МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук

Кафедра інформаційних технологій

ДИПЛОМНА РОБОТА

Рівень вищої освіти бакалавр

на тему: Розробка веб-орієнтованої системи «Nodesi»

Виконала студентка 4 курсу групи К-42
Напрям підготовки 6.050101
комп'ютерні науки
Скорик Ірина Ігорівна

Керівник к.т.н., доц.,
Терещенко Тетяна Михайлівна

Консультант

Рецензент к.т.н.,доц.,
Гнатовська Ганна Арнольдівна

Одеса 2017

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОСИЛАНЬ 5
ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Дослідження предмету, цілей та особливостей веб-системи для публікації свого портфоліо 8
1.2 Аналіз існуючих аналогів 9
1.2.1 Сімейство блогів «Tumblr» 9
1.2.2 Онлайн-портфоліо «Behance» 11
1.2.3 Спільнота дизайнерів «Dribbble» 12
2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ 15
2.1 Вибір архітектури для розробки веб-системи 15
2.2 Вибір мови для програмування веб-систем 17
2.2.1 Мова програмування Java 18
2.2.2 Мова програмування Python 21
2.2.3 Мова програмування PHP 23
2.3 Вибір СКБД для управління контентом веб-системи 26
2.3.1 СКБД PostgreSQL 27
2.3.2 СКБД Microsoft SQL Server 29
2.3.3 СКБД MySQL 31
2.4 Веб-сервер Apache HTTP Server 32
3 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 35
3.1 Загальні вимоги до інформаційної системи 35
3.2 Функціональні можливості користувачів системи «Nodesi» 35
3.3 Проектування бази даних веб-орієнтованої системи «Nodesi» 39
3.3.1 Опис таблиць бази даних веб-системи «Nodesi» 40
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 44
4.1 Схема роботи веб-орієнтованої системи «Nodesi» 44
4.2 Реалізація структури веб-орієнтованої системи онлайн-портфоліо 44
4.3 Розробка інтерфейсу користувача веб-системи 46
ВИСНОВКИ 49
ПЕРЕЛІК ПОСИЛАНЬ 50
Д О Д А Т К И 51
ДОДАТОК А СХЕМА ФУНКЦІОНУВАННЯ ВЕБ-СИСТЕМИ 52
ДОДАТОК Б ГРАФІЧНЕ ЗОБРАЖЕННЯ БАЗИ ДАНИХ СИСТЕМИ 52
ДОДАТОК В ОСНОВНІ КОДИ ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ 54

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОСИЛАНЬ

Терміни

Аккаунт – це обліковий запис відвідувача тієї чи іншої веб-сторінки, що дозволяє гостю перейти в статус зареєстрованого користувача.

Консалтинг – це діяльність з консультування керівників, управлінців з широкого кола питань у сфері фінансової, комерційної, юридичної, технологічної, технічної, експертної діяльності.

Фейк – це слово іншомовного походження (від англійського «Fake»), що означає підробка, фальсифікація.

Фреймворк (англ. Framework) – це програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту.

Фріланс – це робота на відстані, без підписання договору з працедавцем. Людей, що займаються фрілансом, називають фрілансерами. Фрілансер виконує призначений об’єм робіт і йому виплачують за це гонорар.

AJAX (Asynchronous JavaScript And XML) – це підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані.

CLR – Common Language Runtime – віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

CRUD – Create, Read, Update, Delete – 4 базові функції управління даними «створення, зчитування, зміна і видалення».

HTTP – протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від Hyper Text Transfer Protocol, протокол передачі гіпертекстових документів.

IDE – Integrated Development Environment – комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду.

IIS – Internet Information Services – web-сервер, здатний виконувати Web-застосування, створені із використанням технології ASP.NET.

Java – мультиплатформенна мова програмування та платформа для виконання додатків, написаних на Java.

JVM – Java Virtual Machine – віртуальна машина Java – основна частина виконуючої системи Java, так званої Java Runtime Environment (JRE). Віртуальна машина Java виконує байт-код Java, попередньо створений з вихідного тексту Java-програми компілятором Java.

MVC – Model-View-Controller (Модель–вигляд–контролер) – це архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Open Source – програмне забезпечення з відкритим сирцевим кодом.

PHP – Hypertext Preprocessor – це мова програмування, спеціально розроблена для написання web-додатків (сценаріїв), що виконуються на веб-сервері.

PostgreSQL – вільна об’єктно-реляційна СКБД. Основана, як і всі реляційні СКБД на мові SQL.

