP-запит на отримання повідомлень із сервера може виглядати так:

```
GET / повідомлення HTTP / 1.1
```

Перша частина, GET - це метод HTTP.

Друга частина - це URI.

Третя частина HTTP / 1.1 - версія HTTP.

Метод HTTP вказує дію, яку ми хочемо виконати, а URI - ресурс, над яким ми хочемо виконати дію.

Існу умовний метод GET. Він повідомляє серверу що на запит потрібно відповісти, якщо виконана умова, розміщена у спеціальному полі заголовка запиту.

Запити GET використовуються для отримання інформації про ресурс, вказаний URI. GET - безпечний метод. Це означає, що запит GET не повинен призводити до змін у стані сервера. Це не повинно спричиняти створення, оновлення або видалення будь-яких даних програми. Його слід використовувати лише для дій "лише для читання".

POST

Запити POST використовуються для передачі даних на сервер. Запити POST можуть містити корисне навантаження даних, яке буде подано на сервер. Дія, яку виконує сервер, визначається кодом сервера. Запити POST можуть бути використані для створення нового ресурсу або для подання даних на обробку.

PUT

Запити PUT використовуються для збереження об’єкта в місці, вказаному в URI запиту. Якщо отримано та виконано два або більше однакових запитів PUT, результат повинен бути еквівалентним виконанню такого запиту лише один раз. Щоб провести аналогію, a = 5 є ідемпотентною операцією, оскільки запуск її один раз або кілька разів призводить до значення істоти 5. На відміну від цього, a = a + 1 не є ідемпотентною операцією, оскільки значення змін на основі скільки разів ми виконуємо операцію.

DELETE

Запити DELETE використовуються для видалення об’єкта в місці, вказаному в URI запиту.

HEAD

Запити HEAD використовуються для отримання лише заголовків, які були б присутні у відповіді на еквівалентний виклик GET. Він може бути

використаний просто для перевірки того, чи існує ресурс, чи для отримання заголовка Content-Length перед тим, як вирішити, завантажувати великий файл чи ні. Ви також можете перевірити заголовок останньої модифікації, щоб побачити, чи файл було змінено з моменту останнього його отримання. Метод HEAD безпечний. Усі безпечні методи також неімпотентні, оскільки одноразове нічого не робить того ж ефекту, що й багаторазове.

Для передачі даних на сервер в мові HTML служать форми, як складові сторінок сайту. Вони призначені для введення інформації користувачем. У них ви можете ввести текст або вибрати відповідні параметри зі списку. Дані, записані у формі, надсилаються для обробки в спеціальну програму (наприклад, скрипт у PHP) на сервері. Залежно від даних, які вводить користувач, ця програма може створювати різні веб-сторінки, звертатись до бази даних, викликати потрібні програми тощо.

Для створення форми в мові HTML служить тег FORM. Всередині ньогоьо є можна розмістити довільну кількість елементів типу INPUT. Атрибути action і method тега FORM вказують на програму для обробки даних з форми, та метод запиту. Команда INPUT визначає тип і додаткові характеристики даних, що передаються. Відправка даних з форми починається після активації користувачем кнопки типу SUBMIT. Розглянемо різницю між методами GET і POST при передачі даних на сервер.

Метод GET.

Коли дані з форми надсилаються на сервер за допомогою методу GET, то вміст форми додається до URL у вигляді пар вигляду ім'я=значення. Цей список відділяється від основної частини адреси знаком амперсанда &:

```
action?name1=value1&name2=value2&name3=value3
```

Тут action – це URL-адреса, що вказує на програму, яка буде обробляти форму, задається за допомогою тегу action. Ідентифікатори name1, name2, name3 отримують значення з відповідних елементів форми, а змінні з іменами value1,

value2, value3 будуть містити значенням відповідних елементів. Спеціальні символи, разом з "=" і "&", у іменах та значеннях параметрів будуть вилучені. Тому не слід застосовувати у назвах чи значеннях ці символи і символи кирилиці. При потребі ці символи можуть бути передані за допомогою спеціальних кодів. Наприклад, символ $ заміниться на %24.

Для полів введення тексту і пароля значенням буде те, що введе користувач. У елементах checkbox і radiobutton значення value задається у атрибуті VALUE, якщо кнопка відмічена. Невідмічені кнопки ігноруються.

Для передачі даних методом GET можна всього лише додати в рядок URL необхідні змінні разом з їх значеннями, а не створювати форму для цього. У зв'язку з цим будь-хто може підставити свої значення параметрів і спотворити запит. Тому не варто використовувати цей метод передачі паролів та іншої інформації, що потребує захисту. Крім того, не слід звертатись до методу GET, щоб передати дані, які користувач не повинен змінювати.

Метод POST.

Вміст всіх полів форми кодується по аналогії до методу GET, але замість додавання вміст запиту до рядка URL відправляється як частина операції POST. Якщо наявний атрибут ACTION, то значення URL, вказане в ньому, задає , куди напправляти цей блок. Такий метод краще використовувати для передачі великих блоків даних. Довжина цього блоку даних (файлу, що передається), міститься у змінній оточення CONTENT_LENGTH, а тип цих даних може букти прочитаний із змінної CONTENT_TYPE.

Передача даних з застосуванням методу POST реалізується лише через HTML-форму, оскільки всі дані передаються в тілі запиту, а не в заголовку. Для запиту POST користувачу дані, що передаються, недоступні. Тому заміна значень з полів форми є проблемною ізловмиснику потрібно шукати інші шляхи підміни даних..

Отже перевагою POST запитів в порівнянні з методом GET є їх захищеність і функціональність. Проте не слід повністю вважати, що метод POST абсолютно самодостатній з точки зору безпеки, бо дані в POST також

можна підмінити через створення власної HTML-форми і відпавки даних з неї на потрібну адресу.

При відправці даних на сервер незалежно від методу у змінних оточення міститься інша інформація. Перерахуємо даякі змінні оточення:

- REMOTE_ADDR – IP-адреса клієнта;
- REMOTE_HOST – ім'я клієнтського хоста;
- HTTP_REFERER – адреса сторінки з посиланням на поточний скрипт;
- REQUEST_METHOD – клієнтський метод для поточного запиту;
- QUERY_STRING – інформація в URL;
- SCRIPT_NAME –шлях до програми, яка буде опрацьовувати дану форму;
- HTTP_USER_AGENT – інформація про браузер клієнта.

2 ПРАКТИЧНА РЕАЛІЗАЦІЯ

2.1 Аналіз принципів побудови систем аналізу відвідувань сайту

Завдання цього розділу розповісти про різні типи лічильників відвідувань і про основні принципи їх роботи. А також допомогти при виборі відповідного типу лічильника.

Аналіз статистики сайту проводиться на основі даних про відвідувачів сайту. Дані про відвідувачів збираються веб-сервером (і потім записуються в лог-файли) або лічильниками відвідувань (і потім записуються в бази даних). У рамках цього матеріалу розглянемо принципи роботи різних типів лічильників.

2.1.1 Загальний принцип побудови лічильників відвідувань веб-сторінок

Принцип роботи усіх лічильників відвідувань полягає у виконанні зовнішньої програми при завантаженні сторінок сайту. При завантаженні лічильника виконується зовнішня програма, при цьому їй передаються так звані змінні оточення. У цих змінних зберігається уся базова інформація про поточного відвідувача сайту, у тому числі:

1. IP-адреса відвідувача (REMOTE_ADDR);
2. Браузер відвідувача (HTTP_USER_AGENT);
3. Адреса сторінки, звідки прийшов відвідувач (HTTP_REFERER);
4. Адреса сторінки, куди прийшов (REQUEST_URI);
5. Параметри виклику сторінки (QUERY_STRING).

Параметри виклику або QUERY_STRING передаються через знак питання ? після адреси сторінки і розділяються знаком амперсанда &, наприклад:

```
//cgi/test.php?i=34433&resolution=2048&color=64
```

Виклик зовнішньої програми може бути здійснений різними способами, найпоширеніший спосіб – використання картинки.

2.1.2 Лічильник відвідувань з використанням графіки

Сторінки веб-сайту зазвичай складаються з тексту і графіки. Текст відформатовано за допомогою спеціальних тегів, а графіка є сукупністю картинок, розміщених в правильних місцях.

Картинка вставляється в сторінку, як на лістингу 2.1:

Лістинг 2.1 – Вставка зображення-лічильника у HTML-код

```
... text of html page..
<img src=http://www.myserver.com/img/picture.gif width=720 height=48>
... text of html page..
```

Браузер, відображаючи сторінку на екрані, формує запит за адресою http://www.myserver.com/img/picture.gif і у відповідь сервер посилає браузеру файл картинки.

Принцип роботи лічильника-картинки заснований на припущенні, що при перегляді сайту браузер користувача автоматично підвантажує усі картинки і, відповідно, завантажує картинку лічильника. Тут нас підстерігають дві основні проблеми:

- деякі користувачі відключають завантаження картинок;
- роботи взагалі не вантажать картинок при скануванні сайтів.

Таким чином, ми не зможемо порахувати користувачів з відключеним завантаженням картинок і не зможемо контролювати діяльність роботів на сайті.

Методика підрахунку статистики на основі лічильника-картинки.

Замість посилання на картинку ми вставляємо виклик зовнішньої програми, яка "прикидається" картинкою:

```
<img src=http://www.myserver.com/counter.php height=0 width=0>
```

Програма counter.php формує файл картинки, який віддається браузеру. Таким чином, для браузера виклик програми виглядає як завантаження звичайної картинки.

Картинка, яку формує програма, може бути будь-чим. Наприклад, прозорий GIF розміром 1x1, або може бути картинка лічильника розміром 88x31 з числами відвідувань сайту (всього переглядів сайту, сьогодні переглядів, сьогодні користувачів), або будь-яка інша.

Аналізуючи змінні оточення, програма отримує IP-адресу відвідувача і дані про браузер і записує цю інформацію в базу даних для наступного аналізу. Проте для повноцінної статистики потрібна додаткова інформація. Для передачі додаткової інформації програмі можна використовувати рядок параметрів виклику. Наприклад, передача роздільної здатності екрану користувача спрощено може виглядати так:

```
<img src=http://www.myserver.com/counter.php?screen=2048 height=0 width=0>
```

Додаткова інформація про користувача виходить за допомогою java-скрипта: реферер, роздільна здатність екрану, глибина кольору, випадкове число, інформація про встановлені cookie і так далі. Саме тому при використанні лічильника-картинки доводиться вставляти на сторінки сайту досить значні коди лічильників на java-скрипті.

За допомогою лічильника-картинки ми можемо дуже добре збирати інформацію про більшість користувачів, проте "за бортом" залишаються пошукові роботи і користувачі з відключеними картинками.

2.1.3 Лічильник відвідувань у вигляді вбудованого модуля збору статистики відвідувань сайту

Більшість сайтів зараз динамічні – їх сторінки генеруються на льоту по запиту відвідувача сайту. Сторінки динамічних сайтів написані з використанням

мови програмування, в основному це PHP, ASP, JSP. Тобто сторінки сайту самі є програмами і мають свої змінні оточення. Ми можемо написати код на мові сайту, наприклад PHP, який збиратиме інформацію про відвідувача і складатиме її в базу даних, але не виводити в результаті роботи ніякої видимої інформації. Для зручності вставки такого коду в сторінки, його зазвичай оформляють окремим файлом, а потім вставляють в сторінки сайту.

Наприклад, код лічильника програмної вставки (PHP – Include) у CNStats виглядає приблизно, як на лістингу 2.2:

Лістинг 2.2 – Використання CNStats

```
... .. php - code ..
include " /usr/www/users/www.myserver.com/cnstats/cnt.php";
... .. php - code ..
```

Оскільки код лічильника включений в код сторінки, то є гарантія, що усі відвідувачі сайту будуть пораховані навіть не помітивши цього.

Проте і при використанні програмної вставки, є певні недоліки:

- інформацію про відвідувача можна взяти тільки із змінних оточення;
- складність визначення унікальності відвідувача.

2.1.4 Комбінований лічильник відвідування у вигляді вбудованого модуля з використанням графіки

Ідея наступна – в сторінки динамічного сайту вставляється код програмного модуля, який при виконанні виводитиме код лічильника відвідування картинки. Що ж виходить?

У момент генерації сторінки викликається код програмної вставки і отримана із змінних оточення інформація записуються в базу даних.

В результаті роботи програмної вставки на генерованій сторінці з'являється java-скрипт код лічильника-картинки.

При перегляді сторінки браузером спрацьовує java-скрипт і викликається лічильник-картинка з додатково зібраною інформацією.

Додаткова інформація зібрана java-скриптом про цього ж відвідувача дописується в базу даних.

Система збору статистики дещо ускладнюється (і не завжди комбінований тип лічильника підходить), зате комбінований спосіб – єдиний варіант отримати максимально повну інформацію про усіх відвідувачів сайту.

На цей момент викладено інформацію про основні типи лічильників відвідувань. На її основі приймемо рішення про побудову нашого лічильника відвідувань та про спосіб збору статистики сайту.

При цьому врахуємо ще такі моменти:

– варто використовувати універсальні програмні продукти, які підтримують різні типи лічильників, – це дасть можливість вибору;

– по можливості варто використовувати комбінований лічильник – це останнє досягнення в плані збору статистики.

2.2 Принцип створення найпростішого аналізатора відвідувань сайту

Статистичні відомості про відвідувачів сайту приносять багато користі. За статистикою можна підігнати дизайн сайту відповідно до роздільної здатності моніторів більшості відвідувачів, підігнати дизайн до браузера, який використовує велика частина відвідувачів, також відомості, хто заглядає на сайт, з під яких ОС, а може це пошуковий робот яндекса або гуглу? Хоча деякі системи стеження за відвідувачами бувають надзвичайно складними, але за допомогою досить простої системи можна отримати цікаві відомості про відвідувачів сайту. Розглянемо на прикладі, як зробити на вигляд простий журнал відвідувань сайту за допомогою PHP і cookies (MySQL не вимагається). До того ж приклад можна легко розширити.

