МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ДАВИДЮК Андрій Олександрович

Програмний модуль для організації спілкування між користувачами соціальних мереж/ Software module for communication organizinf between users of social networks

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
А. О. Давидюк

_____

Науковий керівник:
викладач ЛИСЕНКО О.О.

_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

ЗМІСТ

ВСТУП

Мабуть, всім відомі імена таких web-сайтів, як «Facebook» та «VK». Ці ресурси, це так звані «соціальні мережи», є найбільш відвідуваними сайтами в українському сегменті мережі Інтернет. Їх аудиторія становить десятки мільйонів людей, трафік обчислюється терабайтами, а кількість серверів - сотнями.

Ці ресурси виконують функцію соціальної взаємодії людей в мережі Інтернет. Серед їхніх можливостей можна виділити ведення власного профілю (інформація про користувача, місцях його навчання, роботи, і. т.д.), додавання інших користувачів в список контактів, додавання фотографій в особистий фотоальбом та перегляд альбомів інших учасників, завантаження файлів різних типів, таких як відеофайли, аудіофайли, можливість перегляду відео і прослуховування аудіокомпозицій, ведення особистого блогу, і багато інших функцій.

У зв'язку з розвитком нової ери в світі Web, що носить назву Web 2.0, появляється велика кількість подібних проектів. Щорічна виручка компанії-власника Facebook.com оцінюється в 300 мільйонів доларів. Однак, незважаючи на популярність існуючих соціальних мереж, в кожній з них є свої недоліки. Причому недоліки є як у функціональності (недостатня гнучкість налаштувань профілю користувача, мала кількість доступних сервісів або їх платний статус), так і в зручності взаємодії користувача з системою. Тому актуальною є ідея створення засобу спілкування між користувачами соціальної мережі нового покоління. Метою цієї роботи є створення функціонального і зручно для користувача засобу спілкування соціальної мережі, який за своїми експлуатаційними якостями зміг б успішно конкурувати з існуючими проектами подібного роду.

Принциповою особливістю проекту є орієнтація на відкрите програмне забезпечення. Програмування модулів системи велося на мові програмування

PHP з використанням платформи web-розробки Symfony компанії Sensio Labs. В якості СУБД була обрана система MySQL 5.5.

В даний час web-додаток встановлено на сервер компанії-замовника і проходить закрите тестування. З метою отримання порівняльних характеристик системи була проведена порівняльна оцінка додатку з залученням сторонніх експертів, в якості яких виступали 10 осіб з різних соціальних груп.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПІДТРИМКИ ПРОЦЕСІВ ОРГАНІЗАЦІЇ СПІЛКУВАННЯ МІЖ КОРИСТУВАЧАМИ СОЦІАЛЬНИХ МЕРЕЖ

1.1. Види та функції соціальних мереж

Соціальная мережа - платформа, онлайн-сервіс або веб-сайт, призначений для побудови, відображення та організації соціальних взаємовідносин, візуалізацією яких являються соціальні графи [2]. До закритих соціальних мереж відносяться: відповідні за певними умовами або сайти з обмеженою кількістю людей. Одна зі звичайних рис соціальних мереж - система «друзів» і «груп».

Характерними особливостями соціальної мережі є:

- створення особистих профілів (публічних або напівпублічних), в яких найчастіше потрібно вказати реальні персональні дані і іншу інформацію про себе (місце навчання і роботи, хобі, життєві принципи та ін.);

- надання практично повного спектра можливостей для обміну інформацією (розміщення фотографій, відео-записів, розміщення текстових записів (в режимі блогів або мікроблогів), організація тематичних співтовариств, обмін особистими повідомленнями і т.д.);

- можливість задавати і підтримувати список інших користувачів, з якими у нього є деякі відношення (наприклад, дружби, спорідненості, ділових і робочих зв'язків і т. д.) [2].

Тому помилково вважати соціальними мережами такі сервіси як LiveJournal (майданчик блогів), foursquare, Twitter (майданчики мікроблогів), так як вони мають досить вузький спектр можливостей.

З розвитком технологій Web 2.0 соціальні мережі здобули відчутну основу у вигляді порталів і веб-сервісів. Виникла ідея створити перші соціальні мережі.

Переможний хід по Інтернету соціальні мережі почали в 1995 році з американського порталу Classmates.com. «Однокласники» є його аналогом. Проект виявився досить успішним, тому що в наступні кілька років спровокувало появу не одного десятка аналогічних сервісів. Але офіційним початком буму соціальних мереж прийнято вважати 2003-2004 роки, коли були запущені LinkedIn, MySpace і Facebook.

І якщо LinkedIn створювалася з метою встановлення/підтримки ділових контактів, то власники MySpace і Facebook зробили ставку в першу чергу на задоволення людської потреби в самовираженні. Адже, відповідно до піраміди Маслоу, сааме самовираження є вищою потребою людини, випереджаючи навіть визнання і спілкування [2].

Кожен з нас проводить якусь кількість часу в соціальних мережах. Це життя більшості людей. Хочеться подивитися нові фотографії своїх друзів, написати що-небудь на стіні, поспілкуватися.

Соціальні мережі настільки глобальні, що важко осягнути це розумом, в них мешкають мільйони людей. Серед них можна і потрібно навчитися відбирати потенційних клієнтів і партнерів. Для цього треба поставити собі метц - створити поступово своє, так зване «плем'я», тобто своє коло спілкування. Подружитися з людьми, зарекомендувати себе як експерта, давати людям те, що вони хочуть і таким чином мотивувати до переходу на свій сайт і просувати свою інформацію. Розглянемо докладніше, що таке «Спільнота» [1].

Згодом читачі і передплатники, які отримують від вас цінну інформацію стають для вас кругом номер один. Ви написали для них безліч статей, провели якусь кількість вебінарів протягом від 30 днів до півроку. За цей час виникає довіра і зав'язуються теплі відносини.

Наше завдання полягає в тому, щоб все більше утеплювати коло спілкування і поступово розширювати його. І тоді ви вже зможете просувати якийсь свій чи чужий товар (партнерські програми) зі свого сайту, ці люди будуть готові купити товар або ваш, або по вашій рекомендації.

Відповідно, якщо люди вам вірять, то вам необхідно не підводити їх. Тоді своїми правильними діями, кроками ви сформуєте маркетингову мережу споживачів. І навіть, якщо на перших порах ви не є експертом в тій області, в якій просуваєте свій сайт, ви поступово ним станете, так розбираєтеся в усьому досконально. І не можна недооцінювати свою цінність.

Регулярно треба влаштовувати опитування на своєму сайті, щоб мати зворотний зв'язок, знати, що людям треба і розуміти, що ви йдете в правильному напрямку. Мотивувати при цьому до переходу на свій сайт, показати, що у вас цікаво, кожен раз подати щось нове [1].

Чому ж для цих цілей дуже непогано підходять соціальні мережі. Справа в тому, що тут і проявляються чітко функції соціальних мереж:

- Глобально і перспективно;

- Безкоштовно або дуже дешево;

- Чітке виділення цільової аудиторії (ЦА) - за віком, інтересам і так далі;

- Можливість контактувати з ким завгодно - подружитися з відомою людиною і взяти у нього інтерв'ю, або опублікувати його статтю на своєму сайті.

Ця співпраця може бути вигідною вам обом. Але найважливіше, що потрібно розуміти - цей метод, так скажемо, довгограючий. Він не приносить результату відразу. Але це досить надійний метод в перспективі і йому треба приділяти увагу по ходу роботи.

Тобто прийшли в соціальну мережу, зробили якісь механічні дії (додали друзів, наприклад) і пішли, не затримуючись там надовго. Тому як на то вони і мережі, щоб затягувати. І через якийсь час - кілька місяців - результат буде дуже цікавим. У вас будуть вже тисячі друзів. Написавши статтю, ви натискаєте кнопочку «опублікувати в Vkontakte, наприклад, і послати в Facebook і отримуєте масу читачів. А Twitter - найчисленніша соціальна мережа. Соціальні мережі стали свого роду Інтернет-притулком, де кожен може закачати технічну і соціальну базу для створення свого віртуального «Я». При цьому кожен

користувач отримав можливість не просто спілкуватися і творити, але і ділитися плодами своєї творчості з багатомільйонною аудиторією тієї чи іншої соціальної мережі.

Найбільші соціальні мережі. У різних регіонах світу популярність різних соціальних мереж варіюється. Так, мережі MySpace, Facebook, Twitter і LinkedIn більш популярні і поширені в Північній Америці.

Англомовні соціальні мережі:

- Facebook (є український інтерфейс, рисунок 1.1);
- Twitter (є український інтерфейс, рисунок 1.2);
- LinkedIn (є український інтерфейс, рисунок 1.3);
- MySpace (є український інтерфейс, рисунок 1.4);
- XING (є український інтерфейс, рисунок 1.5);
- Google+ (є український інтерфейс, рисунок 1.6);

Російськомовні соціальні мережі:

- В Контакте, рисунок 1.7;
- Одноклассники.ru.

Інші мережі:

- Nexopia (Канада);
- Bebo (Великобританія);
- Facebook, Hi5, dol2day (Німеччина);
- Tagged.com (англ.), XING і Skyrock (в різних країнах Європи);
- Public Broadcasting Service, Orkut, Facebook і Hi5 (в Південній і Центральній Америці) (55% бразильських користувачів мереж воліє Orkut);
- Friendster, Multiply, Orkut, Xiaonei і Cyworld (Азія: Філіппіни, Малайзія, Індонезія, Сінгапур).

За кількістю користувачів лідирують:

- Facebook (750 000 000);
- MySpace (255 000 000);
- Twitter (200 000 000);

- В Контакте (150 000 000);
- Windows Live Spaces (120 000 000);
- Habbo Hotel (121 000 000);
- Friendster (90 000 000);
- Hi5 (80 000 000);
- Tagged.com (70 000 000).