Python – об’єктно-орієнтована, мультиплатформенна та скриптова мова програмування, зазвичай служить для написання наукових додатків, але може й використовуватись для написання повноцінних десктопних та веб-додатків.

SQL – structured query language – структурована мова запитів, являється основною для роботи з реляційними СКБД. Існують модифікації цієї мови, наприклад T-SQL (transact SQL), від компанії Microsoft, використовується в СКБД Microsoft SQL.

Скорочення

англ. – англійською.

БД – база даних.

МП – мова програмування.

ОС – операційна система.

ПЗ – програмне забезпечення.

СКБД – система керування базами даних.

JIT – just in time.

ВСТУП

Разом зі значним розвитком інтернет-технологій значно зросла потреба у вмілих та талановитих дизайнерах, оскільки мережа інтернет вже не тільки засіб комунікації та обміну даними, але й широко доступний торговий та рекламний майданчик.

Потреба в дизайнерських вміннях може виникнути в різних галузях суспільства: будь то торгова галузь чи волонтерська діяльність. Знайти працівників-дизайнерів, які могли б виконати необхідні завдання можна на:

- звернутись до знайомих;
- біржі фрілансу;
- на різних веб-сервісах, які надають необхідні послуги.

Власне для самих дизайнерів для пошуку місця роботи можна використовувати ті ж самі джерела, а також різні сервіси пошуку роботи. Проте, для ефективного пошуку необхідні:

- певний стаж (вже виконані до цього дизайнерські роботи);
- постійне слідкування за станом та оновлення власного резюме на всіх ресурсах, куди воно було подане;
- оцінка інших людей (роботодавців чи користувачів) виконаних дизайнерських робіт.

Проблема пошуку творчої роботи актуальна не тільки для самих дизайнерів, а й для роботодавців, що займаються наданням подібних послуг, які на пошуки сумлінного, талановитого співробітника витрачають дуже багато сил і часу.

Метою дипломного проекту є розробка веб-орієнтованої системи, яка допоможе людям з творчими професіями розміщувати і переглядати публікації (роботи) різних областей дизайну, моди, ілюстрації, промислового дизайну, архітектури, фотографії, образотворчого мистецтва, реклами, книжкового оформлення, анімації, звукового дизайну та інших креативних професій. А також роботодавцям допоможе знаходити співробітників, так як відразу можна подивитися їх роботи.

Основна ідея полягає в тому, щоб допомогти користувачеві легко і швидко створити свою персональну сторінку портфоліо, куди б вони (користувачі) завантажувати свої роботи, а інші користувачі, в тому числі й потенційні роботодавці, могли їх оцінювати й можливо пропонувати співпрацю.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предмету, цілей та особливостей веб-системи для публікації свого портфоліо

Як було зазначено у вступі, проблема пошуку роботи дизайнерами та роботодавцями, що знаходяться в пошуку дизайнерів наразі являється досить актуальною, оскільки для підтримання цього процесу витрачається велика кількість часу.

Завдяки веб-системі для публікації власного портфоліо можна вирішити, або значно спростити вирішення певних проблем пов’язаних з пошуком роботи або працівника (-ів). Це і являється основною ціллю даної веб-орієнтованої системи, проте, вона може бути використана і в інших цілях, наприклад, для зберігання та презентації власних робіт певного підприємства, або приватного підприємця (дизайнера-фрілансера) тощо[1].

Веб-орієнтована система для публікації власного портфоліо – це веб-портал, основною цільовою аудиторією якого являються дизайнери, другі представники креативних професій та роботодавці, які знаходяться в пошуку працівників з творчими вміннями.

Враховуючи вищесказане, завданнями даної дипломної роботи є:

- надання безкоштовного майданчика для розміщення власних робіт творчим людям;
- надання можливості перегляду та оцінки чужих робіт користувачами веб-системи;
- надання користувачам можливості контактувати один з одним;
- надання користувачам можливості робити підписку на вподобаного автора робіт;
- надання користувачам можливості переглядати вподобані ними до цього роботи;
- надання користувачам можливості перегляду статистики їх власного аккаунту (кількість переглядів аккаунту користувача за певний період, кількість підписаних на нього користувачів тощо).

Для підтримки ідеї веб-системи розміщення власного портфоліо «Nodesi» система повинна володіти певними особливостями, такими як строгий функціонал розміщення власних робіт (мається на увазі, що потрібен механізм, який не дозволяв би завантажувати роботи, що не відносяться до творчої тематики), система контролю за плагіатом робіт тощо.