Для того, що б система працювала, скрипт статистики треба вбудувати в кожну сторінку, або в ті сторінки, статистику відвідувань яких потрібно побачити. Наш скрипт записуватиме наступні дані:

Браузер + ОС (HTTP_USER_AGENT)

IP-адреса (REMOTE_ADDR)

Хост (REMOTE_HOST)

Сторінку-реферер (HTTP_REFERER)

Час візиту (date("d.m.Y H : i: s"))

Запрошувана адреса (REQUEST_URI)

Навіть ці дані будуть дуже цікаві веб-розробникам. Отже, почнемо. Скрипт називатиметься sniffer.php. Приведемо текст усього скрипта і доповнимо його коментарями (лістинг 2.3):

Лістинг 2.3 – Каркас скрипта-сніфера

```
<?php
//sniffer.php
//захист від безпосереднього запуску
//скрипта стороннім користувачем
if (eregi("sniffer.php",$PHP_SELF)){
        Header("Location: index.php");
        die();
}
extract($HTTP_GET_VARS);
extract($HTTP_POST_VARS);
extract($HTTP_COOKIE_VARS);
extract($HTTP_SERVER_VARS);
```

Далі оголошуємо змінні (лістинг 2.4):

Лістинг 2.4 – Оголошення змінних для скрипта

```
$fileName="stat.txt"; //ім'я файлу із статистикою
$maxVisitors=30; //кількість записів, що відображаються
```

При перегляді статистики (лістинг 2.5):

Лістинг 2.5 – Використання куків у скрипті

```
$cookieName="visitorOfMySite"; //ім'я куки
```

```
$cookieValue="1"; //значення куки
$timeLimit=86400;
```

Термін задано в секундах, який повинен пройти з моменту останнього відвідування сайту, щоб інформація про відвідувача записалася повторно. Це значення дорівнює одному дню, тобто один і той же відвідувач записується в статистику раз в одну добу. Якщо цю змінну прирівняти до нуля, то враховуватимуться усі відвідування одного і того ж відвідувача. Далі ідуть змінні, такі, що відповідають за відображення статистики (лістинг 2.6).

Лістинг 2.6 – Змінні, що відповідають за відображення статистики

```
$header_Color="#818181";
$header_FontColor="#EFFEFF";
$font_Face="Arial, Times New Roman ";
$font_Size="2";
$tableColor="#000011";
$row_Color="#CEDECEC";
$font_Color="#0AA0A0";
$text_FontColor="#00AA0A";
```

Усі змінні підготовлені.

Функція запису даних про відвідувача (лістинг 2.7).

Лістинг 2.7 – Функція запису даних про відвідувача

```
function saveUserData() {
GLOBAL $fileName, $HTTP_USER_AGENT, $REMOTE_ADDR,
$REMOTE_HOST,
 $HTTP_REFERER, $REQUES_URI;
$curTime=date("d.m.Y @ H : i: s"); //поточний час і дата
//готуємо дані для запису
if (empty($HTTP_USER_AGENT)){$HTTP_USER_AGENT = "Unkwnown";}
if (empty($REMOTE_ADDR)){$REMOTE_ADDR = "Not Resolved";}
if (empty($REMOTE_HOST)){$REMOTE_HOST = "Unknown";}
```

```
        if (empty($HTTP_REFERER)){$HTTP_REFERER = "No Referer";}
        if (empty($REQUEST_URI)){$REQUEST_URI = "Unknown";}
       $data_   =   $HTTP_USER_AGENT".   ::   ".$REMOTE_ADDR".   ::
".$REMOTE_HOST". :: ".$   ".$HTTP_REFERER". :: ".$REQUEST_URI". ::
".$curTime".\r\n";
```

Роздільником будемо використовувати два символи ":".

Наступний крок – запис зібраних даних у файл (лістинг 2.8).

Лістинг 2.8 – Запис зібраних даних у файл

```
if (is_writeable($fileName)) :
 $fp = fopen($fileName, "a");
  fputs ($fp, $data_);  fclose ($fp);
endif;
}
```

Функція запису готова. Тепер створимо функцію виведення даних з файлу статистики (лістинг 2.9).

Лістинг 2.9 – Функція виводу даних про відвідувачів сайту

```
function showStat () {
    GLOBAL $header_Color,
    $headerFont_Color,
    $font_Face,
    $font_Size,
    $table_Color,
    $file_Name,
    $max_Visitors,
    $row_Color,
    $font_Color,
    $text_Font_Color;
```

Виводимо таблицю (лістинг 2.10):

Лістинг 2.10 – Формування таблиці з даними

```
 $fbase=file($fileName);
 $fbase = array_reverse($fbase);
 $count = sizeOf($fbase);
  echo "<font face=\"$fontFace\"
color=\"$textFontColor\" size=\"$fontSize\">";
  echo "Усього відвідувань: $count<br><br>";
  echo "<div align=\"center\">
  <table cellpadding=\"2\" cellspacing=\"1\" width=\"95%\"
  border=\"0\" bgcolor=\"$tableColor\">";
  echo "<tr bgcolor=\"$headerColor\"><td><font
face=\"$fontFace\"
color=\"$headerFontColor\"  size=\"$fontSize\"> Браузер
  </font>
  </td><td>
  <font face=\"$fontFace\"
color=\"$headerFontColor\"
size=\"$fontSize\">IP</font>
  </td> <td>
<font face=\"$fontFace\" color=\"$headerFontColor\"
  size=\"$fontSize\">Хост</font></td>
  <td><font face=\"$fontFace\" color=\"$headerFontColor\"
  size=\"$fontSize\">Посилання</font></td>
  <td><font face=\"$fontFace\" color=\"$headerFontColor\"
  size=\"$fontSize\">Сторінка</font></td>
  <td><font face=\"$fontFace\" color=\"$headerFontColor\"
  size=\"$fontSize\">Час візиту</font></td></tr>";
  echo "</font><font face=\"$fontFace\" size=\"$fontSize\">";
```

Відкриваємо файл і запускаємо цикл (лістинг 2.11):

Лістинг 2.11 – Відкриття файлу з даними про відвідувачів

```
 $fbase=file($fileName);
```

```
 $fbase = array_reverse($fbase);
for ($i=0; $i<$maxVisitors; $i++) :
  if ($i>= sizeof($fbase)){break;}
 $s = $fbase[$i];
 //розділяємо
 $strr = explode("::"s);
  if (empty($strr)){break;}
```

Виводимо дані (лістинг 2.12):

Лістинг 2.12 – Формування сторінки звіту

```
echo "<tr><td bgcolor=\"$rowColor\"><
  font face=\"$fontFace\" color=\"$fontColor\"
  size=\"$fontSize\">$strr[0]</font>
  </td><td bgcolor=\"$rowColor\"><
  font face=\"$fontFace\" color=\"$fontColor\"
  size=\"$fontSize\">$strr[1]</font>
  </td><td bgcolor=\"$rowColor\"><
  font face=\"$fontFace\" color=\"$fontColor\"
  size=\"$fontSize\">$strr[2]</font>
  </td><td bgcolor=\"$rowColor\"><
  font face=\"$fontFace\" color=\"$fontColor\"
  size=\"$fontSize\">$strr[3]</font>
  </td><td bgcolor=\"$rowColor\"><
  font face=\"$fontFace\" color=\"$fontColor\"
  size=\"$fontSize\">$strr[4]</font>
  </td><td bgcolor=\"$rowColor\"><
  font face=\"$fontFace\" color=\"$fontColor\"
  size=\"$fontSize\">$strr[5]</font></td>
</tr>";
endfor;
}
?>
```

Скрипт збору і показу статистики готовий. Тепер його треба вставити в ті сторінки, інформацію про відвідувачів якої потрібно проглянути (лістинг 2.13):

Лістинг 2.13 – Використання скрипта збору статистики у HTML-сторінці

```
<?php
    include("sniffer.php");
    if (! isset($$cookieName)) :
   //встановити куки
      setcookie($cookieName, $cookieValue, time()+$timeLimit);
      saveUserData();
      endif;
?>
```

Слід звернути увагу, що цей код треба вставляти в самий верх сторінки, до того, як дані передаватимуться в браузер. Інакше встановити куки не вийде. Далі зробимо сторінку, котра виводить статистику (лістинг 2.14):

Лістинг 2.14 – Формування сторінки статистики

```
<html><body>
<?php include("sniffer.php"); ?>
Статистика<br>
<?php
  showStat();
?></body></html></i>
```

Тут ми просто включили файл sniffer.php і викликали з нього функцію showStat(). За допомогою такого невеликого скрипта, довжиною усього приблизно в 100 рядків, можна за допомогою PHP отримати і в зручному виді проглянути статистику відвідувань сайту. Тут ще багато чого належить зробити, наприклад, зробити статистику по реферерах, браузерах. Так само можна з HTTP_USER_AGENT витягнути браузер і ОС і записати їх в зручнішому виді.

До речі, усі розміри при виведенні статистики розраховані на роздільну здатність 1024*768 і все зручно поміщається в один рядок.

2.3 Аналіз технічного завдання

Згідно технічного завдання розроблюване програмне забезпечення призначене для ведення обліку відвідувань сайту. Інформація, яка збирається розробленим програмним забезпеченням, повинна містити:

- ведення обліку відвідувань сайтів;
- ведення обліку відвідувань сторінок сайту;
- формування списків по IP-адресах відвідувань;
- формування списку по реферерах (сторінках, з котрих виконано перехід на сайт);
- формування списків пошукових роботів, котрі індексували сайт.

Для цього можна піти двома шляхами.

1. Створення "вбудованого" в сторінки сайту скрипта, який викликатиметься кожен раз при відкритті сторінки.
2. Аналіз лог-файлів веб-сервера.

Кожен з цих методів має свої недоліки і переваги.

Для аналізу лог-файлів веб-сервера потрібно хоча б мати доступ до цих файлів. А в реальних умовах хостингу не всюди така можливість є. Крім того, абсолютно природно, що веб-сервер Apache та IIS ведуть логи абсолютно різних форматів. А це означає, що створити "універсальний" скрипт для статистики досить проблемно, особливо, коли врахувати можливе використання і інших серверів.

Тому для вирішення поставленої задачі оберемо принцип використання вбудованого PHP-скрипта, який виконується автоматично при завантаженні сторінки.

На даному етапі можна запропонувати концептуальну модель архітектури розробленої системи для ведення статистики відвідувань сайту. Вона зображена на рисунку 2.1.

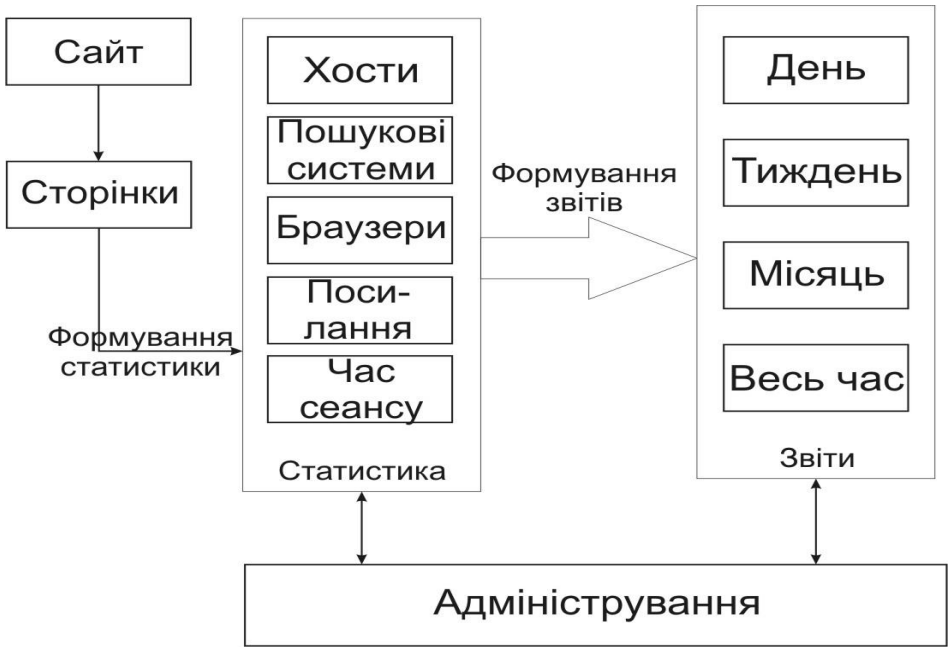


Рисунок 2.1 – Модель функціонування системи ведення статистики відвідувань сайту

З рисунка 2.1 видно, що система складається з блоків ведення статистики, формування звітів та адміністрування системи. Джерелом даних для збору є сам сайт, представлений окермими сторінками, При чому не важливо, чи це статичні сторінки, чи згенеровані динамічно в ході виконання певного сценарію чи cgi-додатку.

2.4 Розробка бази даних

У створеній системі використовується СКБД mySQL, під керуванням котрої створено базу даних для запису зібраної інформації про відвідування сайту.

Виділимо основні таблиці створеної БД:

- system_cities – таблиця для зберігання відомостей про міста та відповідні їм IP-адреси;
- system_ip – відомості про IP-адреси, з котрих заходили на сайт;
- system_ip_compact – скорочена таблиця з відомостями про IP-адреси, з котрих заходили на сайт (потрібна для формування деяких звітів);
- system_links – посилання на сайт;
- system_pages – статистика відвідування сторінок сайту;
- system_refferer – реферери на сайт;
- system_regions – таблиця для зберігання відомостей про регіони та відповідні їм IP-адреси;
- system_searchquerys – запити пошукових роботів, через котрі виходили на сайт;
- system_thits – статистика хітів (активації гіперпосилань) сайту.

Можна також сказати, що зібрана статистика архівується у допоміжних таблицях, перелік яких тут наводити не будемо.

Варто також зазначити, що хоча розроблена база даних складається з 40-ка таблиць, але реляційних зв’язків між ними немає, тобто ніякі правила цілісності посилань для розробленої БД не описано через відсутність потреби у таких реляційних зв’язках.

Відомості про назви всіх таблиць БД можна побачити в додатках.