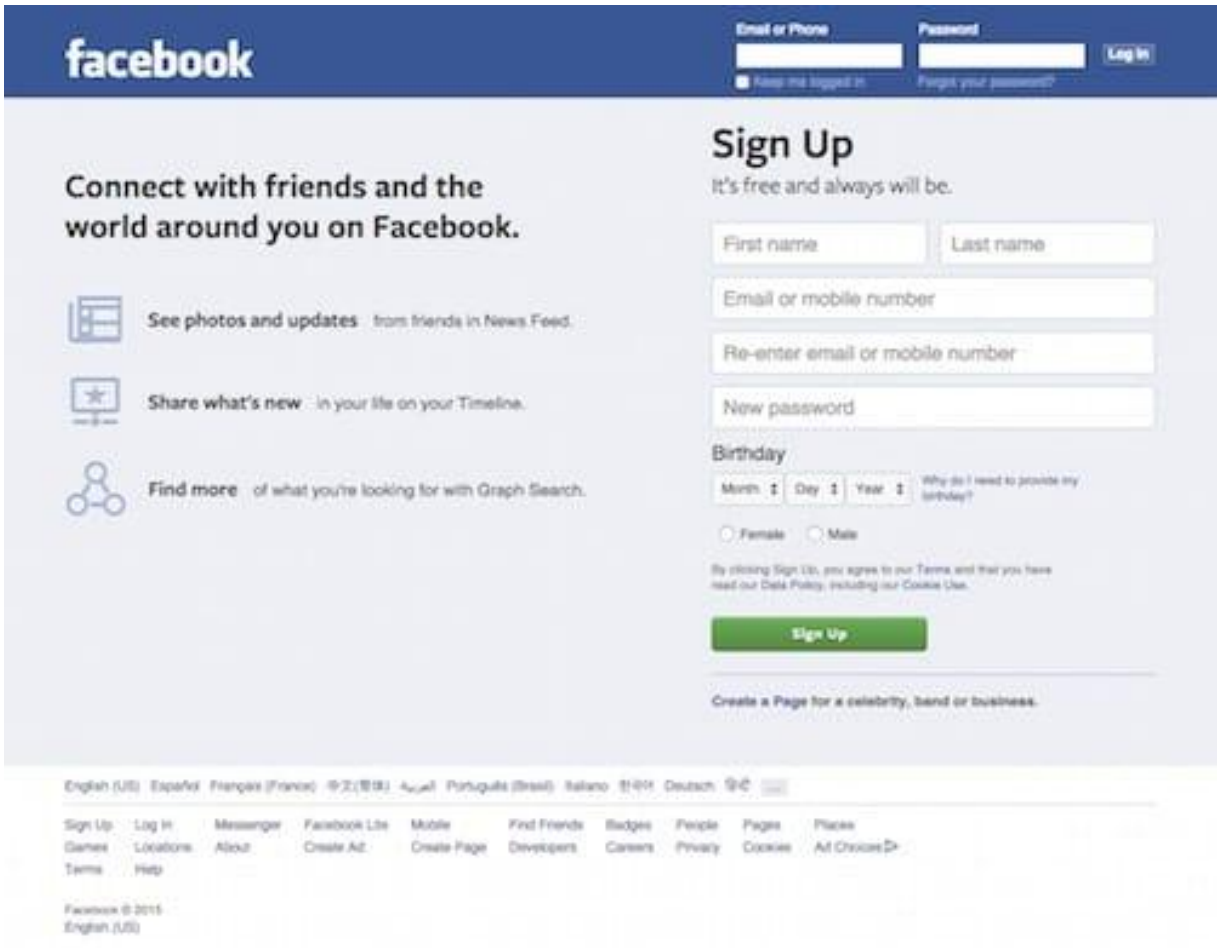

Рис. 1.1. Facebook

Рис. 1.2. Twitter

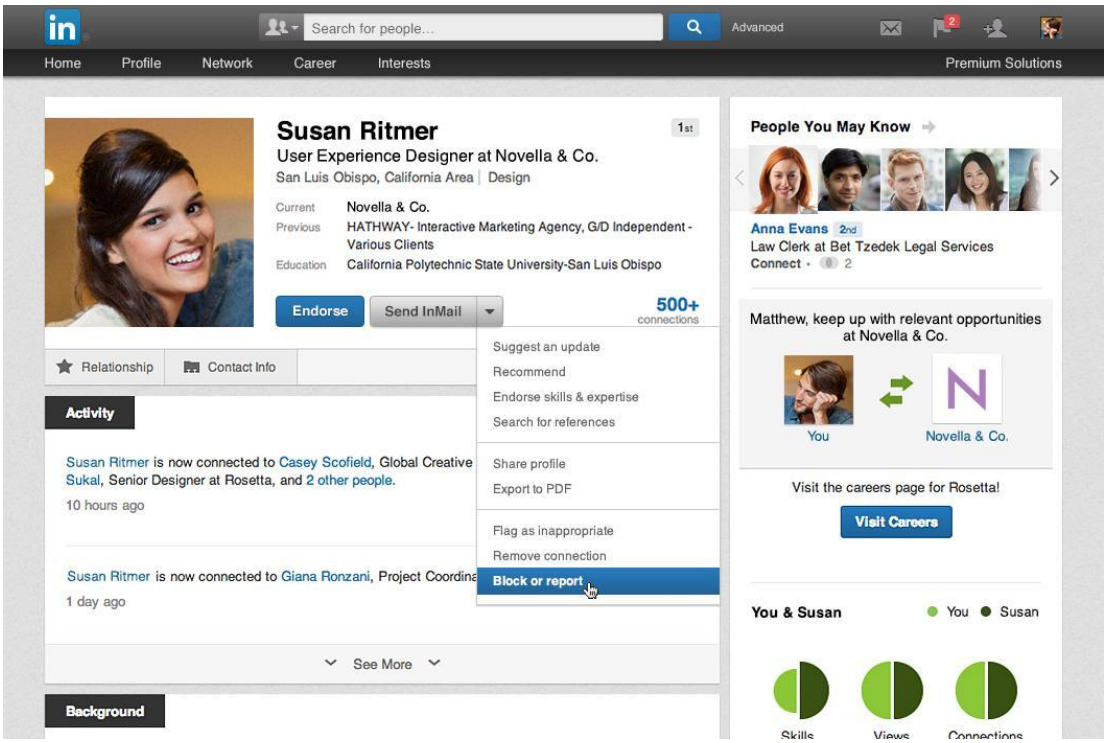

Рис. 1.3. LinkedIn

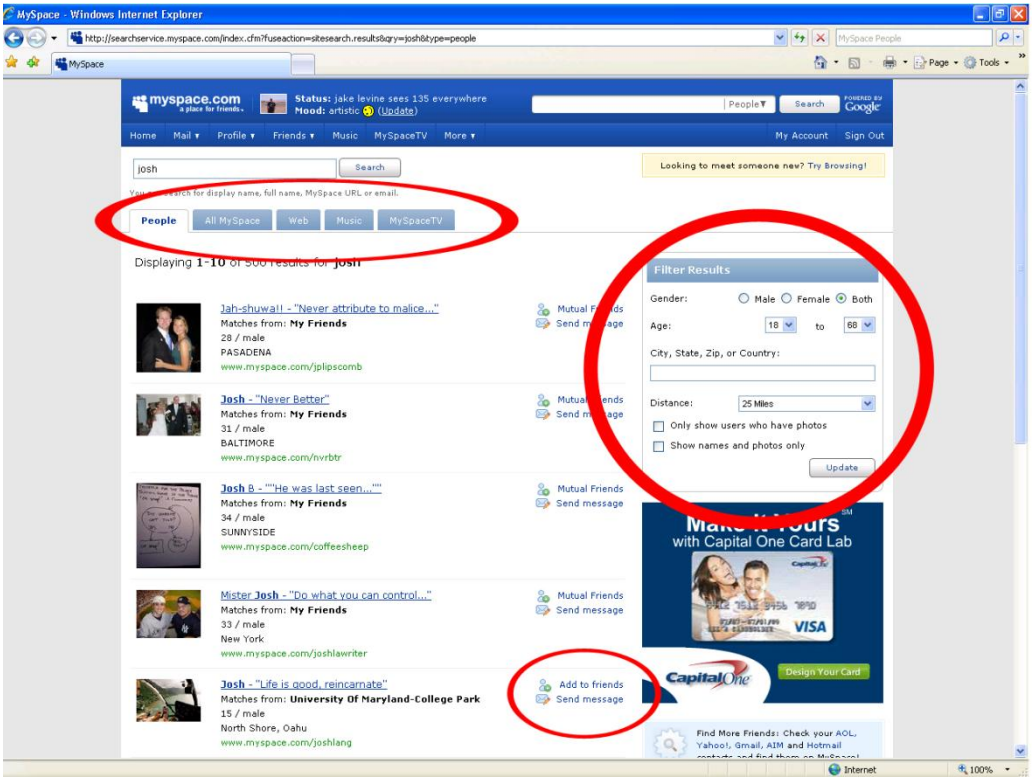

Рис. 1.4. MySpace

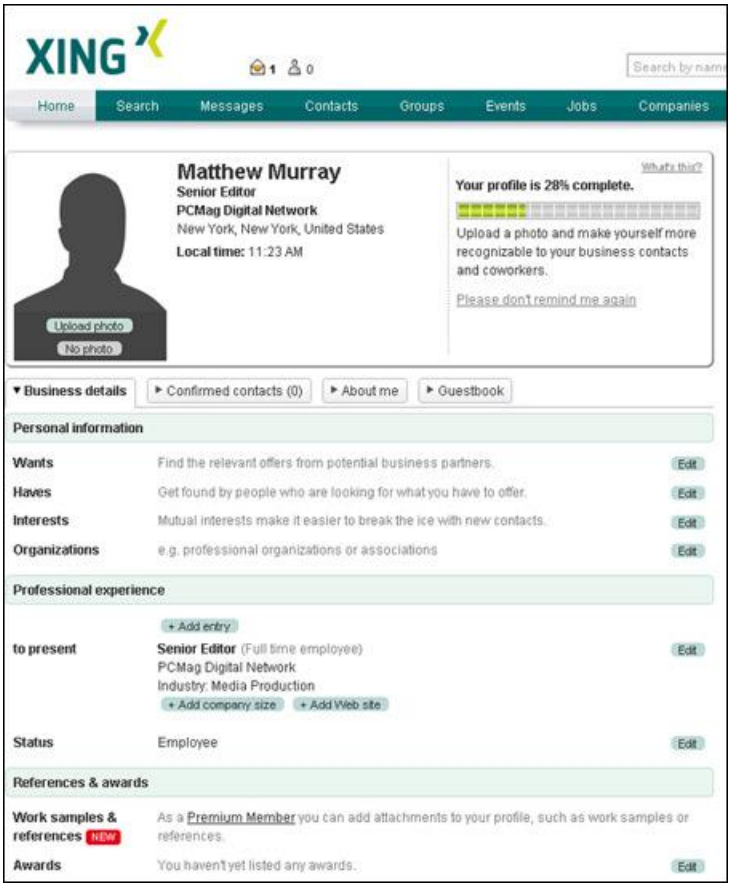

Рис. 1.5. XING

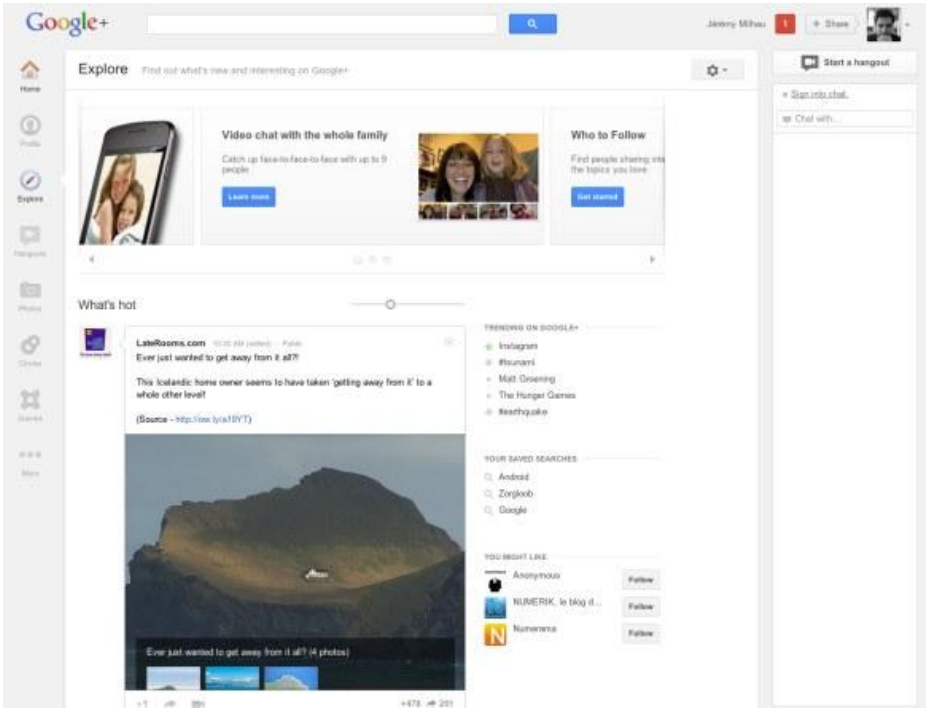

Рис. 1.6. Google+

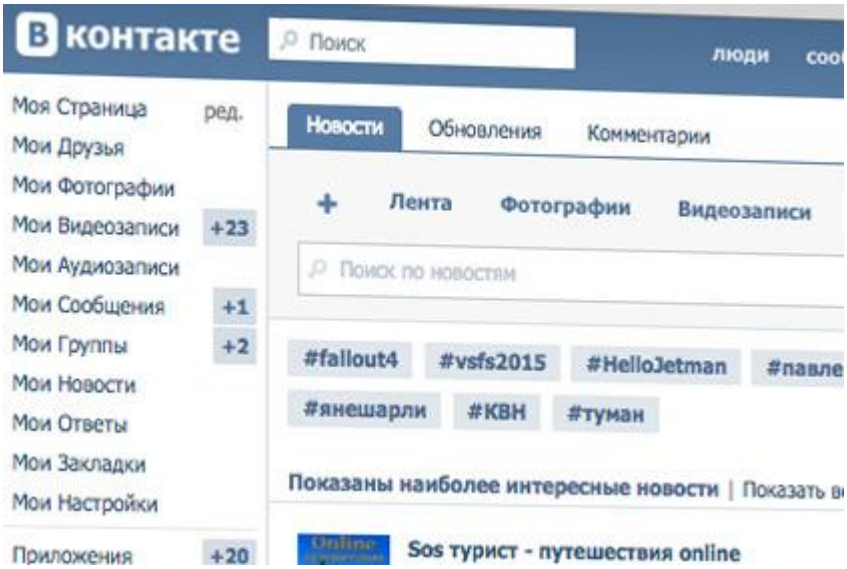

Рис. 1.7. В Контакте

Є наступні типи ресурсов в формат Web 2.0:

- Соціальні закладки дозволяють користувачам веб-сайтів розпоряджатися списком популярних веб-сайтів або інших закладок. Такі сайти также можуть використовуватися для пошуку користувачів із загальними інтересами. Приклад: Delicious;

- Соціальні каталоги часто використовуються в сфері академії, тобто, користувачі можуть працювати з базами даних з наукових статей але не дивлячись на це вони дуже схоже на вище сказані соціальні закладки, схожі на соціальні закладки. Приклади: Acadymic Search Premier, Lexis Nexis Academic University, CiteULike, Connotea;

- Соціальні бібліотеки - є додатки, які дозволяють користувачам залишати посилання на їх колекції, книги, аудіозаписи і т. д., доступні іншим. Передбачена підтримка системи рекомендацій, рейтингів і т. д. Приклади: discogs.com, IMDb.com.;

- Соціальні мережі вебмайстрів використовуються для оголошення корисних, які дозволяють авторам залишати посилання на їхні пости, спілкуватися, голосувати за інтересні анонси і т. д. часто мають рейтинги або рекомендації.

- Сервісні соціальні мережі дають можливість об'єднуватися користувачам з загальними інтересами, захопленнями, а також за іншими приводами;

- В онлайн режимі дозволяють користувачам об'єднуватися в онлайн режимі навколо спільних для них інтересів, захоплень або по різним причинам. Наприклад, деякі сайти надають сервіси, за допомогою яких користувачі можуть розміщувати для загального доступу персональну інформацію, необхідну для пошуку партнерів. Наприклад: LinkedIn, ВКонтакте;

- Сервіси для спільного зберігання медіафайлів – соціальні медіасховища. Їх можна класифікувати за типом файлів що розміщуються на цих серверах;

- Спеціалізовані соціальні мережі - об'єднують людей за певним критерієм, наприклад, вік, стать, віросповідання, певні захоплення і т. д .;

- Професійні соціальні мережі створюються для спілкування на професійні теми, обміну досвідом та інформацією, пошуку і пропозиції вакансій, розвитку ділових зв'язків. Приклади: LinkedIn;

- Корпоративні соціальні мережі вирішують завдання організації та супроводу діяльності компанії;

- Для спільної роботи з документами сервіси;

- Геосоціальні мережі дозволяють на підставі географічного положення користувача налагоджувати соціальні зв'язки. При цьому використовуються різні інструменти геолокації (наприклад, GPS або гібридні системи типу технології AlterGeo), які дають можливість визначати поточне місцезнаходження того чи іншого користувача і співвідносити його позицію в просторі з розташуванням різних місць і людей навколо.

- Комерційні соціальні мережі надають підтримку клієнтам і співробітникам компаній. Вони присвячені збору і публікації відгуків і забезпечують роботу механізму зворотного зв'язку з клієнтами [2].

1.2. Опис предметної області

Незважаючи на безліч переваг і величезне значення існуючих соціальних мереж для інтернет-спільноти, у них є ряд істотних недоліків, які обмежують соціальну взаємодію зареєстрованих в них користувачів:

• наявність жорстко заданих призначених для користувача атрибутів (імєя, прізвище, інтереси, улюблена музика, і т.д.) без можливості додавати свої атрибути до профілю;

• недостатня інтеграція сервісів в системі;

• немає стандартизації, кожна соціальна мережа використовує свої протоколи для організації обміну особистими повідомленнями, доступу до чату, і т. д .;

• недостатня функціональність спільнот, недоступність виділення спільноті окремого доменного імені;

• нерозвиненість системи міток (тегів), її переваги застосовані лише частково;

• недоступність системи групування списку контактів з самого початку;

• (для багатьох ресурсів) недостатня зручність для користувача інтерфейсу, небажання використовувати нові web-технології, такі як AJAX, які в рази підвищують швидкодію і зручність користування системою. Ці недоліки призвели нас до ідеї розробки концепції соціальної мережі нового покоління.

Розглянемо пропоновані нововведення в концепцію соціальних мереж, реалізація яких планується в даному проекті. Організація обміну особистими повідомленнями на основі протоколу IMAP. Як було сказано вище, соціальним мережам не вистачає стандартизації. Цю проблему можна легко вирішити, використовуючи для обміну інформацією відомі протоколи.

Система електронної пошти - найстаріша з нині діючих систем обміну інформацією користувачів інтернету один з одним. Для мережевої взаємодії користувачів за допомогою e-mail, розроблені кілька стандартних протоколів. Найбільш популярні до останнього часу був протокол прийому пошти POP. Він дозволяє отримувати повідомлення електронної пошти з сервера і працювати з ними в offline-режимі. У зв'язку з розвитком інтерактивних систем, що вимагають клієнт-серверного підходу, функціональність протоколу POP виявилася недостатньою, і на перший план вийшов протокол прийому пошти IMAP [4], який увібрав в себе найкраще з багаторічної історії розвитку електронної пошти. Цей протокол дозволяє отримувати тільки заголовки повідомлень, виконувати пошук листів за деякими критеріями, позначати повідомлення прапорами, такими як «Позначено для видалення», «на повідомлення послати відповідь» і виконувати інші операції над віддаленими поштовими скриньками, так якщо б вони були локальними. Для роботи з протоколом IMAP написані тисячі програм-клієнтів для десятків платформ. А це значить, що у користувачів соціальної мережі не буде проблем з отриманням своїх особистих повідомлень, де б вони не знаходилися, - адже їх можна буде отримати за допомогою будь-якого IMAP-клієнта, який сьогодні є в кожному телефоні.

Для підтримки функцій електронної пошти в соціальну мережу пропонується вбудувати поштовий клієнт. Причому відправити email-повідомлення в системі має бути також просто і зручно, як відсилати приватні повідомлення в звичайній соціальній мережі. Еталоном web-пошти на даний момент є поштовий клієнт компанії Google - Gmail. Пропонується реалізувати схожий по зручності і функціональності інструмент.

При реєстрації в соціальній мережі користувачеві видається поштова скринька на IMAP-сервері. Однак учасник може вибрати використання свого вже існуючого електронного ящика будь-якого провайдера, якщо він не хоче заводити додаткову пошту.

Для цього йому потрібно лише ввести в систему пароль від свого ящика, припустимо, на Yahoo, - і він буде працювати зі своєю поштою з нашої системи.

Розширене редагування профілів. Поряд з загальнодоступними заповнюваними полями: «ім'я», «прізвище», «стать», «дата народження»,«інтереси», «улюблена музика» і т. д., користувач може ввести свої унікальні поля. Причому стандартні атрибути будуть просто включені за замовчуванням, а для додавання своїх атрибутів користувачеві доведеться лише один раз клацнути мишкою на відповідній кнопці.

Обмін миттєвими повідомленнями на основі протоколу XMPP. Поряд з системою особистих повідомлень пропонується стандартизувати також використання web-чатів в соціальній мережі. Тому обмін миттєвими повідомленнями буде ґрунтуватися на відкритому протоколі XMPP [5], раніше відомому як Jabber. В результаті користувач, якщо захоче, завжди зможе прочитати свої миттєві повідомлення за допомогою будь-якого Jabber-клієнта, які існують практично для будь-якої платформи.

Примусове групування контактів зі списку «друзів». Однією з найбільших проблем соціальних мереж є те, що в них не було доступно системи розподілу «друзів» по групах: «однокласники», «однокурсники»,

«співробітники» і т.д. З плином часу, проведеного в соціальній мережі, список контактів також зростає.

Рис. 1.8 Використання Jabber-клієнта

Через кілька років користування сервісом цей список у більшості людей досягає 100-200 записів, серед яких досить складно знайти потрібні. Групування користувачів по групах була пізніше введено, але далеко не всі почали їй використовувати: розгрупувати 200 вже існуючих контактів з новим групами – заняття не з приємних. Тому пропонується з самого початку запуску мережі при додаванні в друзі завжди питати користувача, в яку групу він хоче додати контакт.

Розширена функціональність спільноти. Спільноти (групи) в сучасних соціальних мережах є всього лише сторінки з ідентичними головній сторінці дизайном, досить бідним функціоналом (коментарі, форум, блог, фото-галерея), хоча багатство наповнення цієї сторінки часто може позмагатися з багатьма повноцінними web-сайтами. Це стосується спільнот, присвячених різноманітним фірмам, рок-групам, колективам різного роду. Пропонується надати користувачам ресурсу великі можливості в індивідуалізації сторінки групи - починаючи від дизайну, закінчуючи функціональними можливостями, такими як ті ж блоги, музичні архіви, галереї, і т. д. Адміністратор групи зможе

самостійно вибрати дизайн, функції групи, розташувати їх по сторінці потрібним чином, в загальному, сконструювати свій web-сайт, тільки на основі ресурсу соціальної мережі. Пропонується також виділяти для цих ресурсів доменне ім'я або як піддомен доменного імені самої соціальної мережі, або, якщо користувач готовий зареєструвати своє доменне ім'я, - пов'язувати відповідне доменне ім'я з адресою групи.

Система розширеної каталогізації та пошуку всього змісту ресурсу на основі міток. Мітки в сучасних соціальних мережах або не використовуються зовсім, або використовуються дуже мало. У соціальній мережі нового покоління пропонується можливість помічати абсолютно будь-яку інформацію міткою, починаючи від листа особистих повідомлень, закінчуючи музичними записами і відеофайлами. При цьому мітки будуть пропонуватися користувачеві самою системою, за принципом вибору найбільш вживаних слів в тексті. При пошуку по мітці учаснику будуть надані можливості перемикання між знайденими ресурсами. Наприклад, при пошуку по мітці «Бетховен» користувачу будуть доступні музика Бетховена, листи його друга з Німеччини з прізвищем Бетховен або відомий фільм «Бетховен-2». Вибрати потрібний йому ресурс людина зможе шляхом натискання мишкою по відповідній вкладці.

1.3. Специфікація вимог до системи

Специфікація вимог до програмної системи – це специфікація окремого програмного продукту, програми або набору програм, які виконують деякі дії в деякому середовищі. Тобто – це повний опис поведінки системи що розробляється [3].

В загальному випадку специфікація включає наступне:

- глосарій проекту;
- опис варіантів використання.

Наведемо список основних термінів та понять в області розробки програмної системи для організації спілкування між користувачами соціальних мереж – глосарій. Глосарій – список понять в специфічній області знання з їх визначеннями [3]. Ці поняття та визначення подано у таблиці 1.2.

Таблиця 1.1

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Соціальна мережа	веб-сайт або інша служба у Веб, яка дозволяє користувачам створювати публічну або напівпублічну анкету, складати список користувачів, з якими вони мають зв'язок та переглядати власний список зв'язків і списки інших користувачів.
Чат	мережевий засіб для швидкого обміну текстовими повідомленнями між користувачами інтернету в режимі реального часу.
Поштова скринька	популярний сервіс в інтернеті, що робить можливим обмін даними будь-якого змісту (текстові документи, аудіо-, відеофайли, архіви, програми).
Програмне забезпечення	сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм [3].
Програмний продукт	програмний засіб, програмне забезпечення, які призначені для постачання користувачів (покупцеві, замовникові) [3].
Бізнес-процес	будь-яка діяльність, що має вхідний продукт, додає вартість до нього, та забезпечує вихідний продукт для внутрішнього або зовнішнього споживача [3].
База даних	логічно впорядкований та взаємопов’язаний набір даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
2. Користувачі системи	
Адміністратор спільноти	Особа, яка займається безпосереднім управлінням системою на технічному рівні.
Користувач	Особа, яка займається безпосереднім управлінням БД.

Власник профілю	Особа, яка здійснює управлінські функції відділу кадрів та забезпечує взаємодію відділу з іншими структурними підрозділами.
3. Вхідні та вихідні документи	
База даних	логічно впорядкований та взаємопов’язаний набір даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
Список контактів	документ, в якому зберігаються контактні дані користувача

У таблицях 1.2 – 1.12 наведено опис варіантів використання, що реалізують основну функціональність програмної системи. Діаграма варіантів використання представлена на рисунку 1.9.

Таблиця 1.2

Варіант використання «Додати плагін до спільноти»

Контекст використання	Управління спільнотою
Дійові особи	Адміністатор спільноти
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Додати плагін».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення інформації про плагіни.

Таблиця 1.3

Варіант використання «Редагувати зовнішній вигляд спільноти»

Контекст використання	Управління спільнотами.
Дійові особи	Адміністатор спільноти
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Редагувати».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення інформації про спільноту.

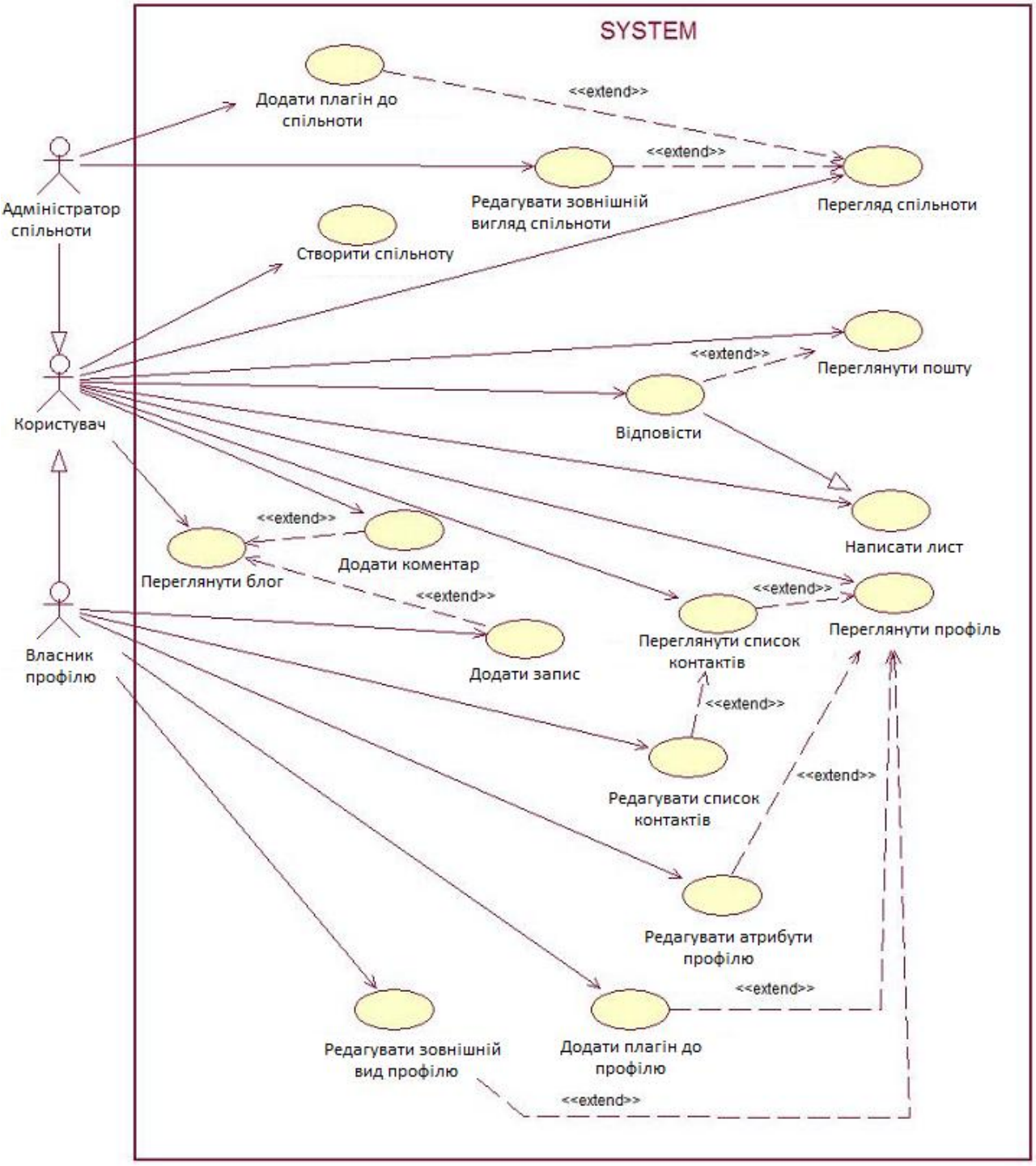

Рис. 1.9. Діаграми варіантів використання

Таблиця 1.4

Варіант використання «Створити спільноту»

Контекст використання	Управління спільнотами.
Дійові особи	Користувач
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Створити».
Сценарій	1. Вибір параметрів. 2. Визначення спільноти. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення списку спільнот

Таблиця 1.5

Варіант використання «Перегляд спільноти»

Контекст використання	Управління інформацією.
Дійові особи	Користувач
Передумова	
Тригер	Відкриття розділу «Спільноти».
Сценарій	-
Постумова	Відображення спільнот в розрізі ознак.