1.2 Аналіз існуючих аналогів

Для визначення функцій, вимог, бізнес-логіки майбутньої веб-системи для публікації власних робіт проведено необхідний аналіз існуючих і функціонуючих в мережі Інтернет аналогічних веб-сервісів. На їх основі виділено основні переваги і недоліки майбутньої веб-системи пошуку плагіату.

1.2.1 Сімейство блогів «Tumblr»

Сімейство блогів «Tumblr» – це веб-сервіс, який дозволяє створювати власні блоги і публікувати в них свої роботи. Ділитись історіями, фотогрфіями, GIF-рисунками, телесеріалами, посиланнями, жартами, розповідями, власними mp3-записами, відео, філософськими міркуваннями, новинками моди та мистецтва[2].

Головна сторінка наведена рис. 1.1.

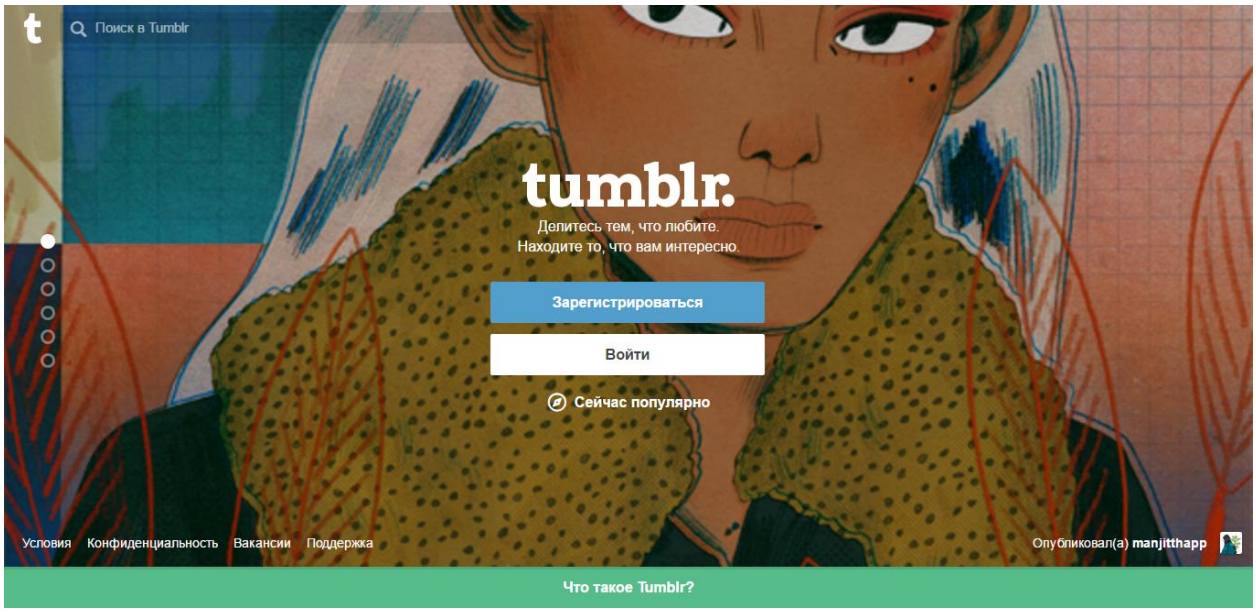

Рисунок 1.1 – Головна сторінка веб-сервісу розміщення власних вподобань «Tumblr»

З наведеного вище рисунка видно, що для того, щоб перед тим, як розпочати роботу з даним веб-сервісом він пропонує пройти процедуру

реєстрації або входу. Проте, натиснувши кнопку «Сейчас популярно» можна перейти до перегляду чужих блогів (рис. 1.2).