Розглянемо структуру таблиць:

system_cities (city_id, region_id, city_name);

system_ip (id_ip, ip, putdate, id_page, browsers, systems);

system_ip_compact(init_ip, end_ip, city_id);

system_links (id_links, name, comment);

system_pages (id_page, name, title, id_site);

system_refferer (id_refferer, name, putdate, ip, ip_page);

system_regions (region_id, region_name);

system_searchquerys (quer_id, query, putdate, ip, ip_page, searches);

system_thits (hits).

Для створення бази даних запустимо на виконання SQL-скрипт для СКБД mySQL, текст котрого наведено у додатках.

2.5 Розробка інформаційної системи

Отже, тепер можна приступити до реалізації самої системи. Створення даного продукту засобами Macromedia Dreamweaver 8 досить обґрунтоване. Адже даний HTML-редактор дозволяє створювати не лише статичні сторінки, але також редагувати файли типу PHP, використовувати різноманітні клієнтські та серверні скрипти.

Перш ніж переходити до розробки інтерфейсу користувача розглянемо функціональну схему системи. Адже саме вона повинна бути спроектована на всі інтерфейсні елементи. Функціональна схема зображена на рисунку 1.1. На концептуальному рівні представлення роботи дана схема відображає загальний принцип застосування цієї системи.

У створеній системі функції збору відомостей і запису їх у БД реалізує один скрипт – count.php. Він вбудовується у сторінки, використання яких потрібно відстежувати. Про те, як це зробити, сказано далі, у наступних розділах.

За формування звітів відповідає частина скриптів адміністрування системи, Тому структурна схема системи може бути представлена у вигляді, зображеному на рисунку 2.2.

На рисунку не наведено деякі модулі системи, але їх текст можна побачити в додатках.

2.6 Робота з модулем збору статистики використання ресурсів сайту

У проекті представлено модуль для збору статистики відвідуваності сайту. По суті, це вбудований у сайт лічильник, але підраховуються не тільки

натискання користувачем певних гіперпосилань, але реєструється також і багато іншої корисної інформації. Цей лічильник відвідувань, написаний на PHP, є досить потужною системою збору і аналізу інформації про відвідувачів сайту.

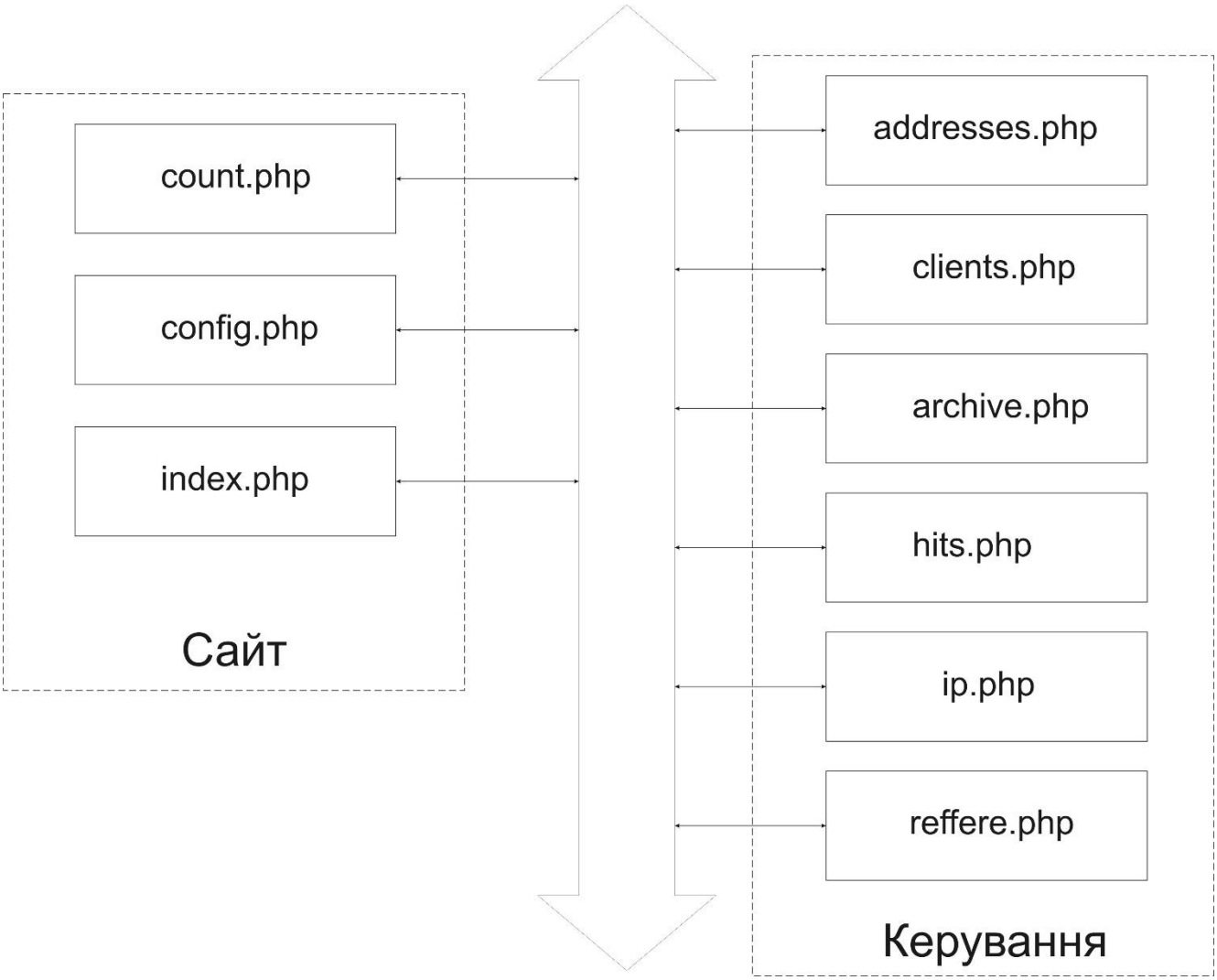


Рисунок 2.2 – Структурна схема системи

Це не просто лічильник – це інструмент, що дозволяє відстежувати потоки відвідувачів на вашому сайті, скільки сторінок вони переглядають, скільки часу перебувають на сайті, звідки вони потрапляють на ваш сайт, з інших сайтів або з пошукових систем, а які ключові слова використовуються? На усі ці питання дасть відповідь цей лічильник.

Перевагою вбудованого лічильника є те, що він на відміну від зовнішніх, даний фіксує усіх відвідувачів, а не тих, які завантажили зображення-банер.

Тому із статистики можна дізнатись і про те, чи відвідували сайт роботи пошукових систем, або чи використовують користувачі менеджери закачок, щоб переписати увесь сайт собі на машину і більше його не відвідувати.

Лічильник відвідувань збирає статистику про відвідування сайту, візуалізує загальну і зараховану кількість хостів, загальне і зараховане число хітів, як по окремих сторінках, так і сайту в цілому. Крім того, проводиться збір і виведення інформації про використовуваних відвідувачами операційних систем і браузерів. Реєструються відвідування роботів найбільш відомих пошукових систем, а також сторінки, які вони проіндексували. Виводяться ключові слова, по яких відвідувачі знаходили ресурс в пошукових системах. Крім того, ключові слова сортуються по частоті запитів і ви може оцінювати адекватність вашої аудиторії. Грубо кажучи, якщо сайт присвячений фіалкам, а його знаходять по запиту "Кольоровий телевізор Горизонт", можна бути впевненим, що відвідувач не затримається на сайті довше, ніж потрібно часу для натиснення на хрестик в правому верхньому кутку екрану, і потрібно щось терміново робити з оптимізацією сторінок по пошукових системах.

Лічильник відвідувань надає можливість додати адреси ресурсів, на яких розміщені посилання на сайт, і відстежувати число відвідувань з них. Інформація надається за 5 часових інтервалів: "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час". Для більшості звітів виводиться щодобова, потижнева і щомісячна статистика.

Об'єм бази даних, навіть для такого відвідуваного сайту не перевищує 2 Мб. Це пов'язано з організацією бази даних. Повна інформація зберігається лише за добу, після чого вона піддається стискуванню і поміщається в добові архівні таблиці. Після тижня інформація стискується в тижневі таблиці, а по закінченню місяця в місячні таблиці. Зрозуємілo, що вся інформація, яка не вимагається для виведення звітів, видаляється. Звичайно, це досягається за рахунок деякого зниження функціональності, але це можна усунути. Окрім цього в розробленій системі статистики можна використовувати таблиці із змінними іменами.

Ведеться облік браузерів Firefox, Mozilla і IE, розширена таблиця pages, для поступового переходу з URL на назви сторінок. Тепер додавши на вашу сторінку назву в змінній $titlepage замість малозрозумілого URL, будуть виводиться назви.

Статистика IP-адрес відображує добові, тижневі і місячні звіти з посторінковою навігацією. Є статистика для Глибини відвідування і Часу відвідування, а також введена щодобова, тижнева і щомісячна статистика для реферерів.

Введено визначення приналежності IP-адреси по містах і регіонах (введено тільки декілька регіонів, для підтримки інших потрібно відповідними даними заповнити таблиці).

2.6.1 Установка модуля збору статистики відвідувань сайту

Для успішного функціонування модуля, необхідно створити базу даних (за замовчуванням count), в якій слід розмістити таблиці з файлу PowerCoutner.sql. Залежно від хостингу користувач має для створення бази даних різні інструменти: він може це зробити або з використанням веб-інтерфейсу або виконавши SQL запити на створення БД (create database count;).

Налаштування з'єднання з базою даних здійснюється у файлі admin/config.php, в якому необхідно ввести адресу сервера MySQL ($dblocation), задати ім'я БД ($dbname), ідентифікатор користувача ($dbuser) та його пароль ($dbpasswd). Усі чотири змінні повинен надати хостинг. Для того, щоб сторінка на сайті була досяжна дл я лічильника і він вів по ній статистику, треба на її початку добавити інструкцію влючення файлу count.php (лістинг 2.15):

Лістинг 2.15 – Використання лічильника

```
<?php
      include "count.php";
?>
```

як це продемонстровано в тестовій сторінці index.php. Користувач не обмежений вибором місця розміщення цієї конструкції. Довантажує відвідувач сторінку до кінця або ні – не має зовсім ніякого значення – він буде врахований. Це пов'язано з тим, що PHP -код виконується на сервері і доки не буде виконаний, клієнтові нічого відправлено не буде. Тому коли відвідувач отримує тільки перші байти, він вже врахований.

При першому ж зверненні до сторінки відвідувачем, в таблиці pages буде створений запис, що відповідає цій сторінці і вона буде зберігатись і відображатись у розроблюваній системі статистики. Кількість сторінок – необмежена.

Якщо перед файлом включенням файлу за допомогою інструкції require_once помістити ім'я сторінки в змінній $titlepage, в звітах системи, ця сторінка братиме участь під цим ім'ям. Більше того, можна об'єднувати декілька сторінок в один рядок, присвоюючи їм однакові назви (лістинг 2.16).

Лістинг 2.16 – Робота з декількома сторінками

```
<?php
     $     $titlepage = "Назва сторінки";
      require_once("count.php");
?>
```

Окремо слід згадати архівацію робочих таблиць в добові, тижневі і місячні таблиці. Стискування відбувається в 00:00, при першому відвідуванні сторінки адміністрування. Проте можемо самостійно змусити систему стискувати дані, наприклад, по cron – за цей процес відповідає скрипт admin/archive.php.

Прив'язавши його до cron, можна архівувати дані самостійно. Проте якщо це зовсім не обов'язково, досить просто відвідувати систему адміністрування час від часу і система сама усе зробить.

2.6.2 Проблеми при інсталяції та модифікація

Слід пам'ятати, що сервера – це не клієнтські машини з Windows XP, вони і їх налаштування часто відрізняються один від одного. Тому виникнення проблем дуже ймовірно і пов'язані вони в першу чергу з нестандартними серверними налаштуваннями змінних. Тому якщо щось не рахується і не враховується, слід аналізувати через PHP-функцію phpinfo() та порівнювати ідентифікатори змінних оточення з такими самими ідентифікаторами у файлі count.php, а при виявленні неспівпадінь – виправляти змінні на ті, які вказані у звіті phpinfo().

За замовчуванням, лічильник рахує сторінки, що відрізняються тільки параметрами за одну сторінку, наприклад, сторінки

```
index.php?id=1
index.php?id=2
```

вважатимуться як одна і та ж сторінка (index.php), з усіма витікаючими наслідками. Для того, щоб такі сторінки вважалися як різні необхідно в кодах усього лічильника замінити $_SERVER['PHP_SELF'] на $_SERVER['REQUEST_URI'] – можна просто відкрити кожен файл в блокноті і скористатися функцією автозаміни.

2.6.3 Адміністрування модуля збору статистики відвідувань сайту

Слідкувати за популярністю сторінок (рахувати кількість переходів) на сторінці адміністрування лічильника розташованою в каталозі admin (файл index.php).Головна сторінка містить список всіх сторінок, що включені до збору статистики, біля кожної з них виводиться загальне число переодів з моменту реєстрації статистики.

Вгорі є меню з посиланнями на сторінки статистики. Перехід по меню дозволяє проглянути статистку для усього сайту в цілому, тоді як перехід по посиланнях з таблиці – статистику для кожної конкретної сторінки.

Перехід по кожній із сторінок призводить до сторінки з таблицею, в якій вказана кількість хітів і хостів за 5 часових інтервалів, : "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час" для цієї сторінки сайту.

Перехід по гіперпосиланнях "Сьогодні" і "Вчора" призводить до сторінки розподілу хостів і хітів по годиннику в ці дні. Відповідно, перехід по гіперпосиланнях "за 7 днів" і "за 30 днів" показує зміну кількості хітів і хостів цей час.