Таблиця 1.6

Варіант використання «Перегляд пошти»

Контекст використання	Управління інформацією.
Дійові особи	Користувач
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Переглянути пошту».
Сценарій	-
Постумова	Відображення листів.

Таблиця 1.7

Варіант використання «Перегляд блогу»

Контекст використання	Управління інформацією.
Дійові особи	Користувач
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Переглянути блог».
Сценарій	-
Постумова	Відображення блогу.

Таблиця 1.8

Варіант використання «Редагувати список контактів»

Контекст використання	Управління контактами
Дійові особи	Власник профілю
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Редагувати список контактів».
Сценарій	1. Вибір контакту. 2. Заповнення необхідних полів. 3. Перевірка введених даних. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення даних.

Таблиця 1.9

Варіант використання «Додати плагін до профілю»

Контекст використання	Управління профілем
Дійові особи	Власник профілю
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Профіль».
Сценарій	1. Вибір плагіну. 2. Формування параметрів. 3. Перевірка. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення списку плагінів.

Таблиця 1.10

Варіант використання «Редагувати атрибути профілю»

Контекст використання	Управління профілем
Дійові особи	Власник профілю
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Профіль».
Сценарій	1. Вибір необхідного атрибуту. 2. Натиснення кнопки «Редагувати». 3. Підтвердження операції.
Постумова	Відображення профілю.

Таблиця 1.11

Варіант використання «Редагування зовнішнього виду профілю»

Контекст використання	Управління профілем
Дійові особи	Власник профілю
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Профіль».
Сценарій	1. Вибір теми профілю. 2. Натиснення кнопки «Редагувати». 3. Заповнення необхідних полів. 4. Перевірка введених даних. 5. Натиснення кнопки «Зберегти».
Постумова	Відображення виду профілю.

В таблиці 1.12 наведено специфікацію функціональних вимог програмної системи.

Таблиця 1.12

Специфікація функціональних вимог

Іденти-фікатор вимоги	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
1	Додати плагін до спільноти	рекомендоване	середня	Адміністратор
2	Редагувати зовнішній вигляд спільноти	обов’язкове	висока	Адміністратор
3	Створити спільноту	обов’язкове	висока	Адміністратор
4	Перегляд спільноти	обов’язкове	висока	Адміністратор
5	Перегляд пошти	обов’язкове	висока	Адміністратор
6	Перегляд блогу	рекомендоване	середня	Адміністратор
7	Редагувати список контактів	обов’язкове	висока	Адміністратор
8	Додати плагін до профілю	обов’язкове	висока	Адміністратор
9	Редагувати атрибути профілю	обов’язкове	висока	Адміністратор
10	Редагування зовнішнього виду профілю	опційне	висока	Адміністратор

Специфікацію нефункціональних вимог наведено в таблиці 1.13.

Наведемо специфікацію суттєвих для проекту нефункціональних вимог:

1. Застосовність:
   - мінімальний час для навчання звичайних і досвідчених користувачів;
   - відповідність стандартам графічного інтерфейсу.
2. Надійність:
   - постійна безвідмовна робота;
   - пропускна здатність каналу зв’язку 100 Mb/s;
   - забезпечення можливості віддаленого доступу до комп’ютера, на якому буде встановлена система;
   - доступність – 5%.
3. Робочі характеристики:
   - швидкість завантаження інтернет-ресурсу: 0,1 – 1 с;
   - число транзакцій: 100 / 1 с;
   - використання ресурсів: від 1 Gb, в залежності від кількості студентів.
4. Проектні обмеження:
   - Операційна система Linux;
   - MySQL;
   - Symfony компанії Sensio Labs;
5. Вимоги до документації
   - наявність інтерактивної довідки.
6. Інтерфейси:
   -інтерфейс користувача – веб-інтерфейс.

Таблиця 1.13

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
Застосовність				
1.1	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Низька	Адміністратор
1.2	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Опційна	Низька	Адміністратор
1.3	Вимоги по відповідальності стандартам графічного інтерфейсу користувача	Рекомендована	Низька	Адміністратор
Надійність				
2.1	Доступність	Обов’язкова	Середня	Адміністратор
2.2	Середній час безвідмовної роботи	Рекомендована	Середня	Адміністратор
2.3	Точність	Обов’язкова	Середня	Адміністратор
Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Адміністратор
4.1	Вимоги до технології програмування	Рекомендована	Середня	Адміністратор

Висновки до розділу 1

Здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих соціальних мереж. Здійснено аналіз вимог до програмної системи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ ОРГАНІЗАЦІЇ СПІЛКУВАННЯ МІЖ КОРИСТУВАЧАМИ СОЦІАЛЬНИХ МЕРЕЖ

2.1. Розроблення архітектури програмної системи

Для вирішення даної бізнес проблеми, була вибрана клієнт-серверна архітектура. Вона найбільш придатна для задач саме такого типу, коли потрібно забезпечити доступ до системи багатьом користувачам. Найголовнішим є те, що клієнт-серверна архітектура надає можливість віддаленого доступу [4].

Проект, який побудований на клієнт-серверній архітектурі, повинен складатися з трьох частин: зв'язок з базою даних, відображення даних клієнту та бізнес логіка проекту яка обробляє запити користувача і відображає саме ту інформацію, яку захотів користувач. Обробка та збереження даних відбувається на боці сервера, відображення даних і надсилання запитів на їхню модифікацію виконується на боці клієнта.

Статичний аспект такої архітектури зображений, за допомогою наступної діаграми, на рисунку 2.1.

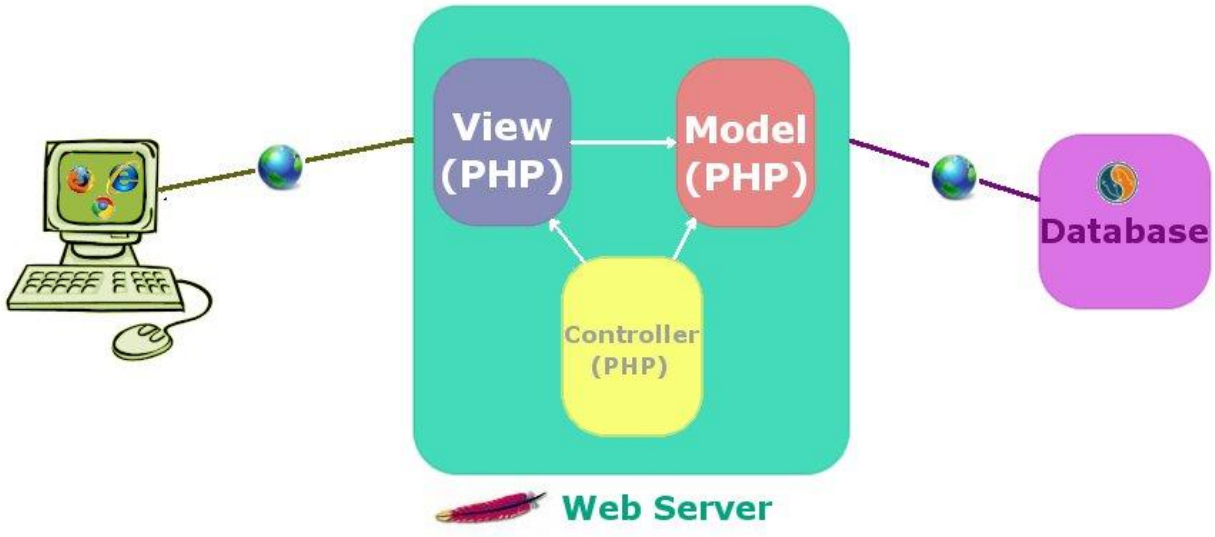

Рис. 2.1. Реалізації клієнт-серверної архітектури

На цій діаграмі клієнтською аплікацією виступає клієнт, розроблений на основі, тобто PHP. Він, у свою чергу, віддає на сервер запити, відповідно до того з якими формами працює працівник відділу кадрів. Реалізація взаємодії із базою даних здійснюється за допомогою SQL запитів.

Уся бізнес-логіка проекту виконується на другому ріні. Вся обробка запитів від клієнта і надсилання йому відповіді здійснюється на цьому рівні. Також тут визначається, які запити повинні іти до бази даних. Третій рівень – це власне сама база даних, яка приймає Sql-запити, і у відповідності їм повертає дані.

На рисунку 2.2. представлено функціональну структуру модулів.

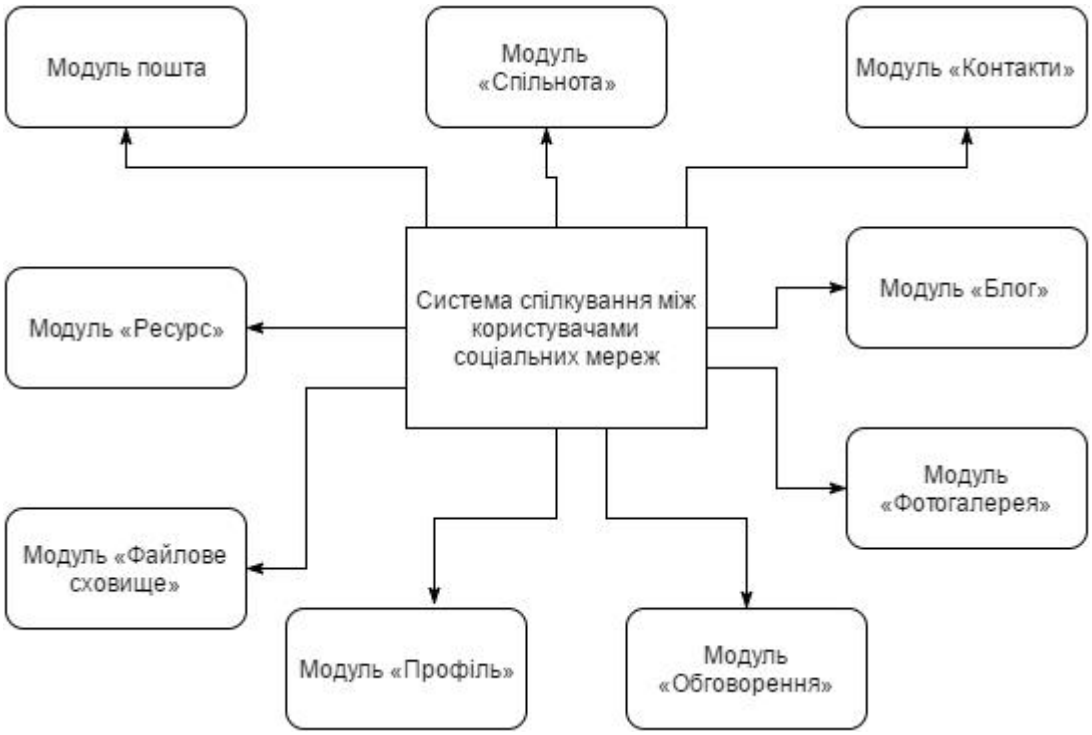

Рис. 2.2. Функціональна структура модулів

На рисунку 2.3 представлено діаграму розміщення системи із врахування особливостей використання.

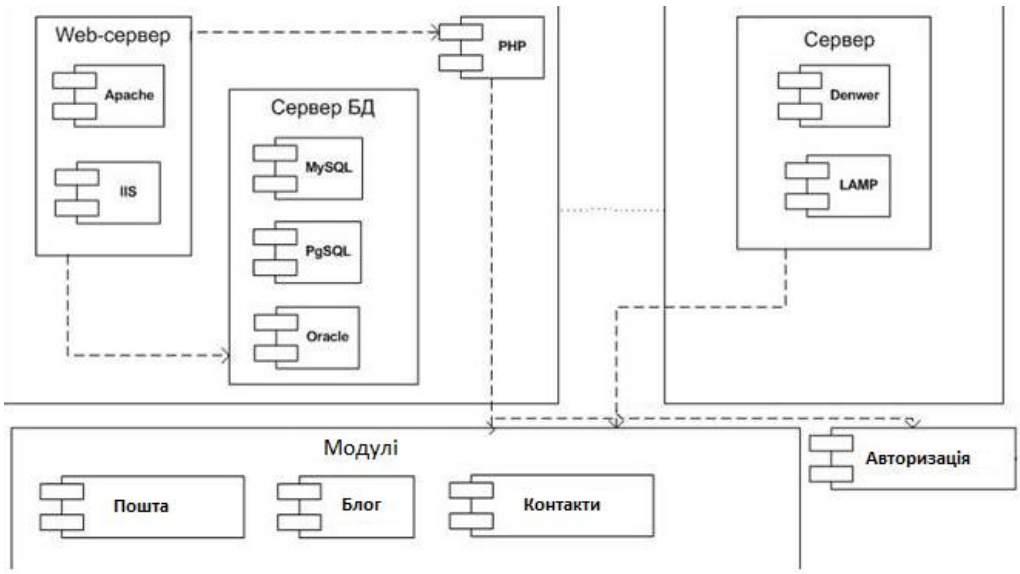

Рис. 2.3. Діаграма розгортання

На рисунку 2.4 представлено діаграму послідовності дій при вході в соціальну мережу

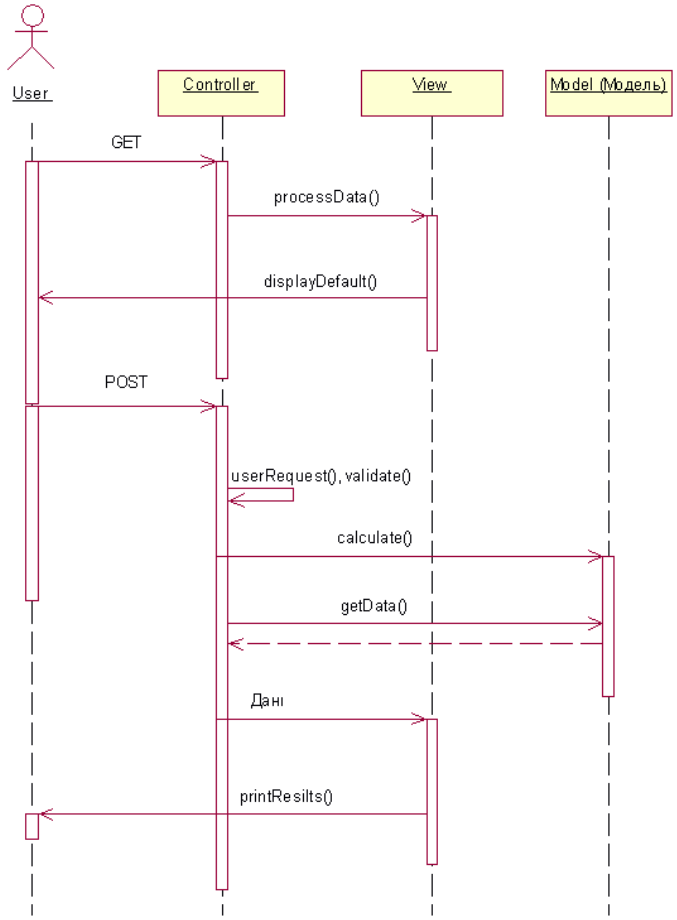

Рис. 2.4. Діаграма послідовності дій при вході в систему

Під час роботи система може знаходитися в наступних станах: авторизація, введення, обновлення, виведення, створення, збереження, які представлено на рисунку 2.5.

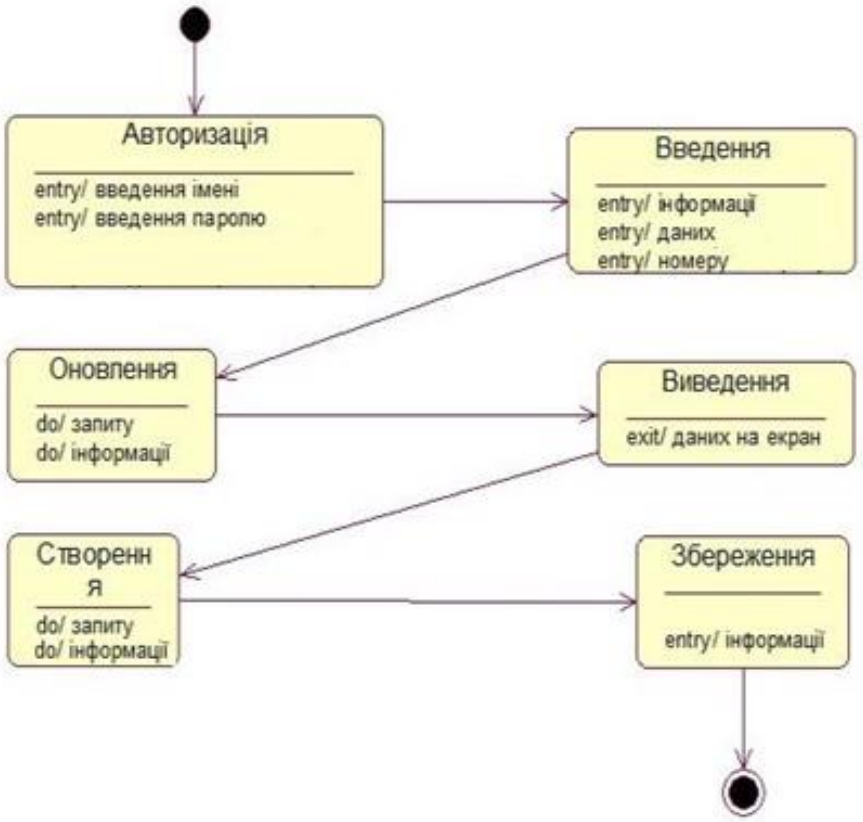

Рис. 2.5. Діаграма станів процедури авторизації

Процес авторизації користувача відбувається за наступними кроками (рисунок 2.6): введення логіну, введення паролю, перевірка логіну та пароля. Якщо логін чи пароль введено невірно, система повертається до першого кроку – введення логіну та пароля.

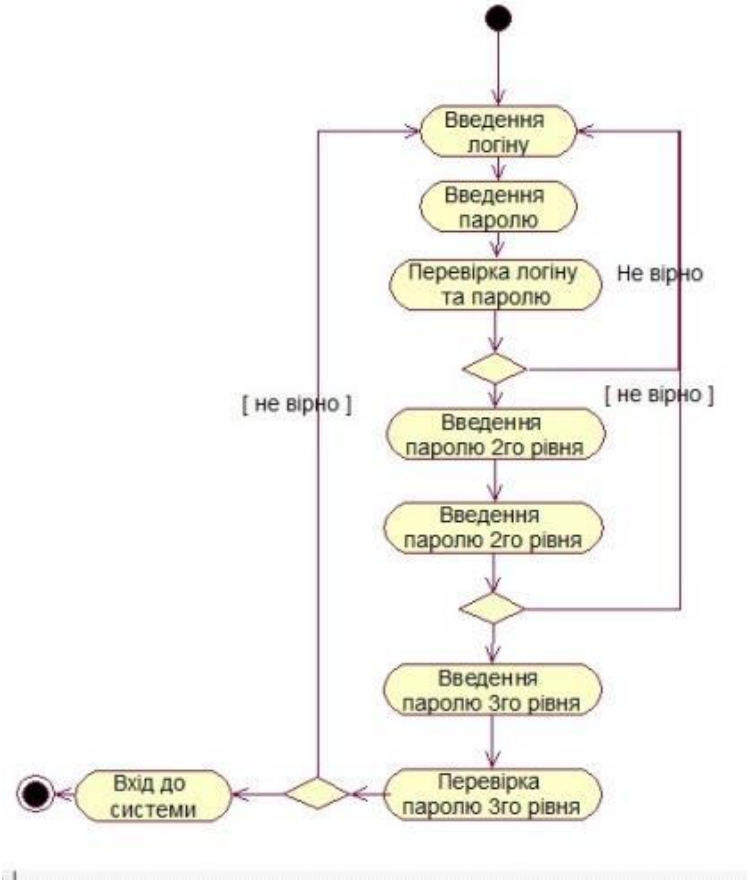

Рис. 2.6. Діаграма діяльності «Авторизація»

2.2. Проектування структури бази даних

У соціальній мережі присутні два типи користувачів: адміністратори (також виконують роль модераторів) і звичайні користувачі. Між звичайними користувачами немає ніяких відмінностей і ніхто з них не володіє великими перевагами, ніж інші, можливостями і повноваженнями. Адміністратор також може мати свій профіль, який для інших користувачів не буде нічим відрізнятися. Але, перебуваючи всередині системи, адміністратору будуть доступні деякі додаткові можливості.