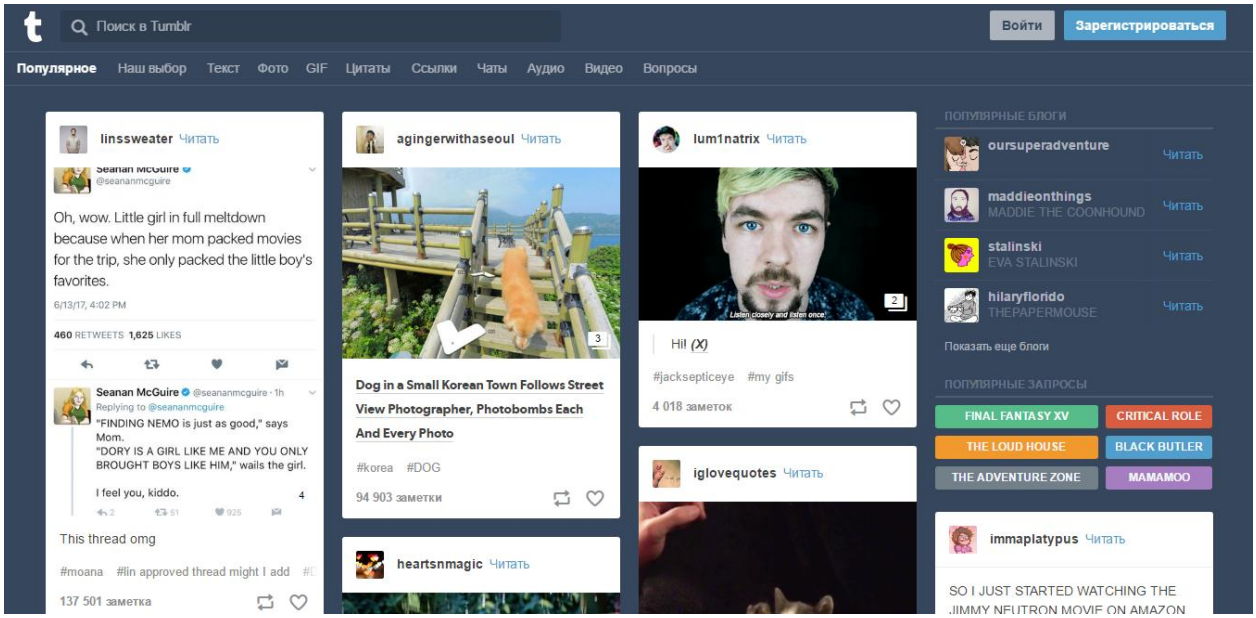

Рисунок 1.2 – Сторінка з контентом різних довільних блогів веб-сервісу «Tumblr»

Після реєстрації в веб-системі стає доступним:

- підписка на чужі блоги, що дозволяє переглядати інші роботи;
- можливість ставити вподобання чужим роботам;
- робити «реблог» в свій блог з можливістю добавити власний коментар[3];
- пересилання чужих робіт в Twitter або Facebook з посиланням на них;
- можливість ведення свого блогу.

Після підписки на чужі блоги вони з’являються в стрічці новин, роблячи її більш «розумнішою» ніж в не зареєстрованих користувачів.

Слід також сказати про функцію «Черга повідомлень» (англ. Queued Posts), яка дозволяє планувати публікацію записів на певний час. Tumblr рекламує цю функцію як «простий спосіб зберігати ваш блог активним і насиченим» (англ. «An easy way to keep your blog active and consistent»).

До недоліків даної веб-системи блогів можна віднести те, що:

- можна розміщувати не тільки власний контент, а й сторонні роботи з мережі інтернет;
- відсутність системи тет-а-тет комунікації в середині системи;

- основна аудиторія англомовна й знаходиться поза межами України й країн-сусідів;
- відсутність інтеграції з соціальними мережами (наприклад Facebook, Google+ тощо), а тому реєстрація займає більше часу.

1.2.2 Онлайн-портфоліо «Behance»

Онлайн-портфоліо «Behance» – це мережа сайтів та сервісів, котрі спеціалізуються на саморекламі, включаючи консалтинг та веб-сайти портфоліо. З 2012 року належить компанії Adobe[3]. Adobe Behance являється провідною інтернет-платформою для перегляду і публікації результатів творчої роботи. Вона забезпечує зручне, централізоване оновлення і публікацію різних творчих робіт для людей творчих професій[4].

Нинішній володар сервісу, компанія Adobe, провела тісну інтеграцію даної платформи з власним сервісом Creative Cloud.

Головна сторінка сервісу онлайн-портфоліо «Behance» зображена на рис. 1.3.