Меню складається з наступних пунктів:

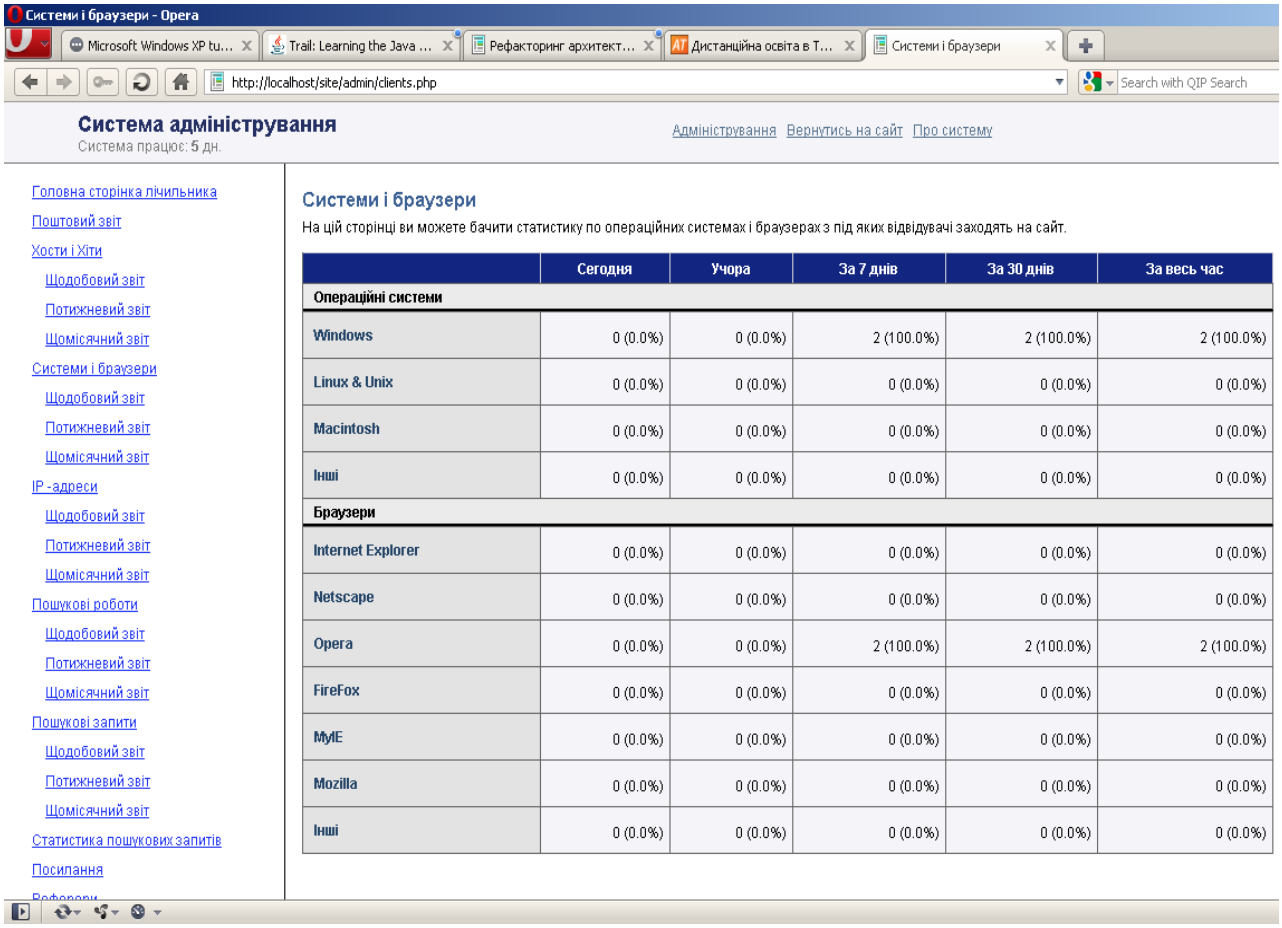
- "Поштовий звіт";
- "Хіти і хости" (щодобовий, потижневий, щомісячний звіти);
- "Системи і браузери" (щодобовий, потижневий, щомісячний звіти);
- "IP-адреси і хости" (щодобовий, потижневий, щомісячний звіти);
- "Пошукові роботи" (щодобовий, потижневий, щомісячний звіти);
- "Пошукові запити" (щодобовий, потижневий, щомісячний звіти);
- "Статистика пошукових запитів";
- "Сторінки з посиланнями";
- "Реферери" (щодобовий, потижневий, щомісячний звіти);
- "Точки входу";
- "Глибина перегляду" (щодобовий, потижневий, щомісячний звіти);
- "Час сеансу" (щодобовий, потижневий, щомісячний звіти).

Перехід за посиланням "Поштовий звіт" відкриває сторінку, з можливістю відправки електронною поштою звітів за минулий день, тиждень, місяць на поштову скриньку адміністратора, задати яку можна в константі EMAIL_ADDRESS в конфігураційному файлі admin/config.php.

Перехід за посиланням "Хіти і хости" приводить до сторінки з таблицею, в якій вказана кількість зарахованих хостів і загальне число хітів за 5 часових інтервалів: "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час" для цієї сторінки сайту. Перехід по гіперпосиланнях "Сьогодні" і "Вчора" відкриває сторінки розподілу хостів і хітів по годиннику в ці дні. Відповідно, перехід по

гіперпосиланнях "за 7 днів" і "за 30 днів" відкриває таблицю з динамікою зміни кількості хітів і остов за тиждень і за місяць.

Перехід до сторінки "Системи і браузери" призводить до сторінки з таблицею, в якій вказана кількість відвідувачів що використовують як операційні системи Windows, один з варіантів UNIX і Macintosh, а також що користуються браузерами Google Chrom, Firefox, та інших відповідно до налаштувань за 5 часових інтервалів: "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час" для цієї сторінки сайту (рисунок 2.3).



	Сегодня	Учора	За 7 днів	За 30 днів	За весь час
Операційні системи					
Windows	0 (0.0%)	0 (0.0%)	2 (100.0%)	2 (100.0%)	2 (100.0%)
Linux & Unix	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Macintosh	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Інші	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Браузери					
Internet Explorer	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Netscape	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Opera	0 (0.0%)	0 (0.0%)	2 (100.0%)	2 (100.0%)	2 (100.0%)
FireFox	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
MyIE	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Mozilla	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Інші	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)

Рисунок 2.3 – Статистика по операційних системах та браузерах

На сторінці "IP-адреси і хости" приводиться таблиця з IP-адресами відвідувачів за останню добу. Для кожної IP-адреси надається інформація про хост, дату останнього відвідування і загальне число відвідувань з цієї IP-адреси.

На сторінці "Пошукові роботи" виводиться таблиця відвідування ресурсу роботами найбільш відомих пошукових систем (Google, BING та інших) за 5

часових інтервалів: "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час" для цієї сторінки сайту. Крім того, по посиланнях можна прослідкувати які сторінки були проіндексовані роботами пошукових систем (рисунок 2.4).

Варто відмітити, що нульові значення, відображені на рисунку, означають, що розроблений модуль тестувався на локальній машині для тестового сайту, і, природно, цей сайт не міг бути проіндексований ніякою пошуковою системою.

На сторінці "Пошукові запити" виводиться таблиця з інформацією про число пошукових запитів і самі ключові слова, по яких ресурс був виявлений як для пошукових систем, так і для кожної окремо.

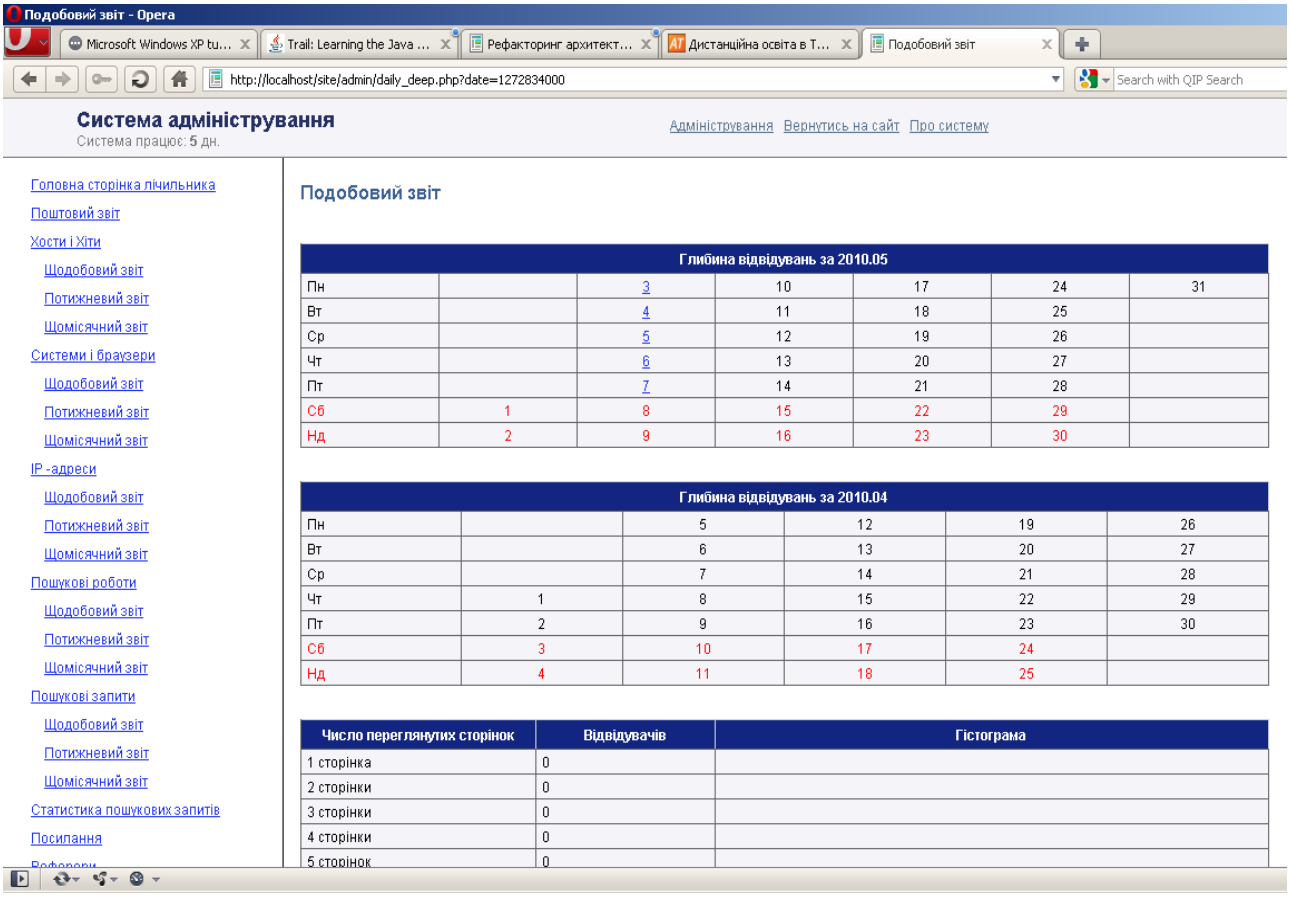


Рисунок 2.4 – Статистика по пошукових роботах

На сторінці "Статистика пошукових запитів" виводиться список ключових слів, по яких відвідувачі знайшли сайт в пошукових системах. Крім того, виводиться частота використання ключових слів відвідувачами.

На сторінці "Сторінки з посиланнями" можна додати адреси ресурсів, на яких розташовані посилання на сайт (наприклад, каталоги типу google.com), що дозволить стежити за переходами на сайт з цих ресурсів.

На сторінці "Точки входу" можна визначити найбільш популярні сторінки сайту, через які відвідувачі потрапляють на цей же сайт, за рахунок посилань з інших сторінок або пошукових систем. Якщо при загальній високій відвідуваності сайту для якихось сторінок ви спостерігаєте цифру 0 – ця сторінка невдало спроектована і через неї відвідувачі не потрапляють на ваш ресурс.

На сторінці "Глибина перегляду" можна з'ясувати інтерес відвідувачів до вашого сайту. Ця сторінка надасть інформацію про число відвідувачів тих, що проглянули 1, 2, 3 і т.д. сторінок сайту. Статистика виводиться за 5 часових інтервалів: "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час" (рисунок 2.5).

На сторінці "Час сеансу" можна з'ясувати скільки часу проводять відвідувачі на сайті. Ця сторінка надасть інформацію про число відвідувачів, що пробули на сайті 1, 2, 3 хвилини або можливо декілька годин. Статистика виводиться за 5 часових інтервалів: "Сьогодні", "Вчора", "за 7 днів", "за 30 днів" і "за весь час".

Крім того, на головній сторінці адміністрування є інструмент для виявлення і видалення так званих мертвих посилань, які з'являються як результат помилок в дизайні. Якщо видалити сторінку, підключену до лічильника із структури ресурсу, то потрібно також видалити його з таблиці pages, активуваши посилання на цю операцію ("Видалити").