Побудова концептуальної моделі соціальної мережі. Соціальна мережа складається з наступних основних таблиць:

- Альбоми;
- Аудіо;

- Банери;
- Місто;
- Спільноти;
- Країна;
- Файли;
- Друзі;
- Повідомлення;
- Новини;
- Користувачі;
- Відео;
- Стіна;
- Фото;
- Чат і т.д.

На головній сторінці сайту будуть новини. Також будуть посилання на новини в інтернеті. Ці новини можуть читати не зареєстровані користувачі. Також на головній сторінці буде погода, курси валют і реклама. На цій же сторінці буде і реєстрація. Користувач повинен клікнути кнопку «Зареєструватися», з'явиться вікно користувацької угоди. Користувач повинен прочитавши цю угоду погодитися або ж не погодитися. Якщо користувач згоден, він повинен натиснути «Я згоден з даним запрошенням »після цього на e-mail користувача буде відправлено лист підтвердження. Після набраного коду активації, користувач стає повноправним членом мережі.

Після становлення учаником соціальної мережі користувачеві дозволяється користуватися основними можливостями сайту. Під час реєстрації кожному члену сайту дається id. Тому по id - відкривається папка цьому користувачеві. Фотографії та інші дані користувача зберігаються в цій папці. Друга сторінка - профіль користувача. Тут всі дані, які стосуються користувача. Наприклад, адреса, освіта і місце роботи, і інші дані, також на цій сторінці будуть коментарі користувача. Якщо користувач з допомогою сторінки

Редагувати не поставить профільну фотографію, то за замовчуванням сайт сам встановить. Під профільної фотографією будуть функції, які стосуються цієї фотографії, наприклад видалити, змінити, додати фотографію, вибрати мініатюру.

Користувач на сторінці редагування має можливість змінити ключове слово і видалити свій профіль коли захоче. На сторінці Друзі користувач може переглядати коротку інформацію про інших користувачів і їх профільну фотографію. На веб-сервері відео-сайту відео не ставиться. Користувач може онлайн дивитися відео з сайту youtube.com, для цього треба заповнити на сторінці Редагування в поле youtube channel свій логін на сайті youtube.

На сторінці пошта користувач може відправляти листи на будь-який електронний адрес. На сторінці коментар показані коментарі всіх користувачів. Це в свою чергу дає можливість переглядати всі коментарі, що не відвідують профіль кожного користувача.

На сторінці чат працює система швидкого обміну повідомлень. Цю можливість може використовувати будь-який зареєстрований користувач. За допомогою команди Вийти користувач завершує свою роботу. Користувач може за допомогою ключового слова та електронної адреси, знову скористатися вищесказаними можливостями.

Ключовим елементом розроблюваної соціальної мережі є ресурс. Ресурс – це сторінка з деякою інформацією, яка є або профілем певного користувача, або сторінкою спільноти. Елементи на сторінці ресурсу інтерактивні, тобто їх можна видозмінювати довільним чином, якщо на це є відповідні права.

Введемо поняття віджета - елемента ресурсу, що містить будь-яку інформацію. Віджет являє собою деяку область на сторінці ресурсу, яку можна переміщати по сторінці, розтягувати, змінювати колір і виконувати інші дії. Віджети можуть бути як вбудовані - список друзів, атрибутів, фотографія користувача, так і підключатися - блог, фотогалерея, і т.д. З кожним віджетом пов’язано якесь місце: список друзів або повідомлень в блозі, фотоальбомів і т.

д. У кожного віджета може існувати «повна» сторінка, яка відображається вкладкою на сторінці ресурсу, там міститься повна інформація по видимій частині віджета.

Як уже згадувалося вище, до системи можливе підключення додаткової функціональності. Це реалізовано за допомогою системи плагінів - підключених до ресурсу модулів. Кожен плагін може містити віджет, що показується на основній сторінці ресурсу і головну сторінку, показує при натисканні на відповідну вкладку плагіну.

Таким чином, сторінка ресурсу складається з набору віджетів, розташованих на екрані в довільному порядку і деякого числа вкладок для переходу на функціональні елементи ресурсу. Набір відображаються на сторінці плагінів і віджетів, авторизований користувач може змінити на сторінці налаштування.

Важливою складовою частиною системи є поштовий клієнт. Він являє собою деякий інтерфейс для відправки і перегляду email-повідомлень.

Відповідно до загальних архітектурних принципів, основною табліцею в реляційній базі даних програми є таблиця ресурсів resource (рисунок 2.1). З нею пов'язані таблиці плагінів, наприклад, blog і post, що зберігають дані для блогу. Таблиці профілів (profile) і спільнот (community) також пов'язані з таблицею resource. Таблиця-словник атрибутів (attribute_voc) зв'язується з таблицею профілів (profile) і зі словником значень атрибутів (attribute_value_voc). Таблиця контактів (contact) позначає знаходження користувача в списку контактів іншого користувача, таким чином пов'язуючи таблицю profile саму з собою. На рисунку 2.7 представлено діаграму «Сутність-зв'язок» бази даних для досліджуваної системи.

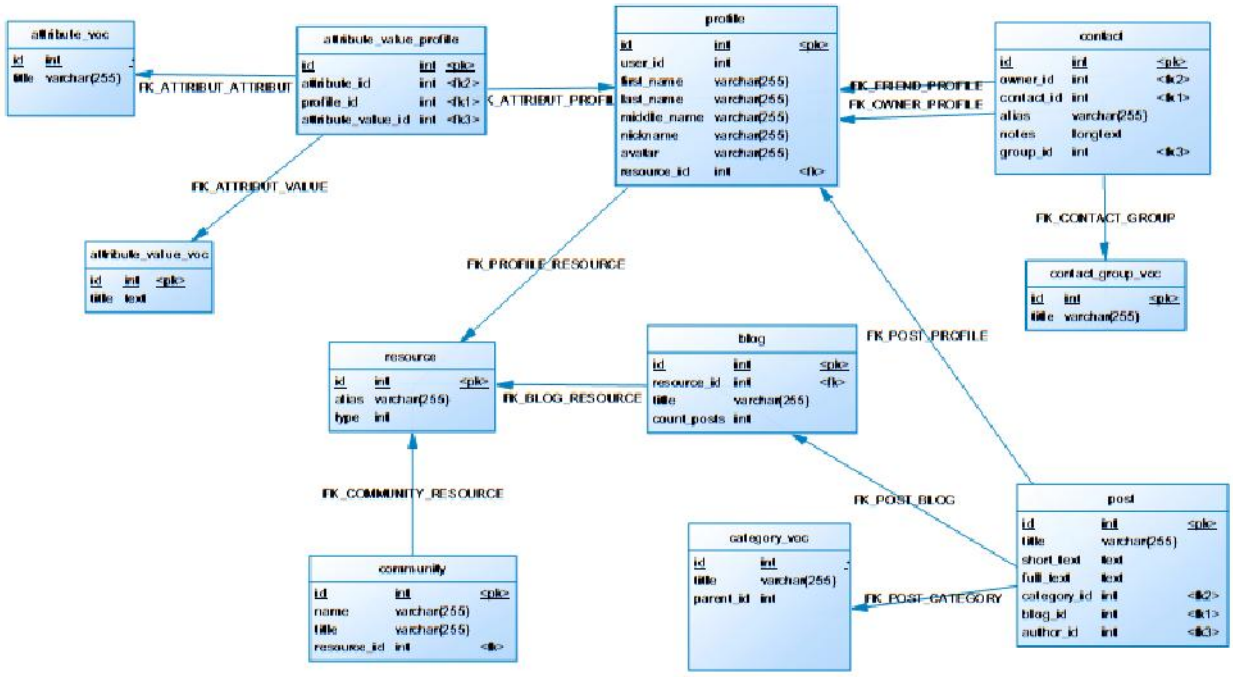

Рис. 2.7. Діаграма сутність зв’язок

Висновки до розділу 2

У цьому розділі розроблено архітектуру програмного модуля, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв’язки між ними. Спроектовано структуру бази даних.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Програмна реалізація модуля

Як було сказано вище, проект, що розробляється на базі платформи Symfony (рисунок 3.1), ділиться на кілька модулів, контролери яких ініціюють виконання визначених дій. У процесі розробки були реалізовані наступні модулі, які є складовими ядра системи: «пошта», «ресурс», «профіль», «спільнота», «контакти», а також кілька прикладних модулів: «блог», «фотогалерея», «файлове сховище», «особистий блог» і «обговорення».

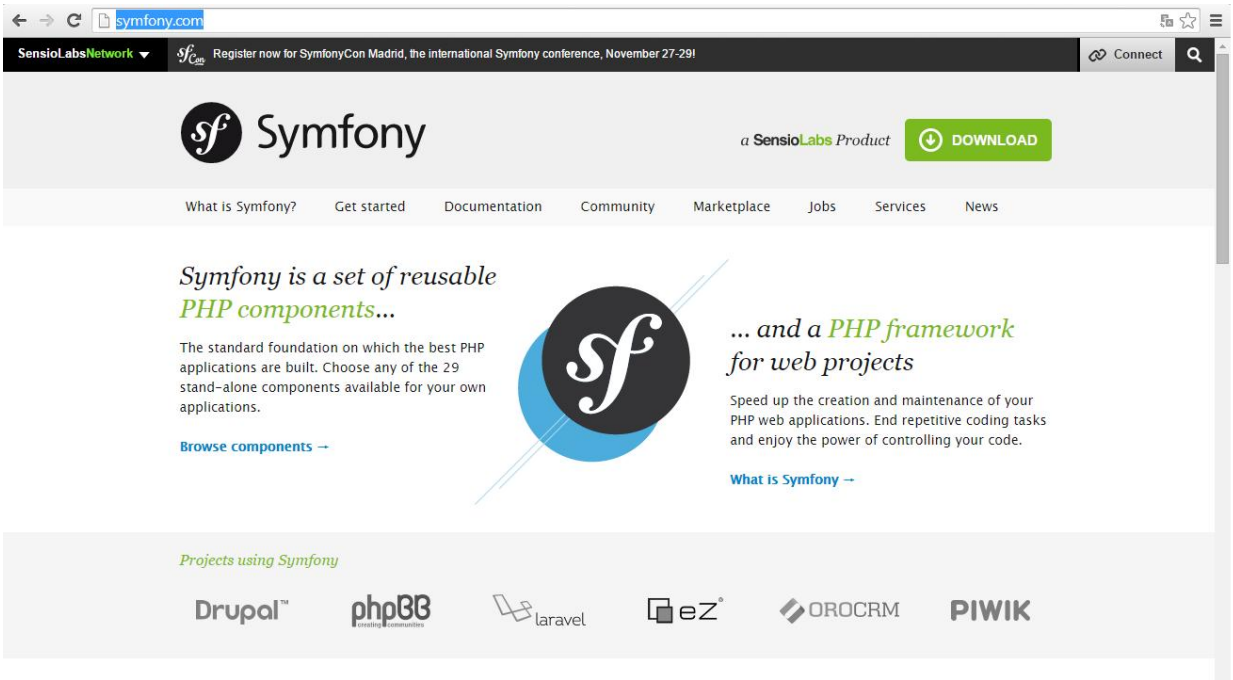

Рис. 3.1. Symfony Framework

У платформі Symfony є багато службових класів для обробки запиту, фільтрації, які викликаються перед тим, як запускається потрібна дія контролера. Описувати їх всі на діаграмі класів немає сенсу. Відзначимо, що необхідний для запуску контролер визначається методом класу sfRouting, після чого запускається метод execute цього контролера, який, в свою чергу, запускає

потрібну дію. На діаграмі (рисунок 3.2) наведені основні класи-контролери (mailActions, resourceActions, і т. д.), їх методи-дії, а також класи моделі і класи, що реалізують додаткову логіку предметної області.

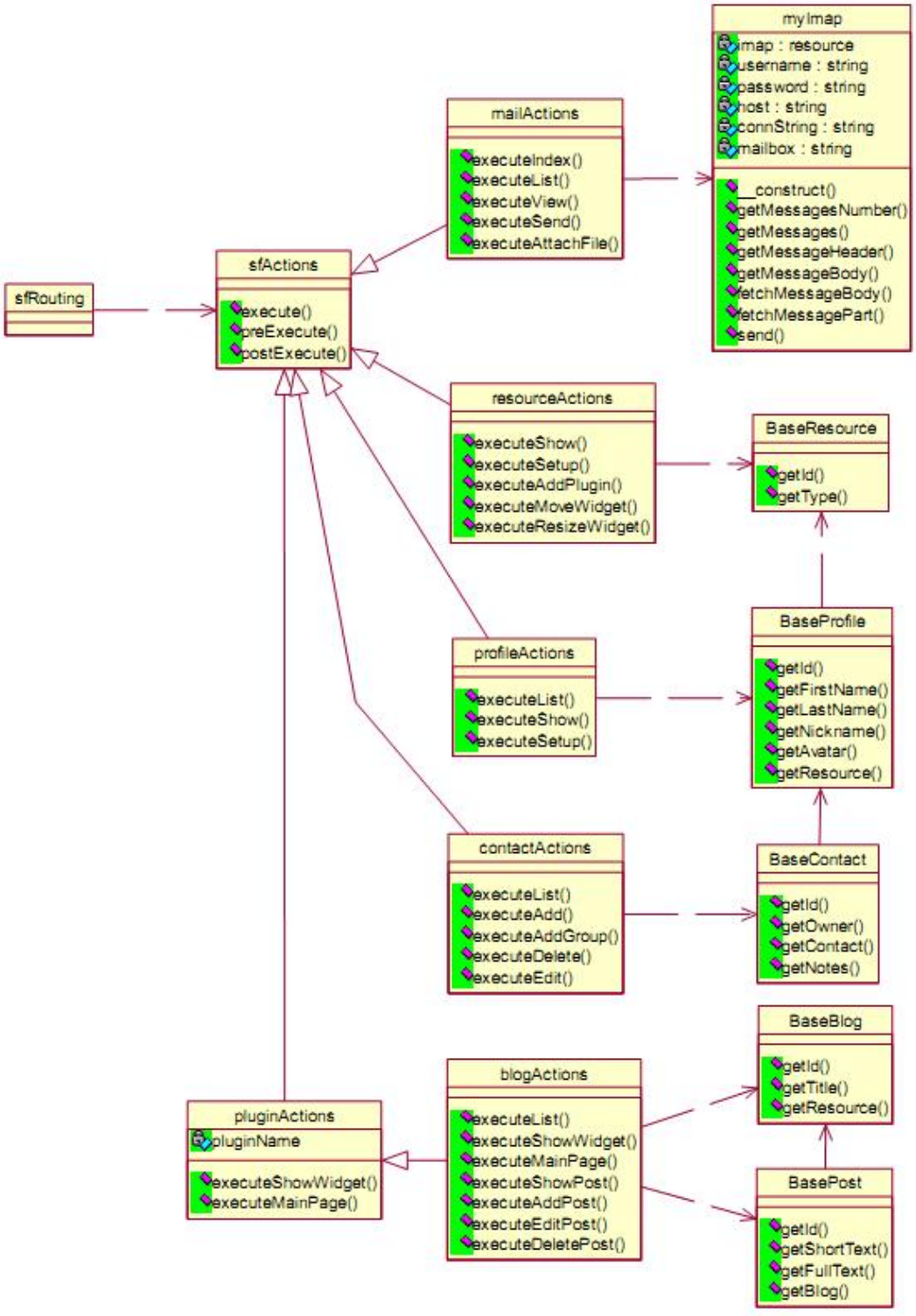

Рис. 3.2. Діаграма класів модуля

При описі модуля буде приводитися короткий огляд його функціональності в цілому, а також методи відповідного класу-контролера, які є діями. При використанні дій модуля сторонніх класів, що реалізують бізнес-логіку, вони також будуть описуватися в даному розділі.

Модуль «Пошта». Даний модуль реалізує функціональність email-клієнта, що працює через протокол IMAP. Так як всі повідомлення, що відправляються в системі, відправляються за допомогою даного протоколу, модуль «Пошта», є ключовим модулем при взаємодії користувачів один з одним.

Клас mailActions модуля містить наступні методи-дії.

• Index - отримує від IMAP-сервера список заголовків поштових повідомлень в поточної папці і виводить їх список на екран. Він також виводить список папок («Вхідні», «Надіслані», «Чернетки», «Дистанційні») і посторінкову навігацію внизу списку повідомлень. Дія приймає один параметр - ім'я папки, з якої потрібно отримати повідомлення.

• List - дія, що викликається за допомогою асинхронного запиту (AJAX). Отримується від IMAP-сервера список заголовків повідомлень в зазначеній папці на вказаній сторінці і повертає їх сценарієм на JavaScript. Сценарій завантажує список в відповідну область на екрані. Використовується для завантаження повідомлень при виборі користувачем певної сторінки або папки на екрані. Приймає два параметри: ім'я папки на сервері і номер сторінки.

AttachItem - зберігає вказаний користувачем файл для подальшої відправки його поштовим вкладенням. Викликається після натискання користувачем на кнопку «Прикріпити файл» у вікні редагування повідомлення (див. нижче) і вибору файлу для завантаження.

• Send - в разі, якщо дія викликається методом GET - виводить форму редагування повідомлення: текстові поля для введення адресата, теми листа, текста листа, кнопки «Прикріпити файл» і «Надіслати лист». При натисканні на «Прикріпити файл» з'являється діалог вибору файлу.

Для поліпшення зручності користування системою завантаження файлів відбувається без перезавантаження сторінки. Такий механізм реалізований за допомогою динамічного створення елемента IFRAME і вказування в атрибуті target форми імені цього елемента. При виборі користувачем файлу для завантаження виконується дія submit в цей елемент, після чого сценарій на JavaScript перевіряє вмісту цього елемента. Після того, як у вмісті з'явиться повідомлення success і ім'я завантаженого файлу, що повернулося в результаті роботи дії AttachItem, файл вважається завантаженим, і JavaScript вставляє текст з ім'ям файлу в спеціальне поле на екрані.

У разі якщо дія викликана методом POST (користувач натиснув на кнопку «Відправити» при редагуванні повідомлення) і всі дані, введені користувачем, коректні - адресати, теми, тексти листа і завантажених файлів формується багатошаровий лист (Multipart) і відправляється за допомогою протокола SMTP.

• DeleteAttach - видаляє файл для вкладення. Приймає на вхід ім'я файлу.

Для забезпечення виконання в діях операцій, пов'язаних з отриманням поштових листів по протоколу IMAP, використовується допоміжний клас myImap. Опишемо його методи:

__construct ($username, $password, $mailbox, $host) - конструктор класу. Встановлює з'єднання з зазначеним поштовим сервером, використовуючи зазначене ім'я користувача та пароль.

GetMessagesNumber() - отримує з сервера і повертає кількість повідомлень в поточній поштовій скриньці.

• GetMessages ($page, $perPage) - повертає заголовки повідомлень на зазначеній сторінці поточної поштової скриньки.

• GetMessageHeader ($messageNum) - повертає заголовок повідомлення з вказаним номером.

• GetMessageBody ($messageNum) - повертає тіло повідомлення із вказаним номером.