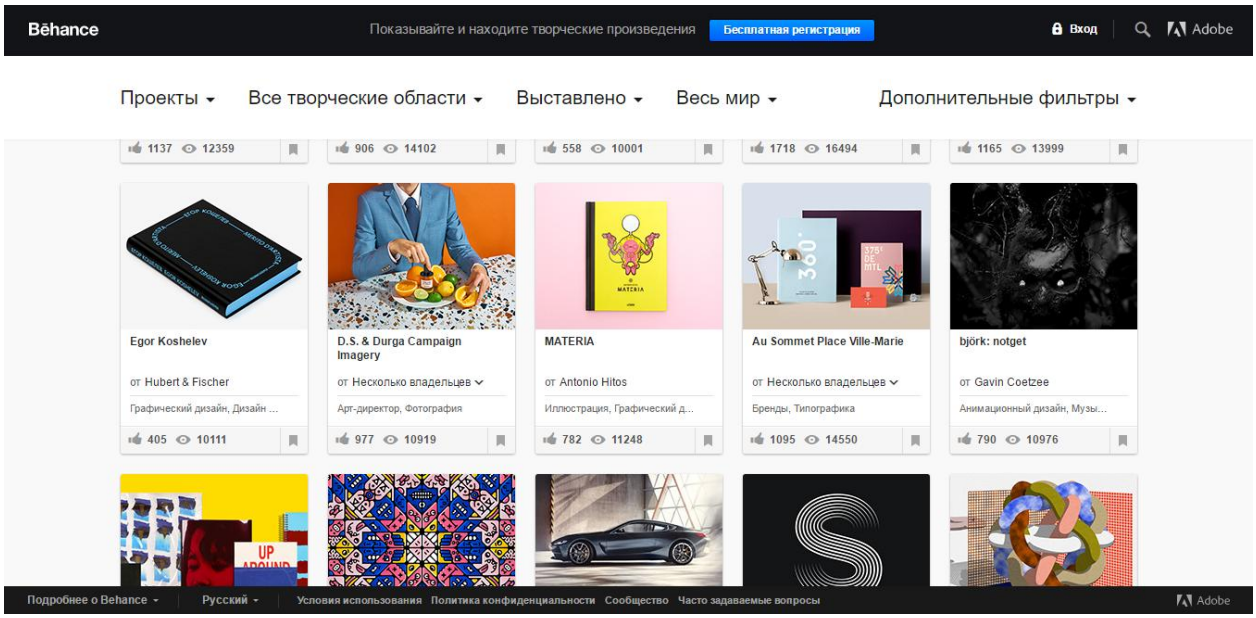

Рисунок 1.3 – Головна сторінка веб-сервісу онлайн-портфоліо «Behance»

На відміну від сімейства блогів «Tumblr», опис якого наведено в пункті 1.2.1, даний веб-сервіс дозволяє відразу переглядати чужі роботи, проте для того, щоб поставити вподобання (англ. Like) або залишити коментар також необхідно зареєструватись (що цілком логічно).

Оскільки власником веб-сервісу «Behance» являється компанія Adobe, то, для просування власних продуктів, реєстрація та авторизація тісно пов’язана з системою Adobe ID, що не є досить зручним. По цій же причині в системі відсутня реєстрація і авторизація за допомогою соціальних мереж, що безумовно можна віднести до мінусів даної веб-системи.

«Behance» дозволяє переглядати роботи інших користувачів, ставити їм вподобання, коментувати їх та проводити листування з іншими користувачами. Безумовно, це являється плюсом даної веб-системи, враховуючи, що, на відміну від попереднього розглянутого сервісу «Tumblr», дана система працює тільки з власними роботами користувачів (даний сервіс не підтримує плагіат). Слід також вказати, що роботи інших користувачів не можна оцінювати в негативному ключі. Можна або підтримати автора, або залишити роботу без уваги.

Дана веб-система надає можливості для компаній по пошуку працівників, а також надає користувачам можливість просування себе, як кандидата на роботу.

Слід також зазначити, що продукти компанії Adobe є досить дорогими. Даний факт несе в собі певні «сюрпризи», пов’язані з використанням інтернет-платформи онлайн-портфоліо «Behance» – з однієї сторони – платформа безкоштовна, а з іншої – користуватись нею без продуктів компанії Adobe не можливо. Це означає, що повноцінне використання даного веб-сервісу не являється безкоштовним, що безумовно також являється мінусом.

Основна аудиторія веб-сервісу «Behance» хоч і являється одною з найбільших в даному сегменті, але так само як і в веб-сервісі «Tumblr» є англомовна і знаходиться поза межами України та її близьких сусідів. Також, від такого великого напливу користувачів та відвідувачів даного інтернет-порталу веб-сайт сервісу іноді працює зі значними затримками.

1.2.3 Спільнота дизайнерів «Dribbble»

Dribble – це спільнота дизайнерів та людей творчих професій. Веб-дизайнери, графічні дизайнери, ілюстратори, художники, друкарі, та інші люди творчих професій можуть ділитися невеликими скріншотами (знімками), які показують їх роботу, процес, і поточні проекти[5].