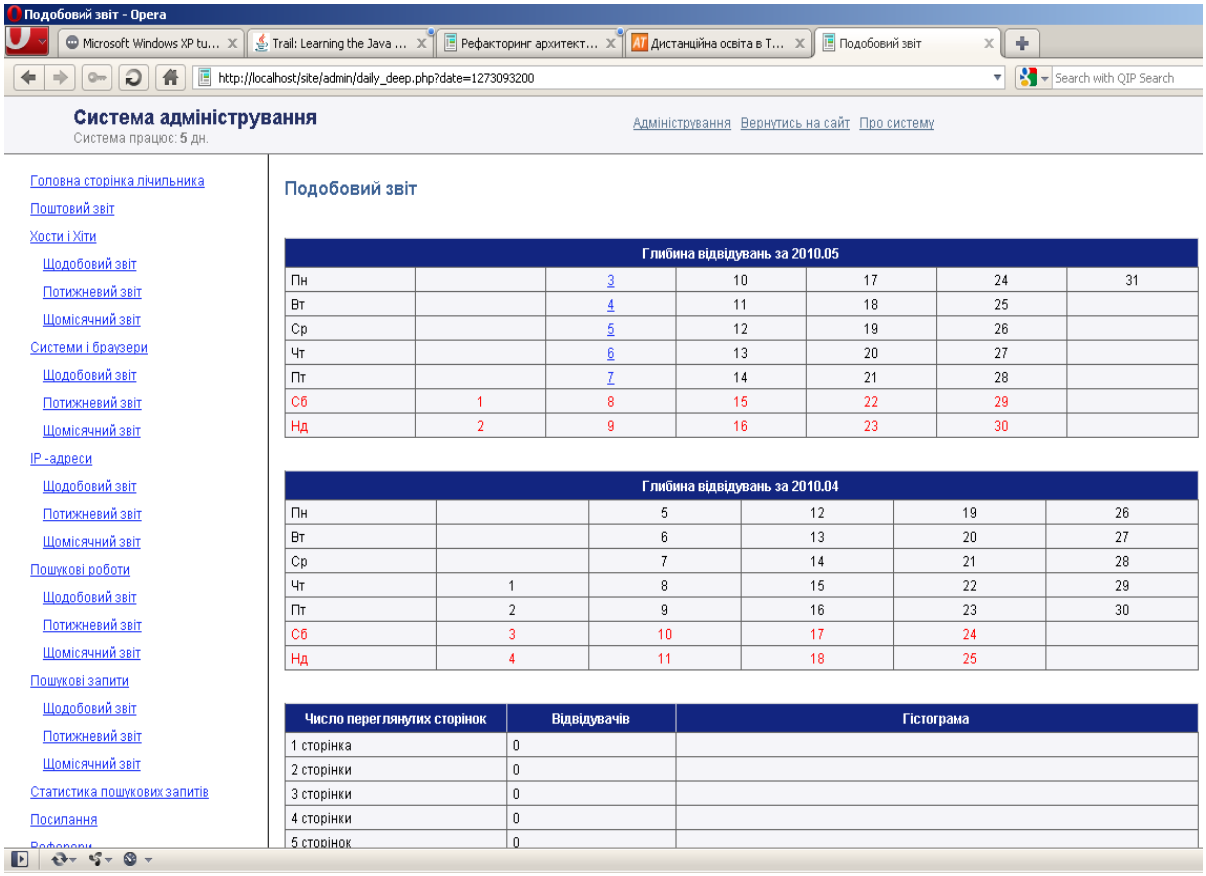


Рисунок 2.5 – Подобова статистика про глибину перегляду сайту

Зрозуміло, що доступ до каталогу admin має бути захищений.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Загальні вимоги законодавства з охорони праці в галузі інформаційних технологій

Правову основу охорони праці становить Конституція України як за своїми юридичними особливостями, так і своїми принципами, тобто юридично вираженими об'єктивними закономірностями організації і функції соціально-економічної, політичної, духовної сфер суспільства, правового положення особи.

Конституційні норми, з одного боку, закладають суть безпеки (норми-принципи), а з іншого, – вказують на цілі подальшого розвитку і реалізацію правового забезпечення безпеки життєдіяльності (норми-програми, норми-завдання, норми-зобов'язання).

Реалізація і розвиток основних конституційних положень, які регламентують суспільні правовідносини, безпосередніми суб'єктами яких є особа і держава, здійснюється за допомогою як чинних фундаментальних нормативно-правових актів, так і спеціальних (Кодексів України про працю, Закону "Про охорону навколишнього природного середовища" та ін.)

Поруч з нормативними актами, які прийняті вищим законодавчим органом держави, для встановлення взаємозв'язків, усунення програм, а в ряді випадків і реалізації окремих правових норм або їх елементів, до правової бази безпеки життєдіяльності належать спеціальні акти, розроблені за дорученням виконавчих державних органів усіх рівнів (Кабінет Міністрів, Міністерства, Державні Комітети та ін.).

Так, наприклад, "Положення...", які розвивають Закон України Про охорону праці, діляться на звичайні "Положення" і "Типові положення". Тут держава розподілила питання своєї прерогативи стосовно розробки нормативних актів і прерогативи своїх повноважень стосовно контролю, "правового простору" у вигляді нормативних актів підприємств.

З іншого боку, формуючи систему "Типових положень" держава на сьогоднішній день ліквідує прогалини в чинному законодавстві, узгоджує взаємозв'язки між суб'єктами правовідносин, створює юридичну базу для удосконалення і розвинення "правового поля" підприємств.

Кожний нормативно-правовий документ має свою структуру, яка визначає собою ідею систематизації відповідно зі своїм рівнем, метою та завданнями. Відповідно до цього в кожному нормативному акті є елементи, що відповідальні за зовнішній його зв'язок і створення передумов для відповідного розвинення за рахунок розробки нижчих нормативно-законодавчих актів. Сама структура нормативного акта формує відповідні внутрішні зв'язки.

Основними систематизуючими ланками нормативних актів безпеки життєдіяльності (які за ієрархією знаходяться нижче законів) є встановлення взаємовідносин в галузі виробництва, в межах дії небезпечного фактора (в тому числі і факторів довкілля), а також відносно управління основних технологій безпеки життєдіяльності (розслідування нещасних випадків, навчання, організації робіт та ін.).

Узагальнюючими ланками систематизації на рівні держави є національна ідея, взаємовідносини в суспільстві, соціально-економічне і політичне становище держави, можливості сприймання і використання законодавчих актів з боку споживачів та ін.

3.2 Розрахунок освітленості робочого місця розробника модуля інформаційної системи для збору статистики про відвідування сайту

Належне освітлення необхідне для виконання більшості задач програміста, а, отже, і розробника даного дипломного проекту. Для того, щоб спланувати раціональну систему освітлення, враховується специфіка робочого завдання, для якого створюється система освітлення, швидкість і точність, з якою це робоче завдання повинне виконуватися, тривалість його виконання і різні зміни в умовах виконання робітників.

Приміщення, у якому знаходиться робоче місце, має наступні характеристики:

- довжина приміщення 16 м;
- ширина приміщення 6 м;
- висота 4 м;
- кількість вікон 3;
- кількість робочих місць 3;
- біла стеля, блідо-зелені стіни, підлога обтягнута лінолеумом зеленого кольору.

У приміщенні, де знаходиться робоче місце, використовується змішане освітлення, тобто сполучення природного і штучного освітлення.

У якості природного – бічне освітлення через вікна. Штучне освітлення використовується при недостатньому природному освітленні.

Розрахунок його здійснюється по методу світлового потоку з врахуванням потоку, відбитого від стін і стелі.

Нормами для даних робіт встановлена необхідна освітленість робочого місця Ен = 300 лк [1]. Загальний світловий потік визначається за формулою:

$$F_{заг} = \frac{E_н * S * z1 * z2}{V}, \quad (3.1)$$

де $E_н$ – нормована освітленість ($E_н$ = 300 лк);

S – площа приміщення;

z1 – коефіцієнт, що враховує старіння ламп і забруднення світильників (z1 = 1,5) [1];

z2 – коефіцієнт, що враховує нерівномірність освітлення приміщення (z2 = 1,1) [1];

V – коефіцієнт використання світлового потоку; визначається в залежності від коефіцієнтів відбивання від стін, стелі, робочих поверхонь, типів світильників і геометрії приміщення.

Площа приміщення S = A*B = 16*6 = 96 м^2.

- коефіцієнт відбивання побіленої стелі $R_п$ = 70% [1];
- коефіцієнт відбивання від стін, пофарбованих у світлий колір $R_{ст}$ = 50% [1];
- коефіцієнт відбивання від підлоги, покритого лінолеумом темного кольору $R_р$ = 10% [1];
- індекс приміщення $i = \frac{A*B}{h*(A+B)} = \frac{16*6}{4*(16+6)} = 1{,}1$.

Знайдений коефіцієнт V = 0,34.

За формулою (3.1) визначаємо загальний світловий потік

$$F_{заг} = \frac{300*96*1{,}1*1{,}5}{0{,}34} = 139764 \text{ лм.}$$

Для організації загального штучного освітлення виберемо лампи типу ЛБ40. Люмінесцентні лампи мають ряд переваг перед лампами накалювання: їхній спектр ближче до природного; вони мають велику економічність (більша світловіддача) і термін служби у 10-12 раз більший.

Поряд з цим вони мають і недоліки: їхня робота супроводжується іноді шумом; гірше працюють при низьких температурах; не можна використовувати у вибухонебезпечних приміщеннях.

Для нашого приміщення люмінесцентні лампи підходять. Світловий потік однієї лампи ЛБ40 складає не менш $F_л$ = 2810 лм. Число N ламп, необхідних для організації загального освітлення визначається наступним чином:

$$N = \frac{F_{заг}}{F_л} = \frac{139764}{2810} = 50 \text{ шт.}$$

Як світильники вибираємо ПВЛ-1, 2*40 Вт. Таким чином, щоб забезпечити світловий потік $F_{заг}$ = 139764 лм треба використати 25 світильників по 2 лампи ЛБ40 у кожному.

Електрична потужність однієї лампи $W_л$ = 40 Вт. Потужність всієї освітлювальної системи:

$$W_{заг} = W_л * N = 40*50 = 2000 \text{ Вт.}$$

Зі зробленого в даному розділі розрахунку випливає, що для нормальної роботи користувача робочого місця необхідне загальне освітлення приміщення зі світловим потоком 139764 лм, для чого необхідна наявність 25 світильників типу ПВЛ-1 з двома лампами типу ЛБ40. Крім того рекомендується використовувати ряд спеціальних заходів захисту від шкідливих факторів екрана дисплея, наприклад, використання занавісок на вікнах.

3.3 Аналіз небезпеки і шкідливості при розробці модуля інформаційної системи збору статистики використання ресурсів сайту

Організація робочого місця розробника модуля впливає на його працездатність.

У своїй діяльності розробник використосвує комп’ютер, пристрої збереження інформації, а тому є необхідність забезпечення зручного доступу до всіх технічних засобів. Тому в даному розділі докладніше розглянемо відомості про систему ергономічних норм і принципів організації робочого місця, на котрому проводяться роботи зі створення модуля збору статистики.

Під робочим місцем розуміється зона, оснащена необхідними технічними засобами, у якій відбувається трудова діяльність виконавця або групи виконавців, які спільно виконують одну роботу або операцію.

Організація робочого місця полягає у виконанні заходів, які забезпечують безпечний і раціональний трудовий процес і ефективне використання знарядь та предметів праці, що підвищує продуктивність праці і знижує стомлюваність працівника.

Організація робочого місця залежить від характеру розв'язуваних задач і особливостей предметно-просторового оточення, що визначають робоче

положення тіла і можливість пауз для відпочинку, типи і способи засобів відображення і керування, необхідність у засобах захисту, спецодягу, простору для налагодження і ремонту устаткування.

Одним з компонентів діяльності на робочому місці є робочі рухи. Їхня раціональна організація створює умови для зниження стомлення, резерви для підвищеної працездатності. Просторові характеристики руху оператора визначаються траєкторіями руху і розмірами моторного поля (зони досяжності).

При організації робочого місця необхідно забезпечити нормальні умови огляду. Зону огляду описує кут, вершина якого знаходиться в центрі ока, а сторони складають границі, в яких людина при фіксованому положенні голови й ока добре розрізняє їхнє місцезнаходження.

У горизонтальній площині цей кут складає 30^0 – 40^0. При організації робочого місця кут огляду можна взяти 50^0 – 60^0, включаючи зону менш ясного огляду. Допустимий кут огляду по горизонталі 90^0. У вертикальній площині оптимальний кут огляду 10^0 вверх і 30^0 вниз від лінії погляду, а допустимий 30^0 вгору і 40^0 вниз від лінії погляду.

Щоб зберегти нормальну гостроту зору, робочу поверхню розташовують від очей на відстані від 0,3 м до 0,75 м. Робочі меблі повинні бути зручними для виконання робочих операцій. В даному випадку робочий стіл є основним устаткуванням. Особливо важливе значення має висота столу, його конструкція, яка повинна передбачати шухляди для розміщення інструментів, документації.

Важливе значення має конструкція робочих крісел. Погано підібрані крісла можуть бути причиною надмірної стомлюваності.

Нахил і висота крісла повинні регулюватися відповідно до висоти робочої поверхні і росту працюючого. Рекомендована ширина крісла 370 – 400 мм, глибина 370 – 420 мм, висота спинки 370 – 1000 мм від рівня крісла. Для розміщення ніг необхідно передбачити вільний простір під робочою площиною [1].

Праця людини, що протікає в умовах надмірного нервово-емоційного напруження, довготривалих статичних навантажень, обмеженої рухової

активності призводить до неврозів, відхилень у психіці, захворювань опорно-рухового апарату, серцево-судинної системи тощо. Комп'ютери, телебачення, системи зв'язку та інші засоби, що використовують досягнення радіоелектроніки, є генераторами цілої низки електромагнітних випромінювань, вплив яких на організм людини ще не зовсім вивчений.

Сучасний розвиток науки та техніки приносить принципові нововведення у всі сфери матеріального виробництва, докорінно змінюючи знаряддя та предмети праці, технологію, методи обробки інформації. Разом з тим, захопившись вдосконаленням засобів праці залишено поза увагою проблеми людини в рамках своєрідної технічної та комп'ютерної революції. З широким впровадженням автоматизації та комп'ютеризації виникла потреба врахування психологічних можливостей людини, таких як швидкість реакції, особливості пам'яті та уваги, емоційний стан та ін. Поява операторської діяльності призвела до суттєвих змін у фаховій структурі праці. Зменшились фізична важкість праці, ризик виробничого травматизму, однак разом з тим, на працюючу людину посилюється вплив нових, раніше не відомих чи мало вивчених несприятливих виробничих факторів фізичного, хімічного і особливо психофізіологічного характеру.

Проте, розвиток сучасної обчислювальної техніки відбувається не лише у бік поктащення її технічних параметрів, але також звертається увага безпеку використання цієї техніки людиною шляхом зменшення потужності випромінювачів, зменшенням рівня випромінювання з моніторів, зменшення напруг живлення, покращення ергономічних характеристик.

Таким чином, в розділі з охорони праці виконано огляд питань безпечної роботи при створенні модуля інформаційної системи збору статистики та встановлено, що умови такої роботи відповідають вимогам з охорони праці, які застосовуються в галузі інформаційних технологій.

ВИСНОВОК

Підводячи підсумки виконаної роботи , можна зробити наступні висновки. Було реалізовано модуль інформаційної системи обліку відвідувань сайту на основі технічного завдання. Виконано огляд можливостей програмного забезпечення для вирішення поставленої задачі на виконання дипломної роботи.