• FetchMessagePart ($part, $partNum, $messageNum, $message) – рекурсивний метод, розбирає структуру складного рекурсивного повідомлення.

• FetchMessageBody ($messageNum) - запускає метод FetchMessagePart від частин повідомлення верхнього рівня. Повертає об’єкт myMessage.

• Send ($to, $subject, $text, $files, $uploadDir) - відправляє повідомлення з вказаною темою, текстом, завантаженими файлами вказаному адресатові.

Всі методи класу myImap використовують бібліотеку php_imap для виконання своїх функцій.

Модуль «Ресурс». Даний модуль забезпечує відображення ресурсів - профілів і співтовариств, а також відповідає за висновок і інтерактивні можливості віджетів. Дії контролера resourceActions:

• Show – здійснює вивід сторінки ресурсу. Витягує з бази інформацію про віджети і виводить їх на екран. Також, виводить вкладки для перемикання між плагінами.

• Setup - витягує і виводить настройки ресурсу: підключення плагіни, віджети, іншу інформацію про дизайн і вміст ресурсу. Якщо метод - Post, записує нові настройки в базу даних.

• AddPlugin - додати новий плагін до функціональності ресурсу. Приймає два параметри - номер ресурсу.

• MoveWidget - заносить в базу даних зміни про переміщення віджета по екрану. Приймає параметром нові координати і ідентифікатор віджета.

• ResizeWidget - заносить в базу даних зміни розміру віджета. Приймає три параметри: новий розмір віджету і його ідентифікатор.

Модуль «Профіль». Відповідає за вивід призначеного для користувача профілю і надає можливості для його редагування. Методи класу profileActions:

• Show - перенаправляє запит користувача на дію Show модуля Resource відповідного профілю ресурсу.

• Setup - виводить настройки призначеного для користувача профілю: ім'я, прізвище, по батькові, псевдонім, аватар, у вигляді текстових полів, а також кнопку «Зберегти».

• Save - зберігає призначені для користувача настройки в базу даних.

Модуль «Спільнота». Відповідає за виведення сторінки спільноти. Дії:

• Show - перенаправляє запит користувача на дію Show модуля Resource для відповідного ресурсу.

Модуль «Контакти». Відповідає за ведення списку контактів. Дозволяє додавати, видаляти учасників з списку контактів, а також призначати їм псевдоніми, під якими вони будуть відображатися в системі. Модуль виконує наступні дії.

Index - витягує з бази даних список контактів для поточного користувача в обраній групі і виводить їх на екран. Тут же доступна кнопка «редагувати», натискання на яку замінює ім'я обраного для редагування контакту на поля для введення. Після натискання на «Зберегти», викликається дія Edit даного модуля. Дія List приймає на вхід два параметра. Перший - ідентифікатор користувача, для якого потрібно вивести список друзів, або нічого, якщо потрібно отримати список друзів для поточного користувача, який увійшов в систему. Другий - ідентифікатор групи, для якої потрібно вивести контакти.

• Edit - дія, що викликається за допомогою технології AJAX при натисканні на «Зберегти» при редагуванні друзів. Приймає на вхід ідентифікатор користувача і дані, які потрібно зберегти. Зберігає нові дані про контакти в базі даних і повертає success, якщо все пройшло вдало

• Delete - видаляє контакт зі списку контактів. Приймає на вхід ідентифікатор контакту. Також викликається за допомогою AJAX-запиту.

• Add - додає контакт в список контактів. Приймає на вхід ідентифікатор користувача, що додається та ідентифікатор групи, до якої слід додати контакт.

• AddGroup - додає нову групу до списку груп користувача і зберігає її в базі даних. Виконується за допомогою AJAX-запиту. Приймає параметр ім’я групи.

Модуль «Блог». Прикладний модуль, додає на сторінку ресурсу блог, в якому адміністратор може залишати повідомлення, а інші користувачі - коментувати їх. Модуль виконує наступні дії.

• Index - витягує з бази і виводить на сторінку список повідомлень в блозі визначеного ресурсу. Приймає один параметр: ідентифікатор ресурсу.

• Post - розміщує повідомлення в блозі: зберігає його в базу даних. Приймає наступні параметри: заголовок, текст поста.

• Show - витягує з бази і виводить повний текст певного поста в блозі. Також виводить коментарі до посту і поле для введення коментаря.

• Comment - зберігає коментар в базу даних. Приймає 2 параметра: ідентифікатор поста і текст коментаря.

Модуль «Обговорення». Прикладний модуль, відповідає за включення в співтовариства або профілі користувачів обговорень - групи тим, з можливістю створення і залишення коментарів кожним учасником. Дії модуля наступні.

• Index - дістає з бази і виводить список тем для поточного ресурсу. Приймає на вхід один параметр - ідентифікатор ресурсу.

• Post - залишає повідомлення в темі. Приймає параметрами тему і текст повідомлення, перевіряє правильність введених даних і вставляє повідомлення в записи в базі даних.

• Edit - редагує повідомлення на форумі, якщо поточний користувач є користувачем, який залишив повідомлення. Приймає параметром ідентифікатор повідомлення.

• List - дістає з бази і виводить список повідомлень для поточної теми. Приймає на вхід один параметр - ідентифікатор теми.

Модуль «Фотогалерея». Являє собою прикладний модуль для перегляду і редагування фотогалереї, прив'язаної до певного ресурсу. Модуль виконує наступні дії.

• Index - виводить список альбомів для поточного ресурсу. Приймає один параметр: ідентифікатор ресурсу.

Album - виводить список фотографій для обраного альбому. Параметри: ідентифікатор альбому.

• Show - виводить на сторінку фотографію з альбому. Параметри: ідентифікатор фотографії.

• Add - додає фотографію в альбом. Параметри: ідентифікатор альбому.

Модуль «Файлове сховище». Даний прикладний модуль дозволяє використовувати на сторінці ресурсу сховище файлів. Дії модуля наступні.

• Index - виводить список файлів за певними критеріями. Параметри: ідентифікатор ресурсу, критерії пошуку файлів (тип, розмір, розширення, і т.д.).

• Upload - дозволяє завантажити файл в файлове сховище.

• Audio - викликає дію Index з критерієм пошуку по аудіо-файлах.

• Video - викликає дію Index з критерієм пошуку по відео-файлах.

• Pictures - викликає дію Index з критерієм пошуку по файлах картинок.

Для розширення функціональності необхідно зрозуміти принцип побудови і розміщення основних компонентів системи. Так як ми маємо справу з MVC-додатком, компоненти діляться на 3 шари - Модель, Вид і Контролер.

Модель. Шар моделі представлений класами для роботи з даними. Ці класи розташовуються в каталозі lib/model директорії додатку. В каталозі model розміщується підкаталог om, в якому знаходяться файли BaseModelNamePeer.php і BaseModelName.php, де ModelName - ім'я моделі (найчастіше збігається з ім'ям таблиці в базі даних). Ці класи відповідають відповідно за витяг/роботу з даними і самі дані. Наприклад, за допомогою виклику методів класу BaseProfilePeer програміст витягує дані з бази даних, які в свою чергу є екземплярами класу BaseProfile. Вищеназвані класи - абстрактні,

вони були згенеровані за допомогою ORM Propel зі схеми бази даних. У роботі з системою використовуються успадковані від цих класів класи ModelNamePeer і ModelName, розташовані на рівень вище в каталозі model. Ці класи програміст може доповнити своїми методами, наприклад, дописати в ProfilePeer.php необхідні нестандартні методи вилучення користувацьких профілів з бази даних, а в Profile.php - методи роботи з окремими примірниками профілів. Таким чином програміст може розширювати шар моделі.

Вид. Шар виду являє собою набір шаблонів на мові Smarty [8]. Всі вони знаходяться в каталозі додатки - apps/frontend. Шаблони діляться на загальні і шаблони модулів.

Загальні шаблони розташовуються в каталозі templates і використовуються різними модулями, наприклад - layout.tpl. Даний шаблон є «рамкою» сайту, що містить логотип, меню, і інше представлення сайту. У нього вставляється результат роботи модулів. Ще один приклад загального шаблону - pager.tpl, що містить зовнішній вигляд посторінкової навігації на сайті.

Поряд із загальними шаблонами існують шаблони модулів. Вони розташовуються в каталозі modules/module/templates і служать для відображення результату роботи дій контролерів.

Контролер. Шар контролера представлений численними класами з іменами ModuleActions, де Module - ім'я модуля. Ці класи розташовані в файлах actions.class.php в підкаталозі actions каталогу кожного модуля. Методи цих класів є дії контролера, які запускаються після обробки клієнтського запиту классм-маршрутизатором. Для розширення функціональності контролерів необхідно додати/змінити відповідний метод в певний клас. Наприклад, щоб додати дію Index до модуля Profile, слід дописати метод indexSuccess до класу ProfileActions, розташованому в apps/frontend/modules/profile/actions/ actions.class.php. Детальну інформацію щодо методів, доступним при реалізації функціональності контролерів, видів і моделей, можна знайти в документації на

офіційному сайті платформи розробки Symfony. Лістинг основних модулів системи представлено в додатку А.

3.2. Програмна реалізація бази даних

MySQL - вільна реляційна система управління базами даних. База даних забезпечує дозволяє шукати, отримувати, сортувати і зберігати дані. Сервер MySQL управляє доступом до даних, дозволяючи працювати з ними одночасно декільком користувачам, забезпечує швидкий доступ до даних і гарантує надання доступу тільки тим хто має на це право. Він застосовує SQL (Structured Query Language – мова структурованих запитів), що використовується по всьому світу стандартна мова запитів до бази даних [15].

Сервер MySQL постійно працює на комп'ютері. Клієнтські програми (наприклад, скрипти PHP) посилають до сервера MySQL SQL-запити через механізм сокетів (тобто за допомогою мережевих засобів), сервер їх обробляє і запам'ятовує результат. Тобто скрипт (клієнт) вказує, яку інформацію він хоче отримати від сервера баз даних. Потім сервер баз даних посилає відповідь (результат) клієнту (скрипту).

Система керування реляційними базами даних MySQL була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL. У 1999 році MySQL обігнала mSQL за популярністю. Остання версія mSQL- 3.8 - була випущена 9 червня 2006.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблицям, для яких використовувалися ISAM — підпрограми низького рівня. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL.

З часом MySQL все розширювалася і зараз вона — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має впевнену підтримку з боку різноманітних мов програмування.

Станом на квітень 2015 року, MySQL пропонувала версію MySQL 5.6 в двох різних варіантах: з відкритим вихідним кодом MySQL Community Server і комерційний Server Enterprise. Вони мають спільний програмний код і включають в себе серед іншого наступні можливості:

- Крос-платформна підтримка.
- Збережувані процедури та функції.
- Тригери.
- Курсори.
- Оновлювані подання (представлення).
- Інформаційна схема (так званий системний словник, що містить метадані).
- Підтримка SSL.
- Кешування запитів.
- Вкладені запити SELECT.
- Підтримка реплікації.
- Повноцінна підтримка Юнікоду (UTF-8 і UCS2).
- Сегментування таблиць.

MySQL являється компактним багатопоточним сервером баз даних та характеризується великою швидкістю, стійкістю і простотою використання.

Ця система вважається гарним рішенням для малих і середніх додатків. Вихідний код сервера компілюється на безлічі платформ. Найбільш повно можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому. Для некомерційного використання MySQL є безкоштовним.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Недоліками сервера MySQL залишається не реалізована підтримка транзакцій, замість якої пропонується використовувати LOCK/UNLOCK TABLE, а також відсутність підтримки для зовнішніх (foreign) ключів, тригерів, збережених процедур та представлень (VIEW). Та для розробки малих та середніх інформаційних систем, призначених для використання в межах робочих груп ці недоліки є фактично невідчутними.

У текстовому режимі робота з базою даних виглядає просто як введення команд у командний рядок, а результати вибірок повертаються у вигляді своєрідних таблиць, поля в яких налазять один на одного, якщо дані не вміщаються на екрані.

На рисунку 3.3 представлено реалізацію відношення profile в середовищі MySQL та його відповідну DDL інтерпретацію.

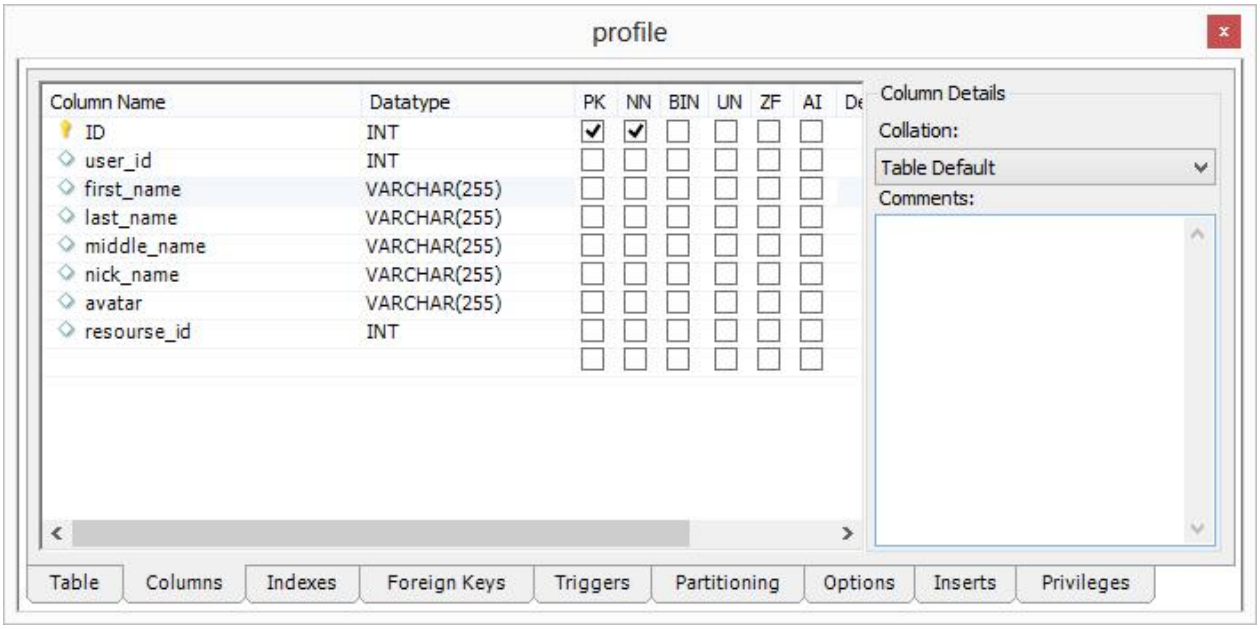

Рис. 3.3. Реалізація відношення profile в середовищі MySQL

```
CREATE  TABLE IF NOT EXISTS `mydb`.`profile` (
  `ID` INT NOT NULL ,
  `user_id` INT NULL ,
  `first_name` VARCHAR(255) NULL ,
  `last_name` VARCHAR(255) NULL ,
  `middle_name` VARCHAR(255) NULL ,
  `nick_name` VARCHAR(255) NULL ,
  `avatar` VARCHAR(255) NULL ,
  `resourse_id` INT NULL ,
  PRIMARY KEY (`ID`) )
ENGINE = InnoDB
```

Висновки до розділу 3

У даному озділі обґрунтовано технологію, мову програмування та розроблено програмну систему. Обґрунтовано засоби розробки бази даних та створено програмну реалізацію бази даних програмної системи за допомогою механізму збережених процедур.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1. Тестування

Для оцінки якості отриманої в результаті роботи соціальної мережі було проведено її порівняння з існуючими аналогами по загальній ступеня зручності використання, для якої в сучасній літературі з програмної інженерії застосовується термін «юзабіліті (usability)».

Відповідно до стандарту ISO-9241, основними критеріями юзабіліті є [9]:

• ефективність - точність і повнота, з якою користувачі досягають поставлених цілей;

• продуктивність - ресурси, які користувач витрачає, щоб з точністю і повнотою досягти поставлених цілей;

• задоволеність користувача - комфорт і прийнятність використання.

Аналогами, з якими порівнювалася розроблена нами система, були обрані найбільші соціальні мережі: FaceBook.com, Myspace.com. Попередні оцінки дають підставу вважати, що розроблена соціальна мережа може успішно конкурувати з існуючими аналогами. Багато користувачів відмітили нові гнучкі можливості і високо оцінили ефективність. Практично всі відмітили зручність користування інтерфейсом, що відповідає критерію задоволеності користувача.

Разом з тим були виявлені деякі недоліки, пов'язані, в першу чергу, з новизною і недостатньою налагодженістю розробленої соціальної мережі. Над виправленням цих недоліків буде вестися подальша робота, і система буде постійно покращуватися.

У процесі промислової експлуатації передбачається провести більш поглиблене експертне оцінювання створеного ресурсу і отримання статистично достовірних результатів.

Тестування веб-орієнтованих систем - один з важливих життєвих етапів, після якого, надається замовнику готовий проект без помилок, з хорошою читабельністю, яка сприймається легкістю, зручністю і надійністю. Тестування - це відхилення фактичного результату від очікуваного, іншими словами - це процес пошуку багів (помилок).

Основні правила тестування веб-сайтів - це кроки, які показують користувачеві, наскільки зручний і логічний буде проект, наскільки просто і можливо знайти ту чи іншу інформацію. Чи добре сприймається людському погляду і правильно працює весь функціонал даного сайту, який був поставлений по ТЗ - це основні показники для тестувальника.

Тестування може відбуватися самими різними способами, однак не варто забувати про сам процес і стратегії тестування. Від нього залежить послідовність ваших дій. На сьогоднішній день, фахівці з тестування веб-сайтів застосовують такі види тестування як:

- функціональне тестування;
- тестування зручності користування (юзабіліті);
- тестування продуктивності;
- тестування інтерфейсу користувача (UI testing);
- тестування безпеки.

Тестування системи надання юридичних консультацій відбувалось за допомогою домашнього сервера Apache, який входить в збірку Denwer. Також для тестування використовувались браузери Google Chrome та Mozilla Firefox на операційній системі Linux та Windows 7.

Для початку перевірки системи встановлено збірку Denwer, яка містить у собі локальний сервер Apache, інтерпретатор PHP5, та систему керування базами даних phpMyAdmin. Потім потрібно включити локальний сервер і запустити браузер для перевірки роботи сайту. Якщо сайт не загрузився, то потрібно перевірити правильність вводу адреси, та перевірити присутність на сервері файл index.php.

Тестування проводилось на таких браузерах як Google Chrome, Mozilla Firefox, Internet Explorer.

Після проведеного тестування системи різними браузерами контент і інтерфейс відображався правильно та без помилок. Всі основні модулі сайту були відображені коректно та в робочому стані.

Розробники-початківці досить рідко роблять тестування навантаження сайтів і веб-додатків. І буває так, що при збільшені онлайну сайт в найбільш відповідальний момент починає гальмувати.

Для тестування нашого сайту використано програму Apache JMeter. JMeter є дуже потужним інструментом навантажувального тестування з можливістю створення великої кількості запитів одночасно завдяки паралельній роботі на декількох комп'ютерах. Підтримує плагіни, за допомогою яких можна значно розширити функціонал. Завантажити JMeter можна за посиланням з офіційного сайту, для запуску потрібно розпакувати і запустити \bin\jmeter.

Запустивши програму, ліворуч бачимо 2 пункти – Test Plan і WorkBench. Додамо Thread Group в Test plan, що дозволить керувати потоками (тобто, симулювати кроситувачів). Для цього потрібно натиснути правою клавішею на Test plan – Add – Threads (Users) – Thread group.