Даний веб-сервіс дуже схожий на соціальну мережу «Twitter», оскільки, аналогічно як це реалізовано в «Twitter» з текстом, в ньому можна

ділитись графічними роботами розміром 120 квадратних пікселів, проте, якщо завантажувана графічна робота більше за вказаний розмір, то система сама її обріже[6].

Головна сторінка веб-сервісу спільноти дизайнерів «Dribbble» зображена на рис. 1.4.

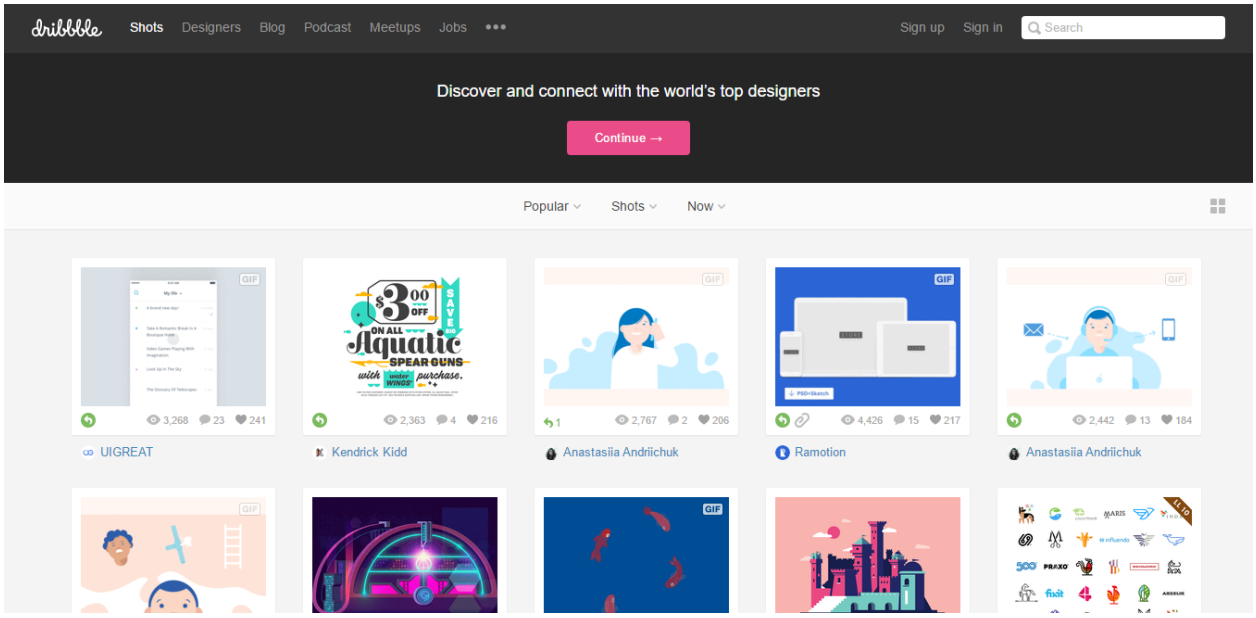

Рисунок 1.4 – Головна сторінка веб-сервісу «Dribbble»

Дана веб-система веде себе як звичайна соціальна мережа – вона дозволяє переглядати зменшені прототипи чужих творчих, дизайнерських робіт, підписуватись на оновлення інших конкретних користувачів, роботи перепости чужих робіт тощо.

Проте, зареєструватись в «Dribbble» можна лише в якості новачка (prospect), тобто без можливості публікації власних робіт. Повноцінний доступ до веб-сервісу можна отримати тільки по запрошенню (draft) іншого користувача. Отримати таке запрошення можна або від друзів-дизайнерів, які мають там повноцінний аккаунт, або після представлення посилання на власні роботи.

З одної сторони, це жорсткий контроль за користувачами, який дозволяє контролювати плагіат чужих робот на даному веб-сервісі. Але з іншої сторони це значно зменшує шанси новачків показати себе.

Враховуючи вищесказане, можна зробити висновок, що веб-сервіс «Dribbble» являється елітним майданчиком для творчих людей, попасти на який досить складно, що певним чином відсіює поганих та середніх (middle)

дизайнерів, але надає змогу компаніям знайти собі гідних та сумлінних робітників, в яких вони можуть бути впевненими.

Як і в попередніх випадках, основна аудиторія веб-сервісу – англомовна й знаходиться поза межами України та її близьких сусідів.