У роботі проведений аналіз існуючої системи збору статистики використання ресурсів сайту, аналіз існуючих способів розв’язування задачі. Обґрунтований вибір проектних рішень по створенню інформаційного та програмного забезпечення, по створенню технології збору, передачі, підготовки та обробки інформації. Крім того, в процесі подальшої роботи було розроблено інформаційну базу, спроектовано форми звітів, структури таблиць бази даних, а також – інтерфейс користувача.

Для практичної спроектованої системи було вибрано мову PHP та вбудовуваний PHP-сценарій для підрахунку статистики використання кожної сторінки сайту. Це дозволить використовувати систему без додаткових затрат на вивчення. Розроблений модуль для підрахунку статистики є простим в користуванні, з відкритим доступом до бази даних. Розробка розв’язує задачу, яка відповідають вимогам технічного завдання.

В кваліфікаційній роботі також розглянуто питання безпеки життєдіяльності та охорони праці.

ПЕРЕІЛК ПОСИЛАНЬ

1. Безпека життєдіяльності: Навч. посібник./ За ред. В.Г. Цапка. – 4-те вид., перероб. і доп. – К.: Знання, 2006. – 397 с.
2. Буров Є. Комп'ютерні мережі. Львів.: БАК, 1999. – 468с.,іл.
3. Высокопроизводительные сети. Энциклопедия пользователя. Марк А. Спортак и др.; перев. с англ. - Киев, ДиаСофт, 2001.
4. Колисниченко Д.Н. Самоучитель PHP 5. –СПб.: Наука и Техника, 2004. – 576 с.: ил.
5. Компьютерные сети. Учебный курс, 2-е изд. (+CD-ROM). – MicrosoftPress, Русская редакция, 1999. – 364 с.
6. Котеров Д.В. Самоучитель PHP 4. – СПб.: БХВ-Петербург., 2001. – 576 с.: ил.
7. Оглтри Т. Модернизация и ремонт сетей. Пер. с .англ. -М.: QUE, 2000. – 928с., іл.
8. Персональные компьютеры в сетях TCP/IP. Крейг Хант; перев. с англ. – BHV-Киев, 2002. – 251 с.
9. Протоколы Internet. С. Золотов. - СПб.: BHV - Санкт-Петербург, 2001. – 423 с.
10. Сетевые средства Microsoft Windows 2000 Server; перев. с англ. СПб.: – BHV-Санкт-Петербург, 1997.
11. Стахнов А.А. Linux. СПб.: БХВ-Петербург., 2003. – 912 с.: ил.
12. Титтэл, Эд, Питс, Натанья, Валентайн, Челси. HTML 4 для чайников, 3-е издание.: Пер. с англ.: Уч. пос. – М.: Издательский дом "Вильямс", 2001. – 464 с.: ил. – Парал. тит. англ.

ДОДАТКИ

Додаток А

SQL-запити на створення бази даних

```
CREATE TABLE system_ip (
  id_ip int(32) NOT NULL auto_increment,
  ip bigint(11) NOT NULL default '0',
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  id_page int(10) NOT NULL default '0',
  browsers enum('none','msie','opera','netscape','firefox','myie','mozilla')
NOT NULL default 'none',
  systems
enum('none','windows','unix','macintosh','robot_yandex','robot_google','robot_r
ambler','robot_aport','robot_msnbot') NOT NULL default 'none',
  PRIMARY KEY  (id_ip),
  KEY putdate (putdate),
  KEY ip (ip)
) TYPE=MyISAM;
CREATE TABLE system_links (
  id_links int(8) NOT NULL auto_increment,
  name text,
  comment text,
  PRIMARY KEY  (id_links)
) TYPE=MyISAM;
CREATE TABLE system_pages (
  id_page int(10) NOT NULL auto_increment,
  name text,
  title text,
  id_site int(4) default NULL,
  PRIMARY KEY  (id_page)
) TYPE=MyISAM;
CREATE TABLE system_refferer (
  id_refferer int(16) NOT NULL auto_increment,
  name text NOT NULL,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  ip bigint(11) NOT NULL default '0',
  id_page int(8) NOT NULL default '0',
  PRIMARY KEY  (id_refferer),
  KEY id_page (id_page),
  KEY ip (ip)
) TYPE=MyISAM;
CREATE TABLE system_searchquerys (
  quer_id int(11) NOT NULL auto_increment,
  query tinytext NOT NULL,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  ip bigint(20) NOT NULL default '0',
  id_page int(11) NOT NULL default '0',
  searches enum('yandex','google','rambler','aport','mail','msn') NOT NULL
default 'yandex',
  PRIMARY KEY  (quer_id)
) TYPE=MyISAM;
CREATE TABLE system_thits (
hits int(11)
) TYPE=MyISAM;

CREATE TABLE system_arch_hits (
  id_ip int(3) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  hosts_total int(11) NOT NULL default '0',
  host int(11) NOT NULL default '0',
  hits_total int(11) NOT NULL default '0',
  hits int(11) NOT NULL default '0',
  PRIMARY KEY  (id_ip)
) TYPE=MyISAM;
```

```
CREATE TABLE system_arch_hits_month (
  id_ip int(3) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  hosts_total int(11) NOT NULL default '0',
  host int(11) NOT NULL default '0',
  hits_total int(11) NOT NULL default '0',
  hits int(11) NOT NULL default '0',
  PRIMARY KEY  (id_ip)
) TYPE=MyISAM;
CREATE TABLE system_arch_hits_week (
  id_ip int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  hosts_total int(11) NOT NULL default '0',
  host int(11) NOT NULL default '0',
  hits_total int(11) NOT NULL default '0',
  hits int(11) NOT NULL default '0',
  PRIMARY KEY  (id_ip)
) TYPE=MyISAM;

CREATE TABLE system_arch_ip (
  id_ip int(3) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  ip bigint(20) NOT NULL default '0',
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_ip)
) TYPE=MyISAM;
CREATE TABLE system_arch_ip_month (
  id_ip int(3) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  ip bigint(20) NOT NULL default '0',
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_ip)
) TYPE=MyISAM;
CREATE TABLE system_arch_ip_week (
  id_ip int(3) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  ip bigint(20) NOT NULL default '0',
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_ip)
) TYPE=MyISAM;

CREATE TABLE system_arch_clients (
  id_client int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  browsers_msie int(11) NOT NULL default '0',
  browsers_opera int(11) NOT NULL default '0',
  browsers_netscape int(11) NOT NULL default '0',
  browsers_firefox INT(11) NOT NULL default '0',
  browsers_myie INT(11) NOT NULL default '0',
  browsers_mozilla INT(11) NOT NULL default '0',
  browsers_none int(11) NOT NULL default '0',
  systems_windows int(11) NOT NULL default '0',
  systems_unix int(11) NOT NULL default '0',
  systems_macintosh int(11) NOT NULL default '0',
  systems_none int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_clients_month (
  id_client int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  browsers_msie int(11) NOT NULL default '0',
  browsers_opera int(11) NOT NULL default '0',
```

```
  browsers_netscape int(11) NOT NULL default '0',
  browsers_firefox INT(11) NOT NULL default '0',
  browsers_myie INT(11) NOT NULL default '0',
  browsers_mozilla INT(11) NOT NULL default '0',
  browsers_none int(11) NOT NULL default '0',
  systems_windows int(11) NOT NULL default '0',
  systems_unix int(11) NOT NULL default '0',
  systems_macintosh int(11) NOT NULL default '0',
  systems_none int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_clients_week (
  id_client int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  browsers_msie int(11) NOT NULL default '0',
  browsers_opera int(11) NOT NULL default '0',
  browsers_netscape int(11) NOT NULL default '0',
  browsers_firefox INT(11) NOT NULL default '0',
  browsers_myie INT(11) NOT NULL default '0',
  browsers_mozilla INT(11) NOT NULL default '0',
  browsers_none int(11) NOT NULL default '0',
  systems_windows int(11) NOT NULL default '0',
  systems_unix int(11) NOT NULL default '0',
  systems_macintosh int(11) NOT NULL default '0',
  systems_none int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_robots (
  id_client int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  yandex int(11) NOT NULL default '0',
  rambler int(11) NOT NULL default '0',
  google int(11) NOT NULL default '0',
  aport int(11) NOT NULL default '0',
  msn int(11) NOT NULL default '0',
  none int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_robots_month (
  id_client int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  yandex int(11) NOT NULL default '0',
  rambler int(11) NOT NULL default '0',
  google int(11) NOT NULL default '0',
  aport int(11) NOT NULL default '0',
  msn int(11) NOT NULL default '0',
  none int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_robots_week (
  id_client int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  yandex int(11) NOT NULL default '0',
  rambler int(11) NOT NULL default '0',
  google int(11) NOT NULL default '0',
  aport int(11) NOT NULL default '0',
  msn int(11) NOT NULL default '0',
  none int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_enterpoint (
  id_enterpoint int(11) NOT NULL auto_increment,
```

```
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  page tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_enterpoint)
) TYPE=MyISAM;
CREATE TABLE system_arch_enterpoint_month (
  id_enterpoint int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  page tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_enterpoint)
) TYPE=MyISAM;
CREATE TABLE system_arch_enterpoint_week (
  id_enterpoint int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  page tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_enterpoint)
) TYPE=MyISAM;

CREATE TABLE system_arch_deep (
  id_client int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  visit1 int(11) NOT NULL default '0',
  visit2 int(11) NOT NULL default '0',
  visit3 int(11) NOT NULL default '0',
  visit4 int(11) NOT NULL default '0',
  visit5 int(11) NOT NULL default '0',
  visit6 int(11) NOT NULL default '0',
  visit7 int(11) NOT NULL default '0',
  visit8 int(11) NOT NULL default '0',
  visit9 int(11) NOT NULL default '0',
  visit10 int(11) NOT NULL default '0',
  visit20 int(11) NOT NULL default '0',
  visit30 int(11) NOT NULL default '0',
  visit40 int(11) NOT NULL default '0',
  visit50 int(11) NOT NULL default '0',
  visit60 int(11) NOT NULL default '0',
  visit70 int(11) NOT NULL default '0',
  visit80 int(11) NOT NULL default '0',
  visit90 int(11) NOT NULL default '0',
  visit100 int(11) NOT NULL default '0',
  visitmore int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_deep_month (
  id_client int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  visit1 int(11) NOT NULL default '0',
  visit2 int(11) NOT NULL default '0',
  visit3 int(11) NOT NULL default '0',
  visit4 int(11) NOT NULL default '0',
  visit5 int(11) NOT NULL default '0',
  visit6 int(11) NOT NULL default '0',
  visit7 int(11) NOT NULL default '0',
  visit8 int(11) NOT NULL default '0',
  visit9 int(11) NOT NULL default '0',
  visit10 int(11) NOT NULL default '0',
  visit20 int(11) NOT NULL default '0',
  visit30 int(11) NOT NULL default '0',
  visit40 int(11) NOT NULL default '0',
  visit50 int(11) NOT NULL default '0',
  visit60 int(11) NOT NULL default '0',
```

```
  visit70 int(11) NOT NULL default '0',
  visit80 int(11) NOT NULL default '0',
  visit90 int(11) NOT NULL default '0',
  visit100 int(11) NOT NULL default '0',
  visitmore int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;