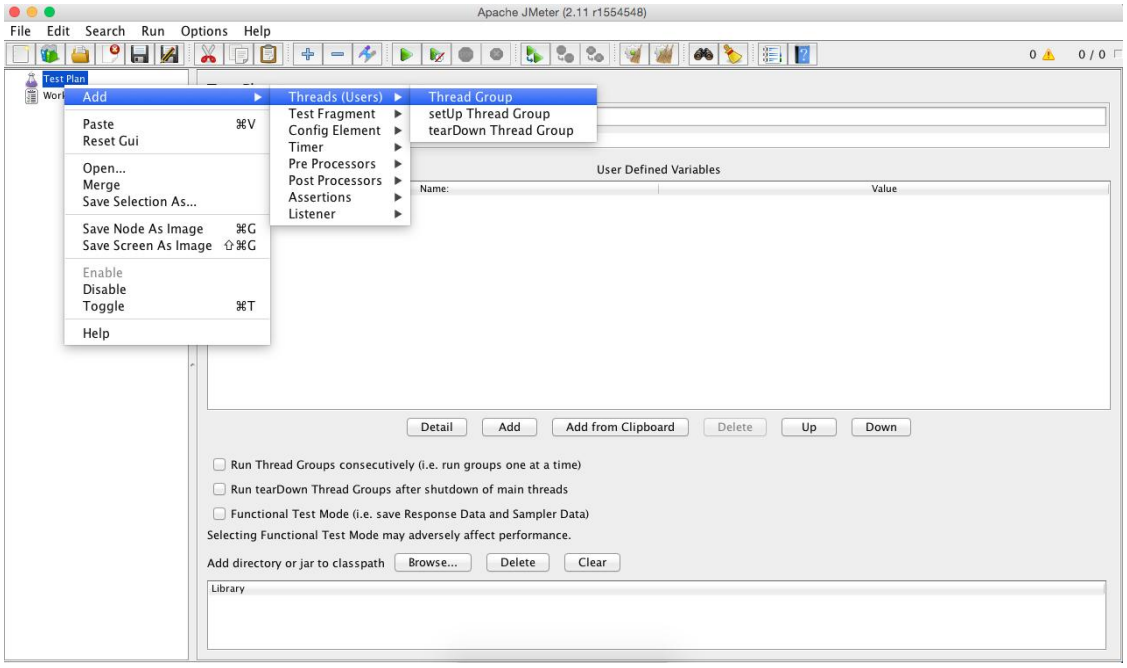

Рис. 4.1. Створення тесту в JMeter

Додамо HTTP Request Defaults, який призначений для налаштування запиту для тестування. Зробити це можна, натиснувши правою кнопкою миші по Thread Group і обравши Add – Config Element – HTTP Request Defaults (рисунок 4.2).

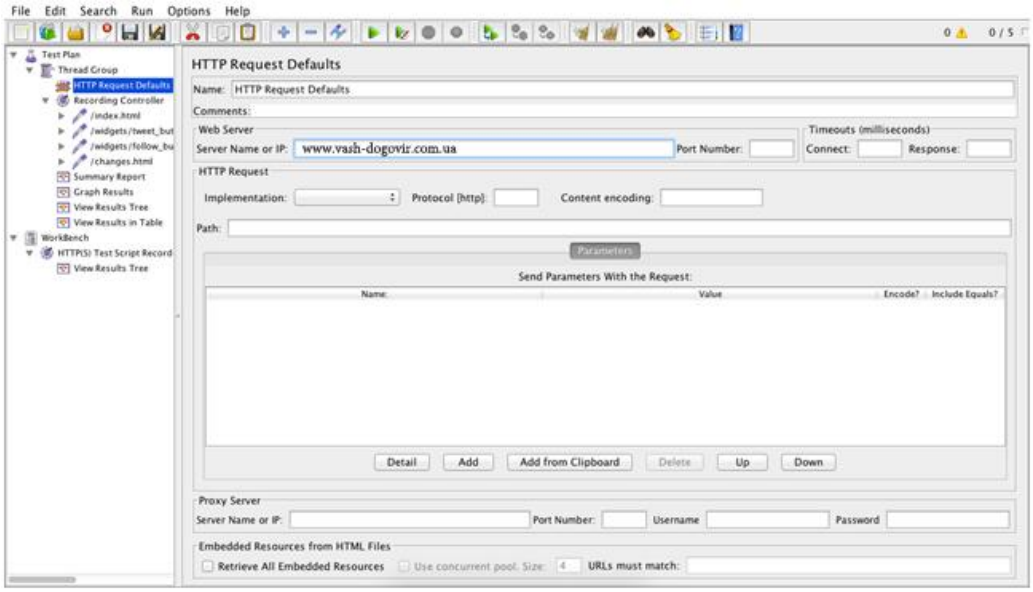

Рис. 4.2. Налаштування запиту для тестування

Нарешті ми можемо розпочати запис. Для цього в браузері перейдемо за адресою сайту і виконаємо певну активність (попереходимо між сторінками, поклікаємо на різні кнопки тощо). Цей сайт використовує кешування інформації для авторизації, тому немає необхідності застосовувати плагін JMeter-а для цього.

Recording Controller записав нашу активність (більш докладно можна подивитися в View Results Tree) і ми майже готові розпочати тестування навантаження.

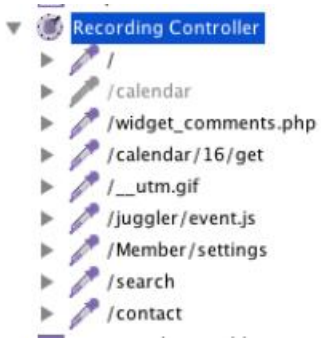

Рис. 4.3. Налаштування Recording Controller

Лише додамо елементи для перегляду результатів Summary Report (Thread Group – Add – Listener – Summary Report) і Graph Results (Thread Group – Add – Listener – Graph Results).

Перейдемо до Thread Group. Виберемо кількість потоків (користувачів) – 5, час приєднання нового користувача встановимо значення 10 (секунд), а в кількість циклів поставимо значення 50.

Перед кожним запуском необхідно виконати наступні дії:

- Ctrl-S для збереження;
- Ctrl-E для очищення минулих результатів;
- Ctrl-R, власне, для запуску (також можна використовувати кнопку Run – Start).

Після виконання тесту можемо подивитися результати. Зайдемо в Graph Results і проаналізуємо (рисунок 4.4).

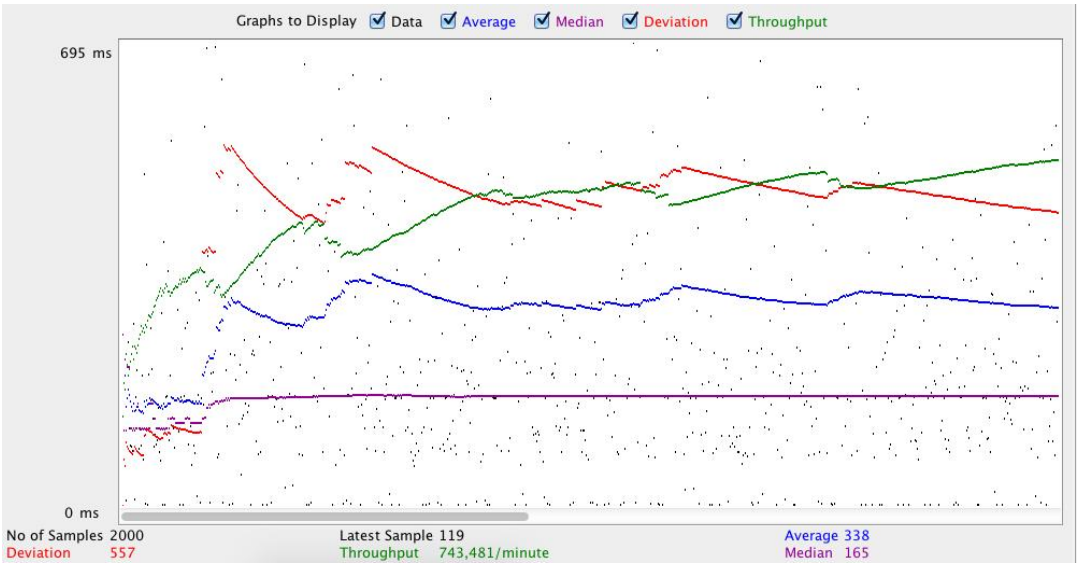

Рис. 4.4. Результати навантажувального тестування

Значення надані в мілісекундах:

- Data – час відповіді кожної окремої одиниці даних тобто кожного перевіреного url.

- Average – усереднений час відгуку, об’єктивний графік зміни навантаження.
- Median – значення медіани (використовується в статистиці).
- Deviation – похибка, стандартне відхилення.
- Throughput – пропускна здатність виконуваних запитів.

Для роботи досить значень Average і Throughput, які відобразять навантаження на веб-сервер і пропускну здатність запитів. За графіком вище видно, що час відповіді приблизно 400мс і не росте, тобто, сервер нормально витримує навантаження в 5 віртуальних користувачів. А ось що вийде, якщо їх буде 50 (рисунок 4.5).

Рис. 4.5. Результати навантажувального тестування

Час відгуку значно збільшується. Summary Report може відобразити статистику по кожному індивідуальному url окремо (рисунок 4.6).

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
/	250	365	80	3358	563,98	0,00%	1,6/sec	5,22	3422,8
/widget_comm...	250	108	60	3314	207,53	0,00%	1,6/sec	9,75	6394,4
/calendar/16/...	250	715	170	4559	822,59	0,00%	1,6/sec	7,11	4666,7
/__utm.gif	250	5	2	47	6,34	0,00%	1,6/sec	0,63	411,0
/juggler/event.js	250	173	0	788	62,62	1,20%	1,6/sec	0,45	295,7
/Member/setti...	250	688	164	4462	760,30	0,00%	1,6/sec	7,10	4664,0
/search	250	346	77	4112	502,32	0,00%	1,6/sec	2,76	1813,3
/contact	250	306	69	2683	399,52	0,00%	1,6/sec	3,30	2166,8
TOTAL	2000	338	0	4559	557,81	0,15%	12,4/sec	36,05	2979,3

Рис. 4.6. Результати навантажувального тестування

У стовпці Average бачимо середній час відгуку, логічно припустити, що чим час більший – тим більше навантаження на даний url. Часто виявляється що є кілька важких сторінок, які в першу чергу і створюють навантаження на систему, і якщо їх оптимізувати, загальна продуктивність багаторазово збільшується (краще всього починати оптимізацію зі сторінок, які за статистикою викликаються часто, а відпрацьовують довго).

View Result in Table відобразить результат у вигляді таблиці, тут вказано час, а також статус (успішно / неуспішно) (рисунок 4.7).

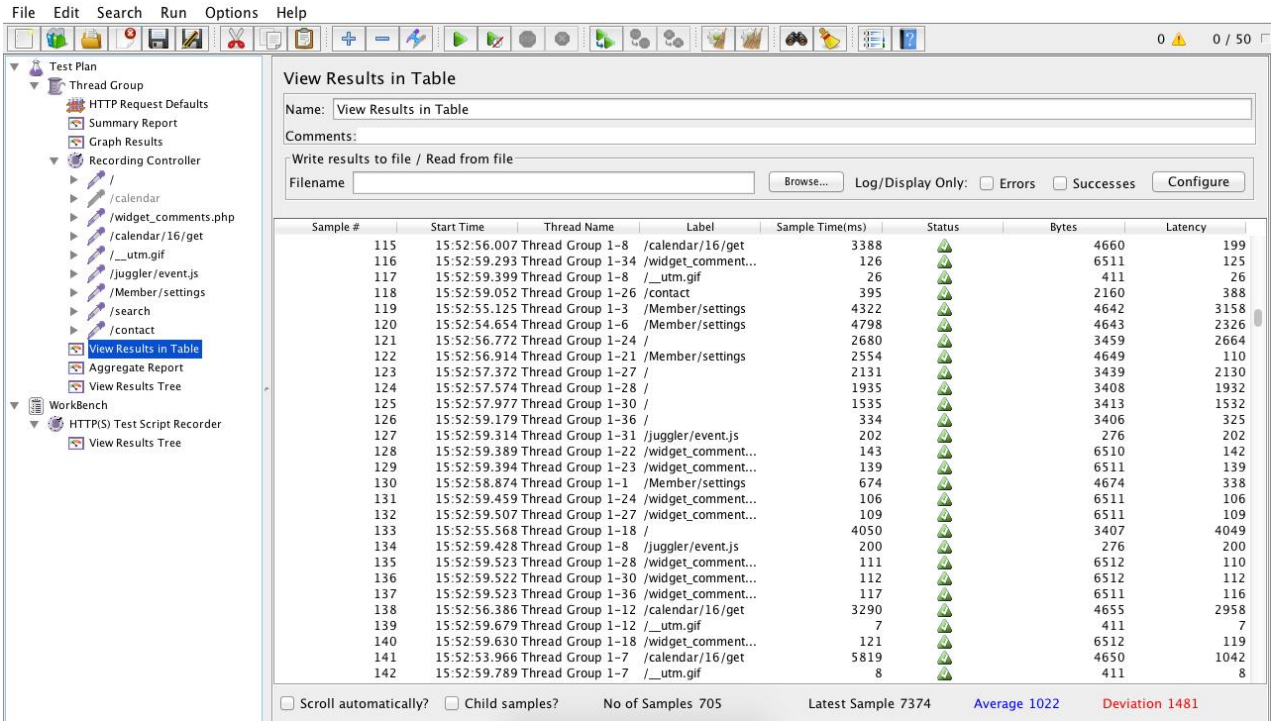

Рис. 4.7. Результати навантажувального тестування

На цьому тестування завершено. В ході нього ми перевірили систему на навантаження та отримали позитивні результати.

4.2. Розгортання програмного продукту

Для розгортання системи необхідний хостинг у гарного провайдера: фізичні файли сайту закачуються по ftp на сервер (рисунок 4.8), за необхідністю

створюється стартова база даних, також проводяться необхідні налаштування, і вже після цього сайт стає доступний будь-якому користувачеві Інтернет.

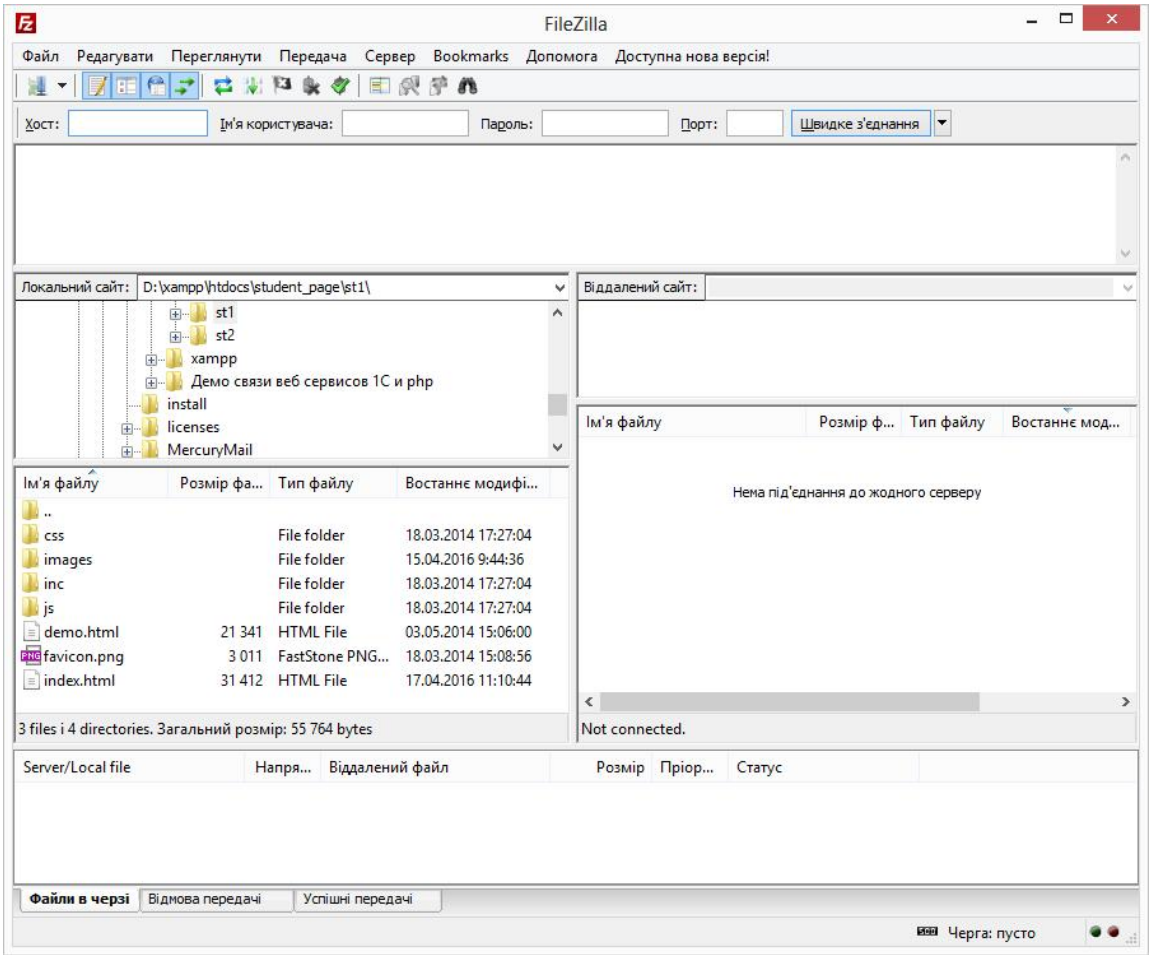

Рис. 4.8. Завантаження системи на сервер

Як правило, встановлення сайту на хостинг проводиться розробником за умови надання замовником доступу до ftp та бази даних, або ж до панелі управління хостингом.

Вимоги до хостингу:

- Apache версії не нижче 3.0 на Unix/linux
- Підтримка PHP версії не нижче 5.0 в повному об'ємі, зокрема відправка поштових повідомлень та відсутність save-mode
- Можливість редагування .htaccess
- Можливість керувати кодуванням (немає одного чітко встановленого кодування)

- В більшості випадків (окрім простих сайтів візиток та ще деяких випадків - уточнюйте перед тим, як обрати хостинг) також база даних Mysql версії не нижче 5.5
- Інколи потрібен Cron - в тих випадках, коли необхідно запускати певні скрипти з певною періодичністю

Втім, більшість провайдерів відповідають зазначеним вимогам, отож при виборі вирішальними чинниками швидше будуть ціна, надійність, регіон та інше.

4.3. Інструкція користувача

Для отримання доступу до функцій соціальної мережі, необхідно виконати вхід в систему. Для цього введіть свій логін і пароль у спеціальну форму (рисунок 4.9). Після введення необхідно натиснути кнопку «Увійти».

Рис. 4.9. Форма входу в систему

Після виконання входу в правому верхньому куті екрану з'явиться зменшена версія вашої фотографії, а також персональне меню.

• Для реєстрації в системі необхідно натиснути на посилання «Реєстрація», розміщене під формою для введення логіна і пароля. На екрані з'явиться форма реєстрації. Введіть своє ім'я, прізвище, псевдонім, адресу електронної пошти (якщо є) і текст, що зображений на рисунку внизу сторінки. Після цього натисніть на кнопку «Реєстрація». В результаті реєстрації вам буде відкритий вхід в систему, створено скриньку електронної пошти, а якщо ви

ввели існуючий адрес в поле для введення - він буде доданий до адрес поштового клієнта.

Перегляд і редагування профілю. Для перегляду профілю користувача необхідно натиснути на його фотографію або ім'я в будь-якому списку людей (списки контактів, учасників спільноти, пошук людей). Перед вами з'явиться фотографія людини, посилання для відправки користувачу повідомлення і додавання його в контакти, список атрибутів профілю (особистої інформації), таких як рідне місто, дата народження, телефон, і т. д., список контактів, набір віджетів, доданих користувачем на сторінку, наприклад, останні записи блогу, останні обговорення (рисунок 4.10). Також, доступні вкладки, що позначають функціональні елементи профілю, такі як фото-архів, блог. Для отримання доступу до них натисніть на відповідну вкладку.