Чужі роботи можна переглянути натиснувши на маленький блок з роботою в стрічці робіт (рис. 1.5). В відкритому вікні можна прочитати опис певної роботи, кількість переглядів роботи, кількість вподобань тощо.

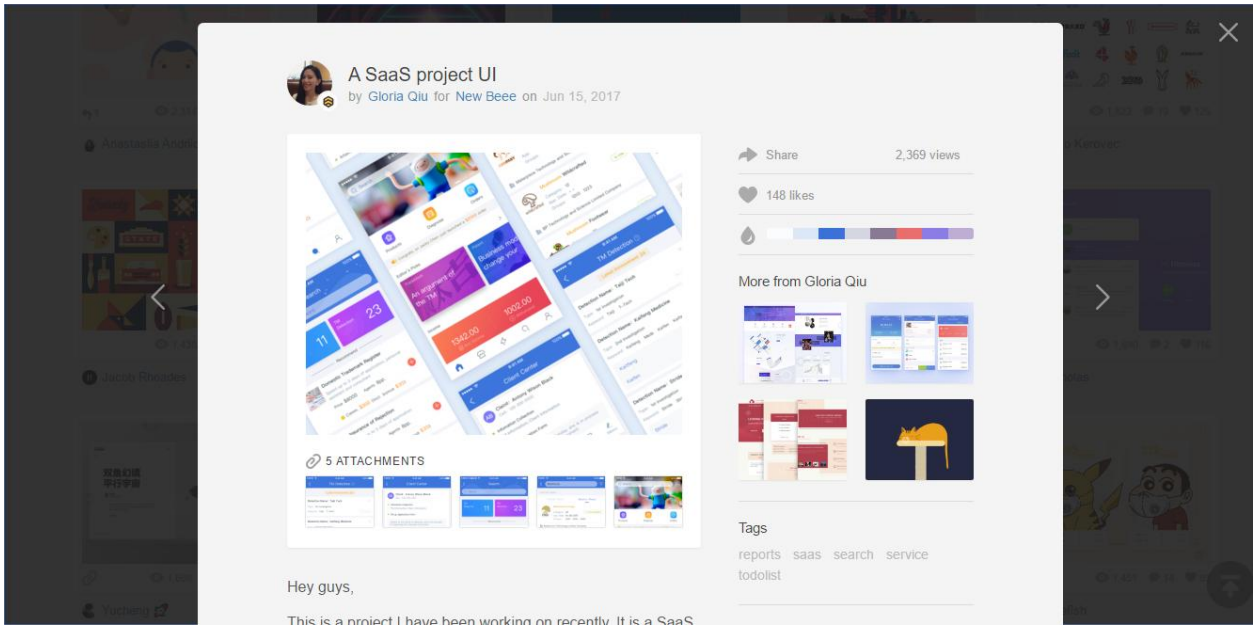

Рисунок 1.5 – Відкрита сторінка з обраною роботу на веб-сервісі «Dribbble»

Веб-сервіс має власний загальний рейтинг користувачів – All Stars, за перші місця в якому йде жорстока боротьба.

Ще однією важливою послугою для роботодавців, яку надає веб-сервіс «Dribbble» являється пошук талановитих людей, виходячи з місця проживання, навиків, доступні в конкретний момент та інших критеріїв. Дана послуга досить дорога – 200 доларів за місяць. Цією функцію даного веб-сервісу користуються такі гіганти як Microsoft, Facebook, Tumblr, Groupon та інші відомі компанії[6].

Як і в попередніх розглянутих веб-сервісах в «Dribbble» відсутня інтеграція з соціальними мережами.

В системі відсутня система тет-а-тет комунікація між іншими користувачами, лише з роботодавцями.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

У результаті проведеного аналізу існуючих систем з обраної предметної області необхідно провести вибір архітектури та апаратно-програмних засобів реалізації веб-системи онлайн-портфоліо «Nodesi».

Перед тим як приступати до реалізації інформаційної веб-системи необхідно здійснити проектування архітектури системи, бази даних предметної області, обрати веб-сервер та мову програмування. При добре продуманій архітектурі бази даних і грамотному виборі інструментів для створення даної системи стає набагато легше вирішувати проблеми: розширення можливостей інформаційної веб-системи; забезпечення продуктивності, адекватного навантаження; поділу повноважень користувачів інформаційної системи; додавання нових функцій інформаційної системи тощо.

2.1 Вибір архітектури для розробки веб-системи

Веб-застосування являють собою особливий тип програмних засобів, побудованих за архітектурою «клієнт-сервер», зображеній на рис. 2.1.