CREATE TABLE system_arch_deep_week (
  id_client int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  visit1 int(11) NOT NULL default '0',
  visit2 int(11) NOT NULL default '0',
  visit3 int(11) NOT NULL default '0',
  visit4 int(11) NOT NULL default '0',
  visit5 int(11) NOT NULL default '0',
  visit6 int(11) NOT NULL default '0',
  visit7 int(11) NOT NULL default '0',
  visit8 int(11) NOT NULL default '0',
  visit9 int(11) NOT NULL default '0',
  visit10 int(11) NOT NULL default '0',
  visit20 int(11) NOT NULL default '0',
  visit30 int(11) NOT NULL default '0',
  visit40 int(11) NOT NULL default '0',
  visit50 int(11) NOT NULL default '0',
  visit60 int(11) NOT NULL default '0',
  visit70 int(11) NOT NULL default '0',
  visit80 int(11) NOT NULL default '0',
  visit90 int(11) NOT NULL default '0',
  visit100 int(11) NOT NULL default '0',
  visitmore int(11) NOT NULL default '0',
  PRIMARY KEY  (id_client)
) TYPE=MyISAM;

CREATE TABLE system_arch_time (
  id_time int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  visit1 int(11) NOT NULL default '0',
  visit2 int(11) NOT NULL default '0',
  visit3 int(11) NOT NULL default '0',
  visit4 int(11) NOT NULL default '0',
  visit5 int(11) NOT NULL default '0',
  visit6 int(11) NOT NULL default '0',
  visit7 int(11) NOT NULL default '0',
  visit8 int(11) NOT NULL default '0',
  visit9 int(11) NOT NULL default '0',
  visit10 int(11) NOT NULL default '0',
  visit20 int(11) NOT NULL default '0',
  visit30 int(11) NOT NULL default '0',
  visit40 int(11) NOT NULL default '0',
  visit50 int(11) NOT NULL default '0',
  visit60 int(11) NOT NULL default '0',
  visit2h int(11) NOT NULL default '0',
  visit3h int(11) NOT NULL default '0',
  visit4h int(11) NOT NULL default '0',
  visit5h int(11) NOT NULL default '0',
  visit6h int(11) NOT NULL default '0',
  visit7h int(11) NOT NULL default '0',
  visit8h int(11) NOT NULL default '0',
  visit9h int(11) NOT NULL default '0',
  visit10h int(11) NOT NULL default '0',
  visit11h int(11) NOT NULL default '0',
  visit12h int(11) NOT NULL default '0',
  visit13h int(11) NOT NULL default '0',
```

```
  visit14h int(11) NOT NULL default '0',
  visit15h int(11) NOT NULL default '0',
  visit16h int(11) NOT NULL default '0',
  visit17h int(11) NOT NULL default '0',
  visit18h int(11) NOT NULL default '0',
  visit19h int(11) NOT NULL default '0',
  visit20h int(11) NOT NULL default '0',
  visit21h int(11) NOT NULL default '0',
  visit22h int(11) NOT NULL default '0',
  visit23h int(11) NOT NULL default '0',
  visit24h int(11) NOT NULL default '0',
  PRIMARY KEY  (id_time)
) TYPE=MyISAM;
CREATE TABLE system_arch_time_month (
  id_time int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  visit1 int(11) NOT NULL default '0',
  visit2 int(11) NOT NULL default '0',
  visit3 int(11) NOT NULL default '0',
  visit4 int(11) NOT NULL default '0',
  visit5 int(11) NOT NULL default '0',
  visit6 int(11) NOT NULL default '0',
  visit7 int(11) NOT NULL default '0',
  visit8 int(11) NOT NULL default '0',
  visit9 int(11) NOT NULL default '0',
  visit10 int(11) NOT NULL default '0',
  visit20 int(11) NOT NULL default '0',
  visit30 int(11) NOT NULL default '0',
  visit40 int(11) NOT NULL default '0',
  visit50 int(11) NOT NULL default '0',
  visit60 int(11) NOT NULL default '0',
  visit2h int(11) NOT NULL default '0',
  visit3h int(11) NOT NULL default '0',
  visit4h int(11) NOT NULL default '0',
  visit5h int(11) NOT NULL default '0',
  visit6h int(11) NOT NULL default '0',
  visit7h int(11) NOT NULL default '0',
  visit8h int(11) NOT NULL default '0',
  visit9h int(11) NOT NULL default '0',
  visit10h int(11) NOT NULL default '0',
  visit11h int(11) NOT NULL default '0',
  visit12h int(11) NOT NULL default '0',
  visit13h int(11) NOT NULL default '0',
  visit14h int(11) NOT NULL default '0',
  visit15h int(11) NOT NULL default '0',
  visit16h int(11) NOT NULL default '0',
  visit17h int(11) NOT NULL default '0',
  visit18h int(11) NOT NULL default '0',
  visit19h int(11) NOT NULL default '0',
  visit20h int(11) NOT NULL default '0',
  visit21h int(11) NOT NULL default '0',
  visit22h int(11) NOT NULL default '0',
  visit23h int(11) NOT NULL default '0',
  visit24h int(11) NOT NULL default '0',
  PRIMARY KEY  (id_time)
) TYPE=MyISAM;
CREATE TABLE system_arch_time_temp (
  time_min int(11) NOT NULL default '0',
  putdate datetime NOT NULL default '0000-00-00 00:00:00'
) TYPE=MyISAM;
CREATE TABLE system_arch_time_week (
  id_time int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
```

```
  visit1 int(11) NOT NULL default '0',
  visit2 int(11) NOT NULL default '0',
  visit3 int(11) NOT NULL default '0',
  visit4 int(11) NOT NULL default '0',
  visit5 int(11) NOT NULL default '0',
  visit6 int(11) NOT NULL default '0',
  visit7 int(11) NOT NULL default '0',
  visit8 int(11) NOT NULL default '0',
  visit9 int(11) NOT NULL default '0',
  visit10 int(11) NOT NULL default '0',
  visit20 int(11) NOT NULL default '0',
  visit30 int(11) NOT NULL default '0',
  visit40 int(11) NOT NULL default '0',
  visit50 int(11) NOT NULL default '0',
  visit60 int(11) NOT NULL default '0',
  visit2h int(11) NOT NULL default '0',
  visit3h int(11) NOT NULL default '0',
  visit4h int(11) NOT NULL default '0',
  visit5h int(11) NOT NULL default '0',
  visit6h int(11) NOT NULL default '0',
  visit7h int(11) NOT NULL default '0',
  visit8h int(11) NOT NULL default '0',
  visit9h int(11) NOT NULL default '0',
  visit10h int(11) NOT NULL default '0',
  visit11h int(11) NOT NULL default '0',
  visit12h int(11) NOT NULL default '0',
  visit13h int(11) NOT NULL default '0',
  visit14h int(11) NOT NULL default '0',
  visit15h int(11) NOT NULL default '0',
  visit16h int(11) NOT NULL default '0',
  visit17h int(11) NOT NULL default '0',
  visit18h int(11) NOT NULL default '0',
  visit19h int(11) NOT NULL default '0',
  visit20h int(11) NOT NULL default '0',
  visit21h int(11) NOT NULL default '0',
  visit22h int(11) NOT NULL default '0',
  visit23h int(11) NOT NULL default '0',
  visit24h int(11) NOT NULL default '0',
  PRIMARY KEY  (id_time)
) TYPE=MyISAM;

CREATE TABLE system_arch_refferer (
  id_refferer int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_refferer)
) TYPE=MyISAM;
CREATE TABLE system_arch_refferer_month (
  id_refferer int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_refferer)
) TYPE=MyISAM;
CREATE TABLE system_arch_refferer_week (
  id_refferer int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  PRIMARY KEY  (id_refferer)
) TYPE=MyISAM;
```

```
CREATE TABLE system_arch_searchquery (
  id_searchquery int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  searches enum('yandex','google','rambler','aport','msn') NOT NULL default
'yandex',
  PRIMARY KEY  (id_searchquery)
) TYPE=MyISAM;
CREATE TABLE system_arch_searchquery_month (
  id_searchquery int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  searches enum('yandex','google','rambler','aport','msn') NOT NULL default
'yandex',
  PRIMARY KEY  (id_searchquery)
) TYPE=MyISAM;
CREATE TABLE system_arch_searchquery_week (
  id_searchquery int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  total int(11) NOT NULL default '0',
  searches enum('yandex','google','rambler','aport','msn') NOT NULL default
'yandex',
  PRIMARY KEY  (id_searchquery)
) TYPE=MyISAM;
CREATE TABLE system_arch_num_searchquery (
  id_num_searchquery int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  number_yandex int(11) NOT NULL default '0',
  number_google int(11) NOT NULL default '0',
  number_rambler int(11) NOT NULL default '0',
  number_aport int(11) NOT NULL default '0',
  number_msn int(11) NOT NULL default '0',
  number_mail int(11) NOT NULL default '0',
  PRIMARY KEY  (id_num_searchquery)
) TYPE=MyISAM;
CREATE TABLE system_arch_num_searchquery_month (
  id_num_searchquery int(11) NOT NULL auto_increment,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  number_yandex int(11) NOT NULL default '0',
  number_google int(11) NOT NULL default '0',
  number_rambler int(11) NOT NULL default '0',
  number_aport int(11) NOT NULL default '0',
  number_msn int(11) NOT NULL default '0',
  number_mail int(11) NOT NULL default '0',
  PRIMARY KEY  (id_num_searchquery)
) TYPE=MyISAM;
CREATE TABLE system_arch_num_searchquery_week (
  id_num_searchquery int(11) NOT NULL auto_increment,
  putdate_begin datetime NOT NULL default '0000-00-00 00:00:00',
  putdate_end datetime NOT NULL default '0000-00-00 00:00:00',
  number_yandex int(11) NOT NULL default '0',
  number_google int(11) NOT NULL default '0',
  number_rambler int(11) NOT NULL default '0',
  number_aport int(11) NOT NULL default '0',
  number_msn int(11) NOT NULL default '0',
  number_mail int(11) NOT NULL default '0',
  PRIMARY KEY  (id_num_searchquery)
) TYPE=MyISAM;
CREATE TABLE `system_cities` (
  `city_id` int(11) NOT NULL default '0',
```

```
  `region_id` int(11) NOT NULL default '0',
  `city_name` varchar(255) NOT NULL default ''
) TYPE=MyISAM;
CREATE TABLE `system_ip_compact` (
  `init_ip` bigint(20) NOT NULL default '0',
  `end_ip` bigint(20) NOT NULL default '0',
  `city_id` int(11) NOT NULL default '0'
) TYPE=MyISAM;
INSERT INTO `system_ip_compact` VALUES (1040547840, 1040553455, 1);
INSERT INTO `system_ip_compact` VALUES (1040553456, 1040553463, 0);
INSERT INTO `system_ip_compact` VALUES (1040553464, 1040580607, 1);
INSERT INTO `system_ip_compact` VALUES (1041244160, 1041252351, 20);
INSERT INTO `system_ip_compact` VALUES (1041252352, 1041252991, 1);
CREATE TABLE `system_regions` (
  `region_id` int(11) NOT NULL default '0',
  `region_name` varchar(255) NOT NULL default ''
) TYPE=MyISAM;
INSERT INTO `system_cities` VALUES (1, 77, 'Ternopil');
INSERT INTO `system_cities` VALUES (2, 78, 'Kyiv');
INSERT INTO `system_cities` VALUES (3, 54, 'Lviv');
INSERT INTO `system_cities` VALUES (4, 25, 'Khmelnytsky');

CREATE TABLE `system_ip_compact` (
  `init_ip` bigint(20) NOT NULL default '0',
  `end_ip` bigint(20) NOT NULL default '0',
  `city_id` int(11) NOT NULL default '0'
) TYPE=MyISAM;
INSERT INTO `system_ip_compact` VALUES (1040547840, 1040553455, 1);
INSERT INTO `system_ip_compact` VALUES (1040553464, 1040580607, 2);
INSERT INTO `system_ip_compact` VALUES (1041252352, 1041252991, 3);
INSERT INTO `system_ip_compact` VALUES (1041253056, 1041259007, 4);
CREATE TABLE `system_regions` (
  `region_id` int(11) NOT NULL default '0',
  `region_name` varchar(255) NOT NULL default ''
) TYPE=MyISAM;
INSERT INTO `system_regions` VALUES (0, 'Kyiv');
INSERT INTO `system_regions` VALUES (1, 'Kyivska oblast');
INSERT INTO `system_regions` VALUES (2, 'Ternopilska oblast');
INSERT INTO `system_regions` VALUES (3, 'USA');
```