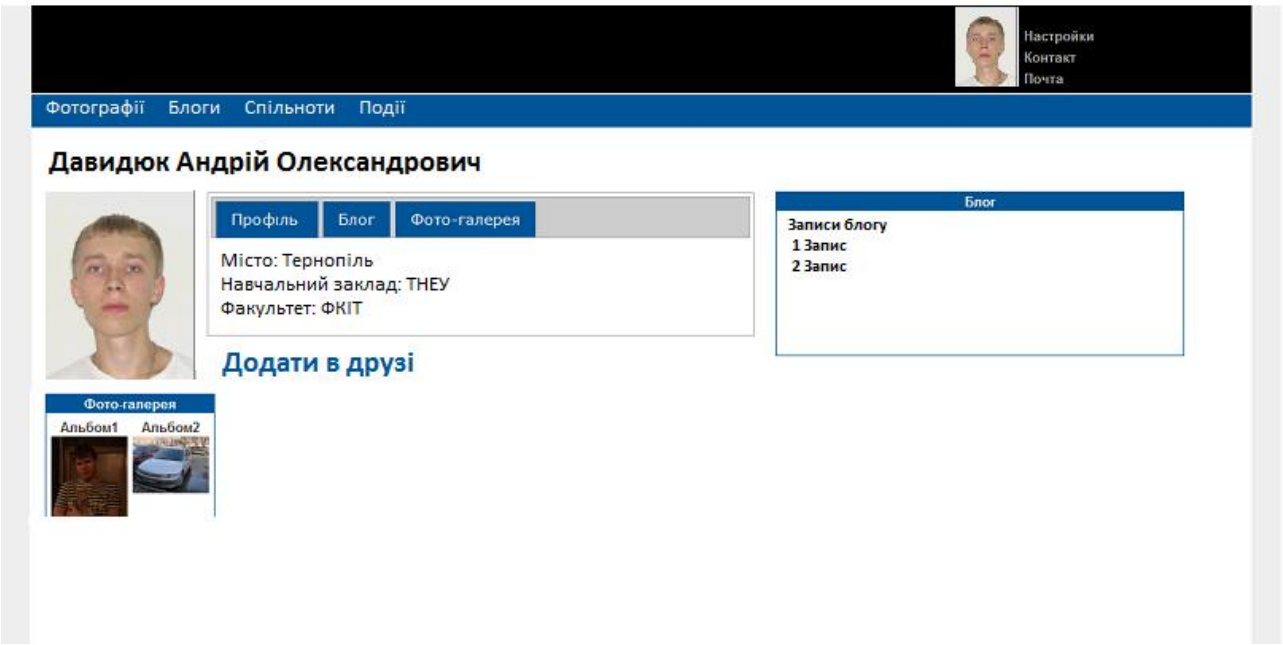

Рис. 4.10. Вид профілю користувача

Якщо ви є господарем профілю, з'являються можливості його редагування. Для входу в режим редагування натисніть на кнопку «Редагувати», розташовану під фотографією профілю. Після цього натисніть на потрібний віджет і перетягніть його мишкою в довільне місце. Виконуйте ці дії для кожного віджета, розташування якого ви хочете поміняти, після чого

натисніть на кнопку «Зберегти», що з'явилася на місці кнопки «Редагувати» (рисунок 4.11). Для зміни списку доступних на сторінці віджетів і вкладок зайдіть в налаштування профілю, посилання на які розташовується в правому верхньому куті сторінки. Поставте галочки необхідні вам плагіни і віджети в секціях і натисніть «Зберегти». Вибрані віджети і плагіни з'являться на вашій сторінці, і ви зможете змінювати їх зовнішній вигляд, як було описано вище.

Для зміни особистої інформації натисніть на іконку редагування, яка поруч з потрібним атрибутом. Після цього, текст атрибута перетвориться на поле для введення. Введіть в це поле нову інформацію і натисніть на іконку зберегти, розташовану поруч з полем.

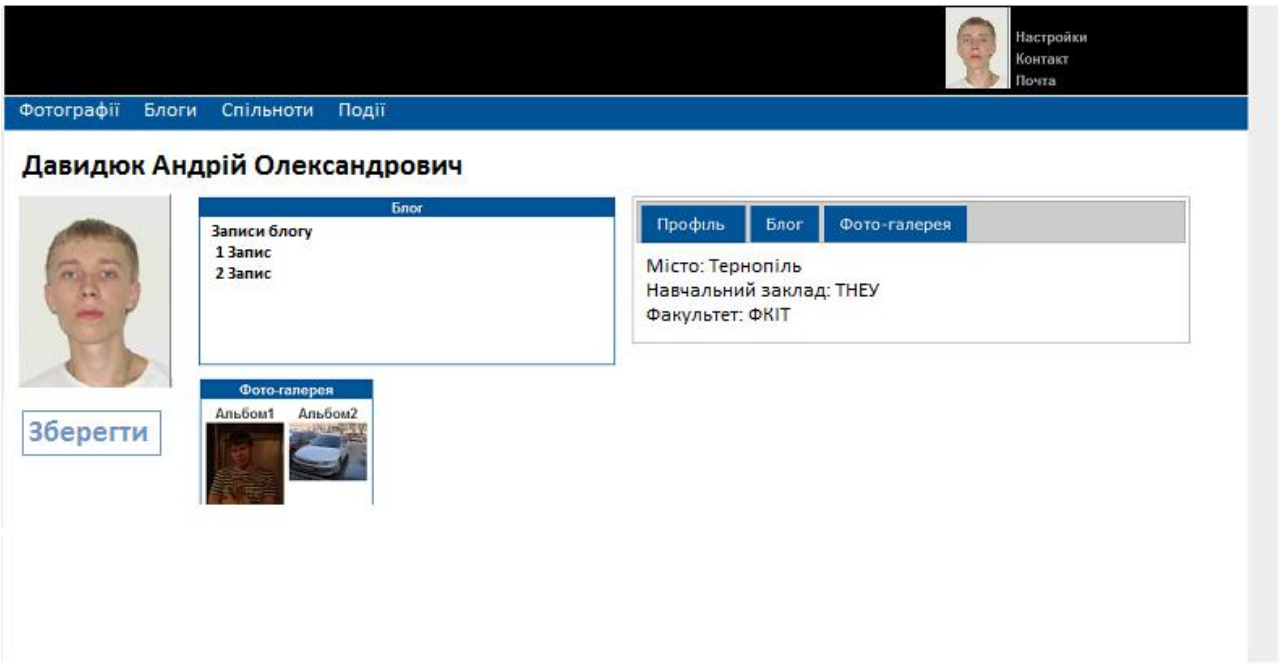

Рис. 4.11. Зміна положення віджетів

Для додавання нового атрибуту в особисту інформацію натисніть на іконку «Плюс», що знаходиться в нижній частині списку атрибутів. З'явиться поле для введення імені атрибута. Після того як ви почнете введення, з'явиться список атрибутів, введених раніше іншими користувачами. Наприклад, якщо ви введете «Мі», в списку підказок з'явиться атрибут «Місто». Після введення

імені, натисніть на іконку, розташовану поруч з полем, для збереження атрибута.

Щоб переглянути список поштових повідомлень, натисніть на лінк «Поштова скринька» в правому верхньому кутку сторінки. З'явиться список повідомлень у вашій поштовій скриньці, а також список папок: вхідні, відправлені, чернетки (рисунок 4.12). При натисканні на папку, в список повідомлень будуть виведені повідомлення з відповідної папки. При натисканні на тему повідомлення - буде показано повідомлення.

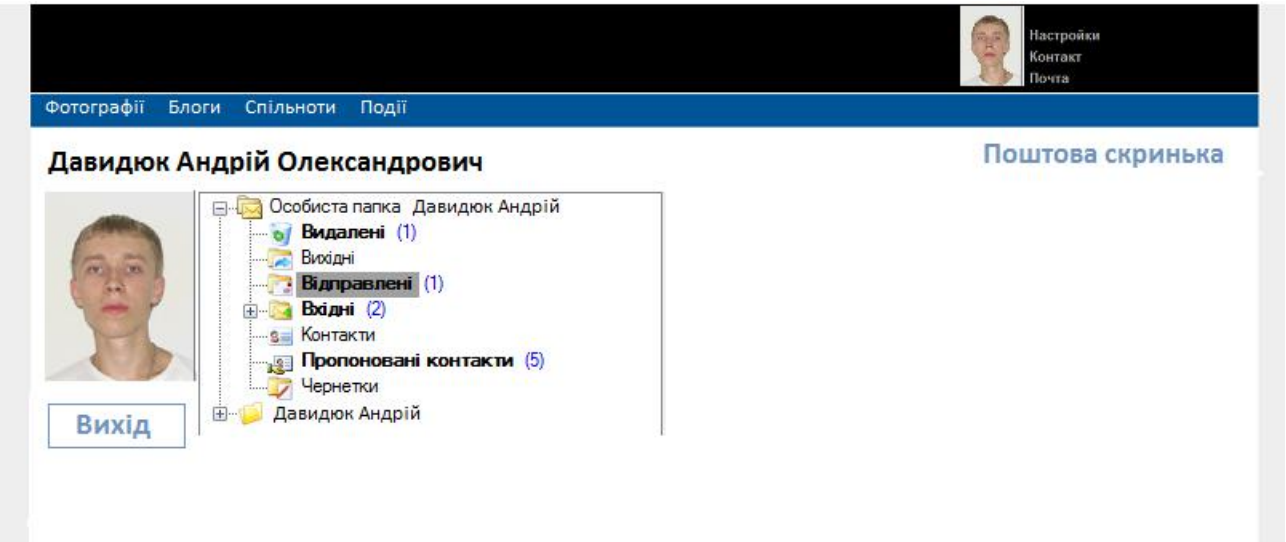

Рис. 4.12. Управління поштою в соціальній мережі

Для відправки повідомлення натисніть на лінк «Надіслати повідомлення», розташований над папками. З'явиться форма для введення теми повідомлення, адресата, тіла листа, кнопки «Прикріпити файл» і «Надіслати» (рисунок 4.12). При введенні в поле адресата, будуть запропоновані знайдені варіанти зі списку контактів. При натисканні на «Прикріпити файл» - буде виведено діалогове вікно для вибору файлів. Виберіть в цьому вікні файл, який ви відправляєте. Завантаження почнеться автоматично, після повного завантаження, долучення буде відображений між полями для введення теми повідомлення і тіла

повідомлення. Ви можете видалити його, натиснувши на іконку «хрестик» поруч з відповідним файлом.

Після введення інформації в форму для відправки, натисніть «Відправити» внизу форми. Після цього, повідомлення буде відправлено адресату.

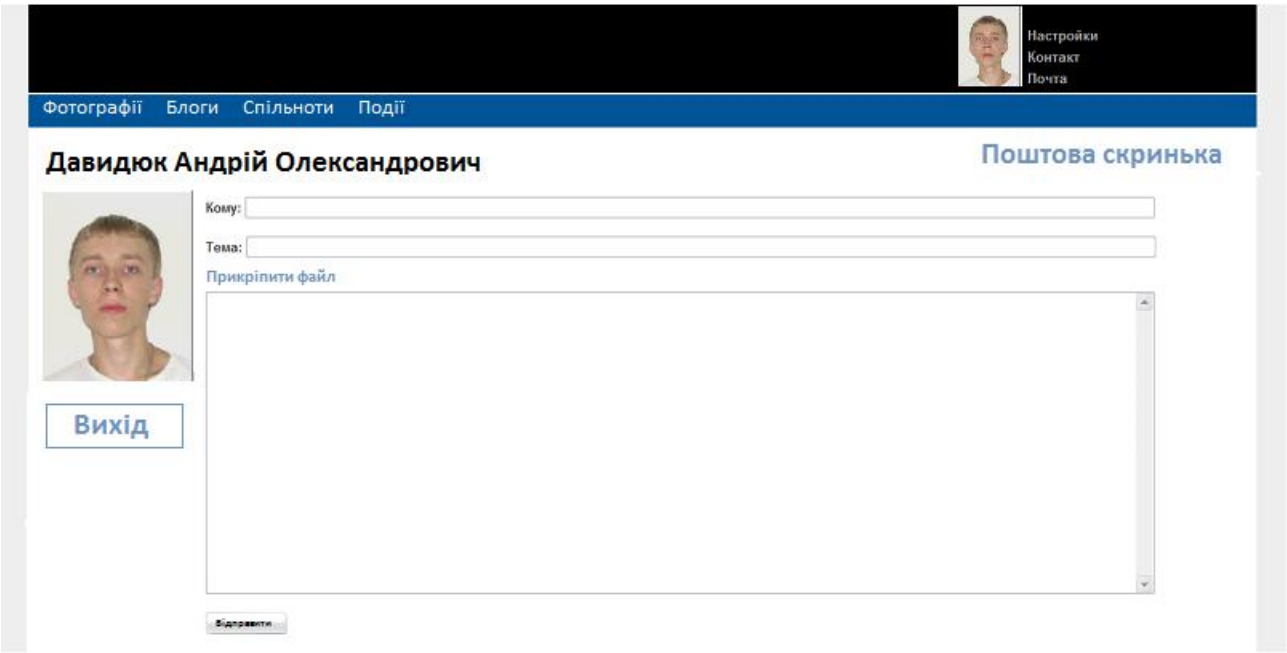

Рис. 4.13. Сторінка відправлення повідомлення

Висновки до розділу 4

Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

ВИСНОВКИ

В результаті роботи над проектом була розроблена робоча версія web-сайту організації спілкування між користувачами соцільної мережі нового покоління. Реалізована базова функціональність, крім того частково реалізовані плагіни: файлове сховище, обговорення, фото-галерея, блог.

Модуль розроблений на базі вільно поширюваного відкритого програмного забезпечення і передбачає можливості розширення.

В ході розробки інформаційної системи дипломної роботи, використовувався найпоширеніший з професійних HTML-редакторів Homesite редактор.

Редактор Homesite 4.5 призначений для тих хто робить web-сторінки вручну, тобто призначений для тих, хто знає HTML. У таких випадках, Ви можете повністю контролювати HTML код, також є можливість зробити зручною свою сторінку в одному з класичних браузерів (MSIE, NN, Opera).

Розроблена система складається з двох функціональних модулів: модуль упорядкування структури виконаний на мові програмування PHP і модуль розробки і показу структури соціальної мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Майк Далворт. Социальные сети. Руководство по эксплуатации, 2010.

2. Сайт Википедия http://ru.wikipedia.org/wiki/Социальная_сеть

3. Джон Коггзалл. PHP 5. Полное руководство. Диалектика, 2006. – 752с.

4. Lincoln Stein, Doug MacEachern. Writing Apache Modules with Perl and C (1st ed.), O'Reilly & Associates, 1999. – 723с.

5. Колисниченко Д.Н. PHP и MySQL. Разработка Web-приложений. 4-е издание. Москва, 2013. – 560с.

6. Алекс Маккоу. Веб-приложения на Javascript. Питер, 2012. – 288с.

7. Andrew Ford. Apache Pocket Reference (1st ed.), O'Reilly & Associates, 2000. – 193с.

8. Бородулин А.Н., Кузнецов В.Н., Мельник М.В. Теория экономического анализа: Учебное пособие. 1-е издание. Тверь: ТГТУ, 2005.

9. Скотт Хокинс. Администрирование Web-сервера Apache и руководство по электронной коммерции. Вильямс. 2001. – 330 с.

10. Сайт Википедия. http://ru.wikipedia.org/wiki/SQL

11. Lars Eilebrecht. Apache Web – Server (3rd ed.), MITP – Verlag GmbH, 2000. – 169с.

12. Mohammed J. Kabir. Apache Administrator's Handbook (1st ed.), IDG Books Worldwide, 1999. – 785с.

13. А. Леоненков. Самоучитель UML (2-е издание), БВХ-Петербург, 2004г. – 432с.

14. Фримен Эрик, Фримен Элизабет Изучаем HTML, XHTML и CSS. Питер. 2012. – 656с.

15. Прохоренок Н. А. HTML, javascript, PHP и MySQL. Джентльменский набор Web-мастера (+ CD-ROM). БХВ-Петербург. 2011. – 912с.

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

Index.php

```
<? Php
/ *
Appointment: Головна сторінка
File: index.php
* /
if (isset ($ _ POST [ "PHPSESSID"])) {
session_id ($ _ POST [ "PHPSESSID"]);
}
@session_start ();
@ob_start ();
@ob_implicit_flush (0);
@error_reporting (E_ALL ^ E_WARNING ^ E_NOTICE);
define ( 'MOZG', true);
define ( 'ROOT_DIR', dirname (__FILE__));
define ( 'ENGINE_DIR', ROOT_DIR. '/ system');
header ( 'Content-type: text / html; charset = utf-8');
// AJAX
$ Ajax = $ _POST [ 'ajax'];
$ Logged = false;
$ User_info = false;
include ENGINE_DIR. '/ init.php';
// Якщо користувач перейшов по реф посиланням, то додаємо ід реферала в сесію
if ($ _ GET [ 'reg']) set_cookie ( 'ref_id', intval ($ _ GET [ 'reg']));
// Визначення браузера
if (stristr ($ _ SERVER [ 'HTTP_USER_AGENT'], 'MSIE 6.0')) $ xBrowser =
'Ie6';
elseif (stristr ($ _ SERVER [ 'HTTP_USER_AGENT'], 'MSIE 7.0')) $ xBrowser
= 'Ie7';
elseif (stristr ($ _ SERVER [ 'HTTP_USER_AGENT'], 'MSIE 8.0')) $ xBrowser
= 'Ie8';
```

```
if ($ xBrowser == 'ie6' OR $ xBrowser == 'ie7' OR $ xBrowser == 'ie8')
header ( "Location: /badbrowser.php");
// Завантажуємо кількість нових новин
$ CacheNews = mozg_cache ( 'user _'. $ User_info [ 'user_id']. '/ New_news');
if ($ CacheNews) {
$ New_news = "<div class = \" ic_newAct \ "> {$ CacheNews} </ div>";
$ News_link = '/ notifications';
}
// Завантажуємо кількість нових подарунків
$ CacheGifts = mozg_cache ( 'user _'. $ User_info [ 'user_id']. '/ New_gifts');
if ($ CacheGifts) {
$ New_gifts = "<div class = \" ic_newAct \ "> {$ CacheGifts} </ div>";
}
// Завантажуємо кількість нових подарунків
$ CacheGift = mozg_cache ( "user _ {$ user_info [ 'user_id']} / new_gift");
if ($ CacheGift) {
$ New_ubm = "<div class = \" ic_newAct \ "> {$ CacheGift} </ div>";
$ Gifts_link = "/ gifts {$ user_info [ 'user_id']}? New = 1";
} else
$ Gifts_link = '/ balance';
// Нові повідомлення
$ User_pm_num = $ user_info [ 'user_pm_num'];
if ($ user_pm_num)
$ User_pm_num = "<div class = \" ic_newAct \ "> {$ user_pm_num} </ div>";
else
$ User_pm_num = '';
// Тех-підтримки +1
$ Supports = $ db-> super_query ( "SELECT COUNT (*) AS cnt FROM
`" .PREFIX. "_ Support` WHERE sfor_user_id");
$ Agent = ($ user_info [ 'user_group'] == '4');
if ($ supports == $ agent) {
$ Supports_owner = "<div class = \" ic_newAct \ "> {$ supports [ 'cnt']} </ div>";
}
if ($ supports [ 'cnt'] == 0) {
```

```
$ Supports_owner = '';
}
if ($ logged) {
 if ($ user_info [ 'user_photo'])
$ Ava = $ config [ 'home_url']. 'Uploads / users /'.$ user_info [' user_id '].' / 100 _ '. $
User_info [' u
ser_photo '];
 else
 $ Ava = '/images/no_ava_50.png';
 $ Myphoto_header. = '<Img src = "'. $ Ava. '" Width = "30" />'."\n ";
 $ Tpl-> set ( '{myphoto_header}', $ myphoto_header);
 $ Tpl-> load_template ( 'main.tpl');
}
//Нові друзі
$ User_friends_demands = $ user_info [ 'user_friends_demands'];
if ($ user_friends_demands) {
$ Demands = "<div
class = \ "ic_newAct \"> {$ user_friends_demands} </ div> ";
$ Requests_link = '/ requests';
} else
$ Demands = '';
// ТП
$ User_support = $ user_info [ 'user_support'];
if ($ user_support)
$ Support = "<div class = \" ic_newAct \ "> {$ user_support} </ div>";
else
$ Support = '';
// Відмітки на фото
if ($ user_info [ 'user_new_mark_photos']) {
$ New_photos_link = 'newphotos';
$ New_photos = "<div
class = \ "ic_newAct \"> ". $ user_info [ 'user_new_mark_photos']." </ div> ";
} Else {
$ New_photos = '';
```

```
$ New_photos_link = $ user_info [ 'user_id'];
}
// Дизайн сторінки
if ($ config [ 'temp'] == 'Old') {
if ($ go! = 'profile' OR $ user_info [ 'mydesign'] == 1 AND
$ Config [ 'temp']! = 'Mobile') {
$ Data_design = xfieldsdataload ($ user_info [ 'user_design']);
$ Tpl-> result [ 'info']. = '<Style type = "text / css" media = "all">';
if ($ data_design [ 'background'])
if ($ data_design [ 'background_repeat'])
$ Tpl-> result [ 'info']. = "Html,
body {background: url ( '/ uploads / users / {$ user_info [' user_id ']} / {$ data_design ['
backgr
ound ']}') no-repeat center center fixed;-webkit-background-size:cover;-mozbackground-
size:cover;-o-background-size:cover;background-size:100%;}";
else
$ Tpl-> result [ 'info']. = "Html,
body {background: url ( '/ uploads / users / {$ user_info [' user_id ']} / {$ data_design ['
backgr
ound ']}'); background-attachment: fixed} ";
if ($ data_design [ 'logo'])
$ Tpl-> result [ 'info']. =
".logo_user {Background: url ( '/ uploads / users / {$ user_info [' user_id ']} / {$ data_design
[' l
ogo ']}') no-repeat;width:67px;height:47px;position:fixed;zindex:99;top:0px;margin-
right:0px}.udinsMy{background:none}";
if ($ data_design [ 'opacity'] <96 AND $ user_info [ 'user_design']
AND $ data_design [ 'opacity'])
$ Tpl-> result [ 'info']. =
"#page {Background: url ( '/ templates / Old / images / {$ data_design [' opacity ']}. Png');
mar
gin: -12px; padding: 12px; margin-bottom: -15px; padding-bottom: 15px} ";
else
$ Tpl-> result [ 'info']. = '#page {Background: #fff; margin: -
```

```
12px; padding: 12px; margin-bottom: -15px; padding-bottom: 15px} ';
if ($ data_design [ 'color_head'] == 1) $ color_head = 'head_red';
elseif ($ data_design [ 'color_head'] == 2) $ color_head =
'Head_orange';
elseif ($ data_design [ 'color_head'] == 3) $ color_head =
'Head_yelllow';
elseif ($ data_design [ 'color_head'] == 4) $ color_head =
'Head_green';
elseif ($ data_design [ 'color_head'] == 5) $ color_head =
'Head_lightblue';
elseif ($ data_design [ 'color_head'] == 6) $ color_head = 'head';
elseif ($ data_design [ 'color_head'] == 7) $ color_head =
'Head_purple';
elseif ($ data_design [ 'color_head'] == 8) $ color_head =
'Head_black';
else $ color_head = 'head';
$ Tpl-> result [ 'info']. =
".head {Background: url ( '/ templates / Old / images / {$ color_head} .png') repeat-x}";
$ Tpl-> result [ 'info']. = '</ Style>';
$ Tpl-> result [ 'content']. = '<Div class = "clear"> </ div>';
}
}
// Запрошення в співтовариства
if ($ user_info [ 'invties_pub_num']) {
$ New_groups = "<div
class = \ "ic_newAct \"> ". $ user_info [ 'invties_pub_num']." </ div> ";;
$ New_groups_lnk = '/ groups? Act = invites';
} Else {
$ New_groups = ";
$ New_groups_lnk = '/ groups';
}
// Якщо включений AJAX то завантажуємо стор.
if ($ ajax == 'yes') {
$ Speedbar = addslashes ($ speedbar);
```

```
// Якщо є POST Запит і значення AJAX, а $ ajax Не рівняється
"Yes" щось не пропускаємо
if ($ _ SERVER [ 'REQUEST_METHOD'] == 'POST' AND $ ajax! =
'Yes')
die ( 'Невідома помилка');
if ($ spBar)
68
$ AjaxSpBar = "$ ( '# speedbar'). Show (). Html ( '{$ speedbar}')";
else
$ AjaxSpBar = "$ ( '# speedbar'). Hide ()";
$ Rur = '<b>'. DeColNums ($ user_info [ 'balance_rub']). ' К3Т. </ b> <br
/><b>'.deColNums($user_info['user_balance ']).' mix </ b> ';
$ Result_ajax = <<< HTML
<Script type = "text / javascript">
document.title = '{$ metatags [' title ']}';
{$ AjaxSpBar};
document.getElementById ( 'new_msg'). innerHTML = '{$ user_pm_num}';
document.getElementById ( 'new_gifts'). innerHTML = '{$ new_gifts}';
document.getElementById ( 'new_news'). innerHTML = '{$ new_news}';
document.getElementById ( 'new_ubm'). innerHTML = '{$ new_ubm}';
document.getElementById ( 'ubm_link'). setAttribute ( 'href', '{$ gifts_link}');
document.getElementById ( 'new_support'). innerHTML = '{$ support}';
document.getElementById ( 'news_link'). setAttribute ( 'href',
'/ News {$ news_link}');
document.getElementById ( 'new_requests'). innerHTML = '{$ demands}';
document.getElementById ( 'new_photos'). innerHTML = '{$ new_photos}';
document.getElementById ( 'requests_link_new_photos'). setAttribute ( 'href',
'/ Albums / {$ new_photos_link}');
document.getElementById ( 'requests_link'). setAttribute ( 'href',
'/ Friends {$ requests_link}');
document.getElementById ( 'new_support_owner'). innerHTML =
'$ {$ Supports_owner}';
$ ( '# UpBal'). Html ( '{$ rur}');
$ ( '# New_groups'). Html ( '{$ new_groups}');
```

```
$ ( '# New_groups_lnk'). Attr ( 'href', '{$ new_groups_lnk}');
</ Script>
{$ Tpl-> result [ 'info']} {$ tpl-> result [ 'content']}
HTML;
echo str_replace ( '{theme}', '/templates/'.$config['temp'], $ result_ajax);
$ Tpl-> global_clear ();
$ Db-> close ();
if ($ config [ 'gzip'] == 'yes')
GzipOut ();
die ();
}
69
// Якщо звернення до модуля реєстрації або головного і користувач не
авторизований то показуємо реєстрацію
if ($ go == 'main' AND! $ logged)
include ENGINE_DIR. '/ modules / register_main.php';
$ Tpl-> load_template ( 'main.tpl');
// Тих-підтримки +1
if ($ supports)
$ Tpl-> set ( '{supports-owner}', $ supports_owner);
else
$ Tpl-> set ( '{supports-owner}', '');
// Якщо користувач залогінився
if ($ logged) {
$ Tpl-> set_block ( '' \\ [not-logged \\] (. *?) \\ [/ Not-logged \\]' si'', "");
$ Tpl-> set ( '[logged]', '');
$ Tpl-> set ( '[/ logged]', '');
$ Check_short_link = $ db-> super_query ( "SELECT` short_link` FROM
`". PREFIX. "_users` WHERE` user_id` = ". $ User_info [ 'user_id']);
if ($ check_short_link [ 'short_link']! = null &&
$ Check_short_link [ 'short_link']! = 'Empty') {
$ Tpl-> set ( '{my-page-link}', '/'.$check_short_link['short_link']);
} Else {
$ Tpl-> set ( '{my-page-link}', '/u'.$user_info['user_id']);
```

```
}
$ Tpl-> set ( '{my-id}', $ user_info [ 'user_id']);
// Вантаження Фону
$ Row_fon = $ db-> super_query ( "SELECT user_img_fon FROM
`" .PREFIX. "_ Users` WHERE user_id = '{$ user_id}'");
 if ($ row_fon [ 'user_img_fon']) {
 $ Tpl-> set ( '{fon_facemy}', $ row_fon [ 'user_img_fon']);
 } Else {
 $ Tpl-> set ( '{fon_facemy}', '{theme} /images/lot.jpg');
 }
// Заявки в друзі
$ User_friends_demands = $ user_info [ 'user_friends_demands'];
if ($ user_friends_demands) {
$ Tpl-> set ( '{demands}', $ demands);
$ Tpl-> set ( '{requests-link}', $ requests_link);
70
} Else {
$ Tpl-> set ( '{demands}', '');
$ Tpl-> set ( '{requests-link}', '');
}
// Новини
if ($ CacheNews) {
$ Tpl-> set ( '{new-news}', $ new_news);
$ Tpl-> set ( '{news-link}', $ news_link);
} Else {
$ Tpl-> set ( '{new-news}', '');
$ Tpl-> set ( '{news-link}', '');
}
// Повідомлення
if ($ user_pm_num)
$ Tpl-> set ( '{msg}', $ user_pm_num);
else
$ Tpl-> set ( '{msg}', '');
// Підтримка
```

```
if ($ user_support)
$ Tpl-> set ( '{new-support}', $ support);
else
$ Tpl-> set ( '{new-support}', '');
// Відмітки на фото
if ($ user_info [ 'user_new_mark_photos']) {
$ Tpl-> set ( '{my-id}', 'newphotos');
$ Tpl-> set ( '{new_photos}', $ new_photos);
} else
$ Tpl-> set ( '{new_photos}', '');
// UBM
if ($ CacheGift) {
$ Tpl-> set ( '{new-ubm}', $ new_ubm);
$ Tpl-> set ( '{ubm-link}', $ gifts_link);
} Else {
$ Tpl-> set ( '{new-ubm}', '');
$ Tpl-> set ( '{ubm-link}', $ gifts_link);
}
// подарунки
if ($ CacheGifts)
71
$ Tpl-> set ( '{new-gifts}', $ new_gifts);
else
$ Tpl-> set ( '{new-gifts}', '');
// Запрошення в співтовариства
if ($ user_info [ 'invties_pub_num']) {
$ Tpl-> set ( '{groups-link}', $ new_groups_lnk);
$ Tpl-> set ( '{new_groups}', $ new_groups);
} Else {
$ Tpl-> set ( '{groups-link}', $ new_groups_lnk);
$ Tpl-> set ( '{new_groups}', '');
}
} Else {
$ Tpl-> set_block ( " '\\ [logged \\] (. *?) \\ [/ Logged \\]' si", "");
```

```
$ Tpl-> set ( '[not-logged]', '');
$ Tpl-> set ( '[/ not-logged]', '');
$ Tpl-> set ( '{my-page-link}', '');
}
$ Tpl-> set ( '{header}', $ headers);
$ Tpl-> set ( '{speedbar}', $ speedbar);
$ Tpl-> set ( '{mobile-speedbar}', $ mobile_speedbar);
$ Tpl-> set ( '{info}', $ tpl-> result [ 'info']);
// FOR MOBILE VERSION 1.0
if ($ config [ 'temp'] == 'mobile') {
$ Tpl-> result [ 'content'] = str_replace ( 'onClick = "Page.Go (this.href);
return false " ',' ', $ tpl-> result [' content ']);
if ($ user_info [ 'user_status'])
$ Tpl-> set ( '{status-mobile}', '<span style = "fontsize: 11px; color: # 000">'. $ User_info [
'user_status']. '</ Span>');
else
$ Tpl-> set ( '{status-mobile}', '<span style = "fontsize: 11px; color: # 999"> встановити
статус </ span> ');
$ New_actions =
$ User_friends_demands + $ user_support + $ CacheNews + $ CacheGift + $ user_info [
'user
72
_pm_num '];
if ($ new_actions)
$ Tpl-> set ( '{new-actions}', "<div class = \" headm_newac \ "
style = \ "margin-top: 5px; margin-left: 30px \"> + {$ new_actions} </ div> ");
else
$ Tpl-> set ( '{new-actions}', "");
}
$ Tpl-> set ( '{content}', $ tpl-> result [ 'content']);
if ($ spBar)
$ Tpl-> set_block ( " '\\ [speedbar \\] (. *?) \\ [/ Speedbar \\]' si", "");
else {
$ Tpl-> set ( '[speedbar]', '');
```

```
$ Tpl-> set ( '[/ speedbar]', '');
}
$ Obshenie = $ db-> super_query ( "SELECT user_id, text FROM
`" .PREFIX. "_ Obshenie` WHERE date> 'NOW () - 604800' ORDER BY RAND ()
LIMIT 1 ");
if ($ obshenie) {
$ Avatar_obshenie = $ db-> super_query ( "SELECT user_photo,
user_search_pref FROM `" .PREFIX. "_ users` WHERE user_id =
'{$ Obshenie [' user_id ']}' ");
$ Tpl-> set ( '{avatar_obshenie}', '<a href="u'.$obshenie['user_id'].'"> <img
src = "/ uploads / users /'.$ obshenie [ 'user_id']. '/ 100 _'. $ avatar_obshenie [ 'user_photo'].
'" /
> </a> ');
$ Tpl-> set ( '{name_obshenie}', '<a
href = "u '. $ obshenie [' user_id '].'"> '. $ avatar_obshenie [' user_search_pref '].' </a> ');
$ Tpl-> set ( '{text}', $ obshenie [ 'text']);
$ Tpl-> set ( '[obshenie]', '');
$ Tpl-> set ( '[/ obshenie]', '');
} Else {
$ Tpl-> set_block ( " '\\ [obshenie \\] (. *?) \\ [/ Obshenie \\]' si", "");
}
// BUILD JS
if ($ logged)
$ Tpl-> set ( '{js}', '<script type = "text / javascript"
73
src = "{theme} /js/jquery.lib.js"> </ script>
<Script type = "text / javascript"
src = "{theme} / js /'.$ checkLang. '/ lang.js"> </ script>
<Script type = "text / javascript" src = "{theme} /js/main.js?3"> </ script>
<Script type = "text / javascript" src = "{theme} /js/profile.js?125"> </ script> ');
else
$ Tpl-> set ( '{js}', '<script type = "text / javascript"
src = "{theme} /js/jquery.lib.js"> </ script>
<Script type = "text / javascript"
```

```
src = "{theme} / js /'.$ checkLang. '/ lang.js"> </ script>
<Script type = "text / javascript" src = "{theme} /js/main.js?2"> </ script> ');
// FOR MOBILE VERSION 1.0
if ($ user_info [ 'user_photo']) $ tpl-> set ( '{my-ava}',
"/ Uploads / users / {$ user_info [ 'user_id']} / 50 _ {$ user_info [ 'user_photo']}");
else $ tpl-> set ( '{my-ava}', "{theme} /images/no_ava_50.png");
$ Tpl-> set ( '{my-name}', $ user_info [ 'user_search_pref']);
if ($ check_smartphone) $ tpl-> set ( '{mobile-link}', '<a
href = "/ index.php? act = change_mobile"> мобільна версія </a> ');
else $ tpl-> set ( '{mobile-link}', '');
if ($ _ SESSION [ 'skin'])
$ Tpl-> set ( '{design}', 'перейти на новий дизайн');
else
$ Tpl-> set ( '{design}', 'перейти на старий дизайн');
$ Tpl-> set ( '{balance-mix}', deColNums ($ user_info [ 'user_balance']));
$ Tpl-> set ( '{balance-rub}', deColNums ($ user_info [ 'balance_rub']));
// Банери
$ Tpl-> set ( '{banner-top}', '');
$ Tpl-> set ( '{banner-bottom}', '');
$ Tpl-> set ( '{banner-right-1}', '');
$ Tpl-> set ( '{banner-right-2}', '');
$ Tpl-> set ( '{banner-right-3}', '');
if ($ logged) {
if ($ user_info [ 'banner_cat']) {
$ Banner_cat = "AND cat = '{$ user_info [' banner_cat ']}'";
}
74
$ Sql_banners = $ db-> super_query ( "SELECT id, user_id, pos, img, title,
descr, link FROM `" .PREFIX. "_ users_banners` WHERE approve = '0'
{$ Banner_cat} ", 1);
foreach ($ sql_banners as $ rowB) {
$ RowB [ 'title'] = stripslashes ($ rowB [ 'title']);
$ RowB [ 'descr'] = stripslashes ($ rowB [ 'descr']);
if ($ rowB [ 'pos'] == 1) {
```

```
$ Tpl-> set ( '{banner-top}', '<a href = "'. $ RowB [ 'link']. "' Target = "_ blank"
onClick = "recForBannerStat ( '. $ rowB [' id '].')"> <img
src = "/ uploads / mybanners /'.$ rowB [ 'user_id']. '/ ok /'.$ rowB [' img '].'" width = "880"
height = "150" title = " '. $ rowB [' title '].' - '. $ RowB [' descr '].' "/> </a> ');
} Elseif ($ rowB [ 'pos'] == 2) {
$ Tpl-> set ( '{banner-bottom}', '<a href = "'. $ RowB [ 'link']. "' Target = "_ blank"
onClick = "recForBannerStat ( '. $ rowB [' id '].')"> <img
src = "/ uploads / mybanners /'.$ rowB [ 'user_id']. '/ ok /'.$ rowB [' img '].'" width = "880"
height = "150" style = "margin-left: 18px" title = " '. $ rowB [' title '].' - '. $ RowB [' descr '].'
"
/> </a> ');
} Elseif ($ rowB [ 'pos'] == 3 AND! $ _ SESSION [ 'banner13']) {
$ Tpl-> set ( '{banner-right-1}', '<div id = "Xbanner13"> <a
href = " '. $ rowB [' link '].'" target = "_ blank" onClick = "recForBannerStat ( '. $ rowB [' id
'].')"
style = "text-decoration: none">
<Div style = "background: #fff; padding: 10px" class = "border_radius_5
margin_bottom_10 ">
<B> '. $ RowB [' title '].' </ B> <br />
<Img src = "/ uploads / mybanners /'.$ rowB [ 'user_id']. '/ Ok /'.$ rowB [' img '].'"
width = "65" height = "90" style = "margin-top: 5px; margin-bottom: 5px" /> <br />
<Span style = "color: # 000; font-size: 10px"> '. $ RowB [' descr '].' </ Span> <br />
<a class="cursor_pointer" onClick="bannerHide(13)" style="color:#777;fontsize:10px">
Приховати </a>
</ Div>
</a> </ div> ');
} Elseif ($ rowB [ 'pos'] == 4 AND! $ _ SESSION [ 'banner14']) {
$ Tpl-> set ( '{banner-right-2}', '<div id = "Xbanner14"> <a
75
href = " '. $ rowB [' link '].'" target = "_ blank" onClick = "recForBannerStat ( '. $ rowB [' id
'].')"
style = "text-decoration: none">
<Div style = "background: #fff; padding: 10px" class = "border_radius_5
margin_bottom_10 ">
```

```
<B> '. $ RowB [' title '].' </ B> <br />
<Img src = "/ uploads / mybanners /'.$ rowB [ 'user_id']. '/ Ok /'.$ rowB [' img '].'"
width = "65" height = "90" style = "margin-top: 5px; margin-bottom: 5px" /> <br />
<Span style = "color: # 000; font-size: 10px"> '. $ RowB [' descr '].' </ Span> <br />
<a class="cursor_pointer" onClick="bannerHide(14)" style="color:#777;fontsize:10px">
Приховати </a>
</ Div>
</a>
</ Div> ');
} Elseif ($ rowB [ 'pos'] == 5 AND! $ _ SESSION [ 'banner15']) {
$ Tpl-> set ( '{banner-right-3}', '<div id = "Xbanner15"> <a
href = " '. $ rowB [' link '].'" target = "_ blank" onClick = "recForBannerStat ( '. $ rowB [' id
'].')"
style = "text-decoration: none">
<Div style = "background: #fff; padding: 10px" class = "border_radius_5
margin_bottom_10 ">
<B> '. $ RowB [' title '].' </ B> <br />
<Img src = "/ uploads / mybanners /'.$ rowB [ 'user_id']. '/ Ok /'.$ rowB [' img '].'"
width = "65" height = "90" style = "margin-top: 5px; margin-bottom: 5px" /> <br />
<Span style = "color: # 000; font-size: 10px"> '. $ RowB [' descr '].' </ Span> <br />
<a class="cursor_pointer" onClick="bannerHide(15)" style="color:#777;fontsize:10px">
Приховати </a>
</ Div>
</a>
</ Div> ');
}
}
}
$ Tpl-> set ( '{lang}', $ rMyLang);
$ Tpl-> compile ( 'main');
echo str_replace ( '{theme}', '/templates/'.$config['temp'], $ tpl-> result [ 'main']);
$ Tpl-> global_clear ();
76
$ Db-> close ();
```

```
if ($ config [ 'gzip'] == 'yes')
GzipOut ();
?>
```