Програмні коди сценаріївлічильника відвідувань сайту

Count.php

```
       <?php
           Error_Reporting(E_ALL & ~E_NOTICE);

         // Встановлюємо з'єднання з базою даних
           require_once("config.php");

         //перевірка що шлях закінчується index.php
           if(substr($_SERVER['PHP_SELF'], strlen($_SERVER['PHP_SELF']) - 1) ==
"/") $_SERVER['PHP_SELF'] .= "index.php";

         // Змінна для посилання на сторінку для якої проводиться підрахунок
         // Формуємо стрічку з ip
           if($obtip == 0) $ip = $_SERVER['REMOTE_ADDR']; // За умовчанням
           if($obtip == 1) $ip = urldecode(getenv(HTTP_CLIENTIP));
           if(empty($ip)) $ip = '0.0.0.0';

         $  $tbl_ip       = 'system_ip';
         $  $tbl_pages    = 'system_pages';
         $  $tbl_refferer = 'system_refferer';
         $  $tbl_searchquerys = 'system_searchquerys';

         // Це стрічка з реферером - URL сторінки, з якою відвідувач прийшов на
         // сайт
         $  $reff = urldecode($_SERVER['HTTP_REFERER']);
         //   // З'єднуємося з сервером бази даних
         $  $dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
           if($dbcnx)
           {
           // Вибираємо базу даних
               if(@mysql_select_db($dbname,$dbcnx))
               {
                   if(empty($titlepage)) $titlepage =
"http://".$_SERVER['SERVER_NAME'].$_SERVER['PHP_SELF'];
                   if (!get_magic_quotes_gpc()) $titlepage =
mysql_escape_string($titlepage);
               $query = "SELECT id_page FROM $tbl_pages
                                    WHERE title = '$titlepage'";
               $pgs = mysql_query($query);
                   if ($pgs)
                   {
               // З'ясуємо, первинний ключ (id_page) поточної сторінки  (по
назві сторінки)
                       if(mysql_num_rows($pgs)>0) $id_page = mysql_result($pgs,
0);
               // Якщо назва цієї сторінки відсутня в таблиці pages
               // те перевіряємо сторінку за її адресою.
                       else
                       {
                       $query = "SELECT id_page FROM $tbl_pages
                                           WHERE
name='".$_SERVER['PHP_SELF']."'";
                       $pgs = mysql_query($query);
                           if ($pgs)
                           {
                   // З'ясуємо, первинний ключ (id_page) поточної сторінки (за
адресою сторінки)
                               if(mysql_num_rows($pgs)>0)
                               {
```

```
                                $id_page = mysql_result($pgs, 0);
                                $query = "UPDATE $tbl_pages SET title='$titlepage'
WHERE id_page=$id_page";
                                  mysql_query($query);
                                }
                    // Якщо ця сторінка відсутня в таблиці pages
                    // і ні разу не враховувалася - додаємо цю сторінку в
таблицю.
                                else
                                {
                                $query = "INSERT INTO $tbl_pages VALUES (NULL,
'".$_SERVER['PHP_SELF']."','$titlepage ', 0)";
                                    @mysql_query($query);
                      // З'ясовуємо первинний ключ тільки що доданою
                      // сторінки
                                $id_page = mysql_insert_id();
                                }
                            }
                          }
                    }
              // Визначаємо рядок USER_AGENT
                $useragent = $_SERVER['HTTP_USER_AGENT'];
                $browser = 'none';
              // З'ясовуємо браузер
                    if(strpos($useragent, "Mozilla")!== false) $browser =
'mozilla';
                    if(strpos($useragent, "MSIE") !== false) $browser = 'msie';
                    if(strpos($useragent, "MyIE") !== false) $browser = 'myie';
                    if(strpos($useragent, "Opera") !== false) $browser =
'opera';
                    if(strpos($useragent, "Netscape")!== false) $browser =
'netscape';
                    if(strpos($useragent, "Firefox")!== false) $browser =
'firefox';
              // З'ясовуємо операційну систему
                $os = 'none';
                    if(strpos($useragent, "Win") !== false) $os = 'windows';
                    if(strpos($useragent, "Linux") !== false
                    || strpos($useragent, "Lynx") !== false
                    || strpos($useragent, "Unix") !== false) $os = 'unix';
                    if(strpos($useragent, "Macintosh")!== false) $os =
'macintosh';
                    if(strpos($useragent, "PowerPC") !== false) $os =
'macintosh';
              // З'ясовуємо приналежність до пошукових роботів
                    if(strpos($useragent, "StackRambler")!== false) $os =
'robot_rambler';
                    if(strpos($useragent, "Googlebot") !== false) $os =
'robot_google';
                    if(strpos($useragent, "Mediapartners-Google")   !== false)
$os = 'robot_google';
                    if(strpos($useragent, "Yandex") !== false) $os =
'robot_yandex';
                    if(strpos($useragent, "Aport") !== false) $os =
'robot_aport';
                    if(strpos($useragent, "msnbot") !== false) $os =
'robot_msnbot';
            $search = 'none';
              // З'ясовуємо приналежність до пошукових систем
                    if(strpos($reff,"yandex"))  $search = 'yandex';
                    if(strpos($reff,"rambler")) $search = 'rambler';
                    if(strpos($reff,"google"))  $search = 'google';
                    if(strpos($reff,"aport"))   $search = 'aport';
```

```
                    if(strpos($reff,"mail") && strpos($reff,"search"))   $search
= 'mail';
                    if(strpos($reff,"msn") && strpos($reff,"results"))   $search
= 'msn';
            $server_name = $_SERVER["SERVER_NAME"];
                    if(substr($_SERVER["SERVER_NAME"],0,4) == "www.")
$server_name = substr($_SERVER["SERVER_NAME"],4);
                    if(strpos($reff,$server_name)) $search = 'own_site';

              // Заносимо усю зібрану інформацію в базу даних
                $query_main = "INSERT INTO $tbl_ip VALUES (
                                        NULL
                                        INET_ATON('$ip')
                                        NOW()
                                    $id_page
                          '              '$browser
                          '              '$os')";
                    @mysql_query($query_main);
              // Якщо є реферер, заносимо інформацію про нього в окрему таблицю
                    if(!empty($reff) && $search=="none")
                    {
                    $reff = str_replace("'","'",$reff);
                    $query_reff = "INSERT INTO $tbl_refferer VALUES (
                                        NULL
                         '              '$reff
                                        now()
                                        INET_ATON('$ip')
                                    $id_page)";
                       @mysql_query($query_reff);
                    }
            //вноситься пошуковий запит у відповідну таблицю
                if(!empty($reff) && $search!="none" && $search != "own_site")
                {
                      switch($search)
                      {
                          case 'yandex ':
                          {
                                  eregi("text=([^&]*) ", $reff."&", $query);
                                  if(strpos($reff,"yandpage")!=null)
                                  $quer=convert_cyr_string(urldecode($query[1]),
"k","w");
                                  else
                                  $quer=$query[1];
                              break;
                          }
                          case 'rambler ':
                          {
                              eregi("words=([^&]*) ", $reff."&", $query);
                          $quer = $query[1];
                              break;
                          }
                          case 'mail ':
                          {
                              eregi("q=([^&]*) ", $reff."&", $query);
                          $quer = $query[1];
                              break;
                          }
                          case 'google ':
                          {
                              eregi("q=([^&]*) ", $reff."&", $query);
                          $quer = utf8_win($query[1]);
                              break;
                          }
                          case 'msn ':
```

```
                        {
                            eregi("q=([^&]*) ", $reff."&", $query);
                        $quer = utf8_win($query[1]);
                            break;
                        }
                        case 'aport ':
                        {
                            eregi("r=([^&]*) ", $reff."&", $query);
                        $quer = $query[1];
                            break;
                        }
                       }//кінець для switch
                          $symbols = array("\"", "'", "(", ")", "+", ",", "-
");
                    $quer = str_replace($symbols, " ", $quer);
                    $quer = trim($quer);
                    $quer = preg_replace('|[\s]+|',' ',$quer);
                $sql="INSERT INTO $tbl_searchquerys VALUES (NULL, '$quer ',
now(), INET_ATON('$ip'), $id_page, '$search')";
                        @mysql_query($sql);
                  }
              }
          }
        function utf8_win($s)
        {
        $s=str_replace("\xD0\xB0","а",$s);  $s=str_replace("\xD0\x90","А",$s);
        $s=str_replace("\xD0\xB1","б",$s);  $s=str_replace("\xD0\x91","Б",$s);
        $s=str_replace("\xD0\xB2","в",$s);  $s=str_replace("\xD0\x92","В",$s);
        $s=str_replace("\xD0\xB3","г",$s);  $s=str_replace("\xD0\x93","Г",$s);
        $s=str_replace("\xD0\xB4","д",$s);  $s=str_replace("\xD0\x94","Д",$s);
        $s=str_replace("\xD0\xB5","е",$s);  $s=str_replace("\xD0\x95","Е",$s);
        $s=str_replace("\xD1\x91","е",$s);  $s=str_replace("\xD0\x81","Е",$s);
        $s=str_replace("\xD0\xB6","ж",$s);  $s=str_replace("\xD0\x96","Ж",$s);
        $s=str_replace("\xD0\xB7","з",$s);  $s=str_replace("\xD0\x97","З",$s);
        $s=str_replace("\xD0\xB8","і",$s);  $s=str_replace("\xD0\x98","І",$s);
        $s=str_replace("\xD0\xB9","й",$s);  $s=str_replace("\xD0\x99","Й",$s);
        $s=str_replace("\xD0\xBA","к",$s);  $s=str_replace("\xD0\x9A","К",$s);
        $s=str_replace("\xD0\xBB","л",$s);  $s=str_replace("\xD0\x9B","Л",$s);
        $s=str_replace("\xD0\xBC","м",$s);  $s=str_replace("\xD0\x9C","М",$s);
        $s=str_replace("\xD0\xBD","н",$s);  $s=str_replace("\xD0\x9D","Н",$s);
        $s=str_replace("\xD0\xBE","о",$s);  $s=str_replace("\xD0\x9E","О",$s);
        $s=str_replace("\xD0\xBF","п",$s);  $s=str_replace("\xD0\x9F","П",$s);
        $s=str_replace("\xD1\x80","р",$s);  $s=str_replace("\xD0\xA0","Р",$s);
        $s=str_replace("\xD1\x81","з",$s);  $s=str_replace("\xD0\xA1","З",$s);
        $s=str_replace("\xD1\x82","т",$s);  $s=str_replace("\xD0\xA2","Т",$s);
        $s=str_replace("\xD1\x83","у",$s);  $s=str_replace("\xD0\xA3","У",$s);
        $s=str_replace("\xD1\x84","ф",$s);  $s=str_replace("\xD0\xA4","Ф",$s);
        $s=str_replace("\xD1\x85","х",$s);  $s=str_replace("\xD0\xA5","Х",$s);
        $s=str_replace("\xD1\x86","ц",$s);  $s=str_replace("\xD0\xA6","Ц",$s);
        $s=str_replace("\xD1\x87","ч",$s);  $s=str_replace("\xD0\xA7","Ч",$s);
        $s=str_replace("\xD1\x88","ш",$s);  $s=str_replace("\xD0\xA8","Ш",$s);
        $s=str_replace("\xD1\x89","щ",$s);  $s=str_replace("\xD0\xA9","Щ",$s);
          $s=str_replace("\xD1\x8A","ъ",$s);  $s=str_replace("\xD0\xAA","Ъ",$s);
        $s=str_replace("\xD1\x8B","ы",$s);  $s=str_replace("\xD0\xAB","Ы",$s);
        $s=str_replace("\xD1\x8C","ь",$s);  $s=str_replace("\xD0\xAC","Ь",$s);
        $s=str_replace("\xD1\x8D","э",$s);  $s=str_replace("\xD0\xAD","Э",$s);
        $s=str_replace("\xD1\x8E","ю",$s);  $s=str_replace("\xD0\xAE","Ю",$s);
        $s=str_replace("\xD1\x8F","я",$s);  $s=str_replace("\xD0\xAF","Я",$s);
                return $s;
        }
        ?>
```

Config.php

```
        <?php
```

```
        // Основні змінні
        // Ім'я сервера бази даних, наприклад
        // $dblocation = "mysql 28.noweb.ru"
        // зараз виставлений сервер локальної машини
        $dblocation = "localhost";
        //   // Ім'я бази даних, на хостингу або локальній машині
        $dbname = "count";
        // Ім'я користувача бази даних
        $dbuser = "root";
        // і його пароль
        $dbpasswd = "pGf8E34";
        // Спосіб визначення IP -адреса відвідувача
        // 0 Підходить для більшості хостингів у тому числі
        // для використання на локальній машині
        $obtip = 0;
        // 1 На деяких хостингах ip -адрес відвідувача не
        // заноситься в змінну $REMOTE_ADDR, до них відноситься
        // наприклад www.nodex.ru
        // $obtip = 1;
        // Поточна версія системи
          if($version=@file("version.txt"))
              $version = $version[0];
          else $version = 0;
      ?>
```

Archive.php

```
<?php
    set_time_limit(0);

    Error_Reporting(E_ALL & ~E_NOTICE);
  // Встановлюємо з'єднання з базою даних
    require_once("config.php");
  // Бібліотека функцій архівації
    require_once("utils_hits.php");
    require_once("utils_ip.php");
    require_once("utils_client.php");
    require_once("utils_robots.php");
    require_once("utils_enterpoints.php");
    require_once("utils_deep.php");
    require_once("utils_time.php");
    require_once("utils_refferer.php");
    require_once("utils_num_search.php");
    require_once("utils_search.php");
  // Отримуємо дату початку реєстрації даних і число що пройшли з початку
реєстрації
  $dat = mysql_query("SELECT UNIX_TIMESTAMP(MIN(putdate)) AS data FROM
$tbl_ip");
    if (!$dat) exit(mysql_error());
  $arr_date = mysql_fetch_array($dat);
  // Останій повний день
  $last_day = mktime(23,59,59, date("m"), date("d") - 1, date("Y"));
   // Отримуємо останню дату архівації
  $query = "SELECT UNIX_TIMESTAMP(MAX(putdate)) FROM $tbl_arch_hits";
  $arh = mysql_query($query);
    if(mysql_num_rows($arh)> 0)
    {
    $last_date = mysql_result($arh, 0);
    }
    if(empty($last_date))
    {
        if(isset($arr_date['data']))
        {
        // Якщо запуск перший - беремо дату з $tbl_searchquerys
      $last_date = $arr_date['data'];
```

```
        }
        else
        {
        // Інакше беремо поточну добу
      $last_date = time();
        }
    }
   //блок архівації
    if (ceil(($last_day - $last_date)/24/60/60))
    {
    ////////////////////////////////////////////////
    // Архівуємо інформацію в щоденні таблиці
    ////////////////////////////////////////////////
        archive_ip              ($tbl_ip, $tbl_arch_ip);
        archive_client          ($tbl_ip, $tbl_arch_clients);
        archive_robots          ($tbl_ip, $tbl_arch_robots);
        archive_enterpoints     ($tbl_ip, $tbl_pages, $tbl_arch_enterpoint);
        archive_deep            ($tbl_ip, $tbl_arch_deep);
        archive_time            ($tbl_ip, $tbl_arch_time, $tbl_arch_time_temp);
        archive_refferer        ($tbl_refferer, $tbl_arch_refferer);
        archive_searchquery     ($tbl_searchquerys, $tbl_arch_searchquery);
        archive_num_searchquery ($tbl_searchquerys, $tbl_arch_num_searchquery);
        archive_hit_hosts       ($tbl_ip, $tbl_arch_hits);
    ////////////////////////////////////////////////
    // Видаляємо старі записи
    ////////////////////////////////////////////////
    $query = "SELECT MAX(putdate) FROM $tbl_arch_hits";
    $arh = mysql_query($query);
        if(!$arh) exit("Збій при видаленні старих записів");
        if(mysql_num_rows($arh)> 0)
        {
      $last_date_arch = mysql_result($arh, 0);
      $arr[] = "DELETE FROM $tbl_ip WHERE putdate <= '$last_date_arch' -
INTERVAL 30 DAY";
      $arr[] = "DELETE FROM $tbl_refferer WHERE putdate <= '$last_date_arch' -
INTERVAL 30 DAY";
      $arr[] = "DELETE FROM $tbl_searchquerys WHERE putdate <=
'$last_date_arch' - INTERVAL 30 DAY";
            foreach($arr as $query) if(!mysql_query($query))
exit(mysql_error());
        }
    ////////////////////////////////////////////////
    // Архівуємо інформацію в щотижневі таблиці
    ////////////////////////////////////////////////
        archive_hit_hosts_week       ($tbl_arch_hits, $tbl_arch_hits_week);
        archive_ip_week              ($tbl_arch_ip, $tbl_arch_ip_week);
        archive_client_week          ($tbl_arch_clients,
$tbl_arch_clients_week);
        archive_robots_week          ($tbl_arch_robots, $tbl_arch_robots_week);
        archive_enterpoints_week     ($tbl_arch_enterpoint,
$tbl_arch_enterpoint_week);
        archive_deep_week            ($tbl_arch_deep, $tbl_arch_deep_week);
        archive_time_week            ($tbl_arch_time, $tbl_arch_time_week);
        archive_refferer_week        ($tbl_arch_refferer,
$tbl_arch_refferer_week);
        archive_searchquery_week     ($tbl_arch_searchquery,
$tbl_arch_searchquery_week);
        archive_num_searchquery_week ($tbl_arch_num_searchquery,
$tbl_arch_num_searchquery_week);

    ////////////////////////////////////////////////
    // Архівуємо інформацію в еженемесячные таблиці
    ////////////////////////////////////////////////
        archive_hit_hosts_month       ($tbl_arch_hits, $tbl_arch_hits_month);
```

```
        archive_ip_month              ($tbl_arch_ip, $tbl_arch_ip_month);
        archive_clients_month         ($tbl_arch_clients,
$tbl_arch_clients_month);
        archive_robots_month          ($tbl_arch_robots,
$tbl_arch_robots_month);
        archive_enterpoints_month     ($tbl_arch_enterpoint,
$tbl_arch_enterpoint_month);
        archive_deep_month            ($tbl_arch_deep, $tbl_arch_deep_month);
        archive_time_month            ($tbl_arch_time, $tbl_arch_time_month);
        archive_refferer_month        ($tbl_arch_refferer,
$tbl_arch_refferer_month);
        archive_searchquery_month     ($tbl_arch_searchquery,
$tbl_arch_searchquery_month);
        archive_num_searchquery_month ($tbl_arch_num_searchquery,
$tbl_arch_num_searchquery_month);
  }
?>
```