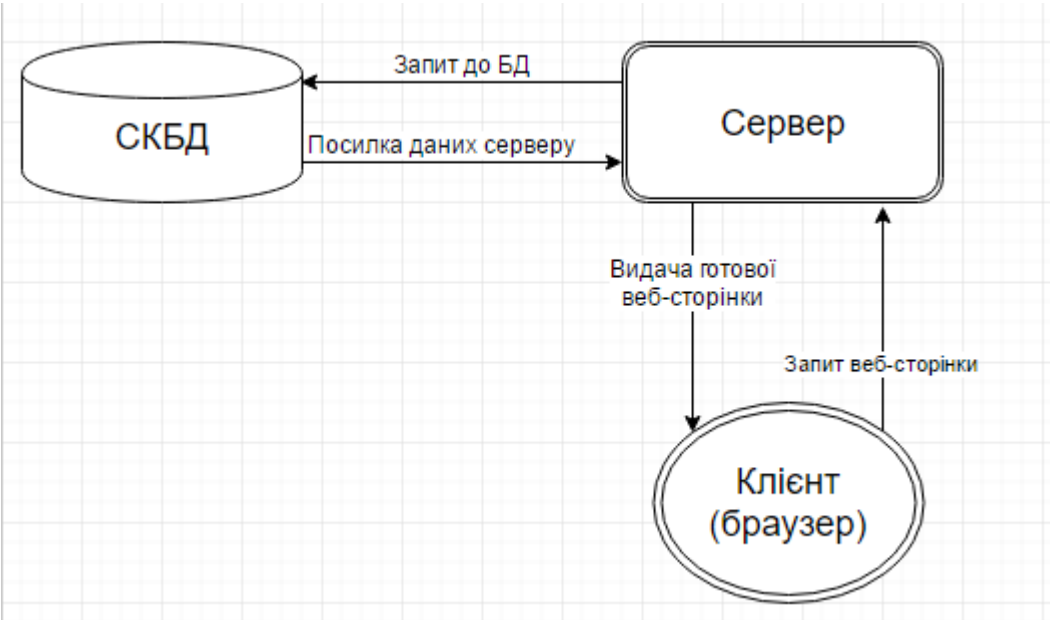

Рисунок 2.1 – Схема архітектури типу «клієнт-сервер»

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні

розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів[3].

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає). Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі[3].

2.2 Вибір мови для програмування веб-систем

На даний момент більшість програмних платформ та мов програмування надають можливість розробляти та підтримувати за їх допомогою веб-системи будь-якого рівня складності.

Вибір мови програмування досить важливий етап, оскільки в залежності від обраної МП можуть змінюватись такі атрибути веб-системи як:

- швидкість розробки веб-системи;
- складність розробки веб-системи (оскільки для підтримки і розробки під різні веб-сервери використовуються різні програмні модулі мов програмування, які можуть бути досить складними для розуміння);
- швидкість роботи веб-системи;
- необхідне обладнання веб-серверу для підтримки роботи веб-системи.

Також грає свою роль досвід, вміння та навички програміста працювати з тою чи іншою мовою програмування.

В веб-програмуванні існує дві групи мов програмування: клієнтські та серверні.

Клієнтські МП виконуються на стороні клієнта, а тому результат виконання скриптів цієї групи мов програмування може залежати від браузера клієнта і тому, виконання може відрізнятись, тому, ця група МП розглядатись не буде.

Виконання серверних МП значно відрізняється від клієнтських: коли користувач дає запит на яку-небудь сторінку (переходить на неї по посиланню або вводить адресу в адресному рядку свого браузера), то викликана сторінка спочатку обробляється на сервері, тобто виконуються всі програми, пов'язані зі сторінкою, і тільки потім повертається до відвідувача по мережі у вигляді файлу.

Робота програмних модулів вже повністю залежить від серверу, на якому розташована веб-система, і від того, яка версія тієї чи іншої мови програмування підтримується. До серверних МП можна віднести: PHP, Perl,

Python, Ruby, будь-яка .NET мова програмування (технологія ASP.NET), Java.

Важливою стороною роботи серверних мов є можливість організації безпосередньої взаємодії з системою керування базами даних.

2.2.1 Мова програмування Java

Java – сильно типізована об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (в подальшому придбана компанією Oracle). Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі, за допомогою віртуальної Java-машини (JVM)[3]. Приклад коду Java EE та IDE NetBeans зображені на рис. 2.2.

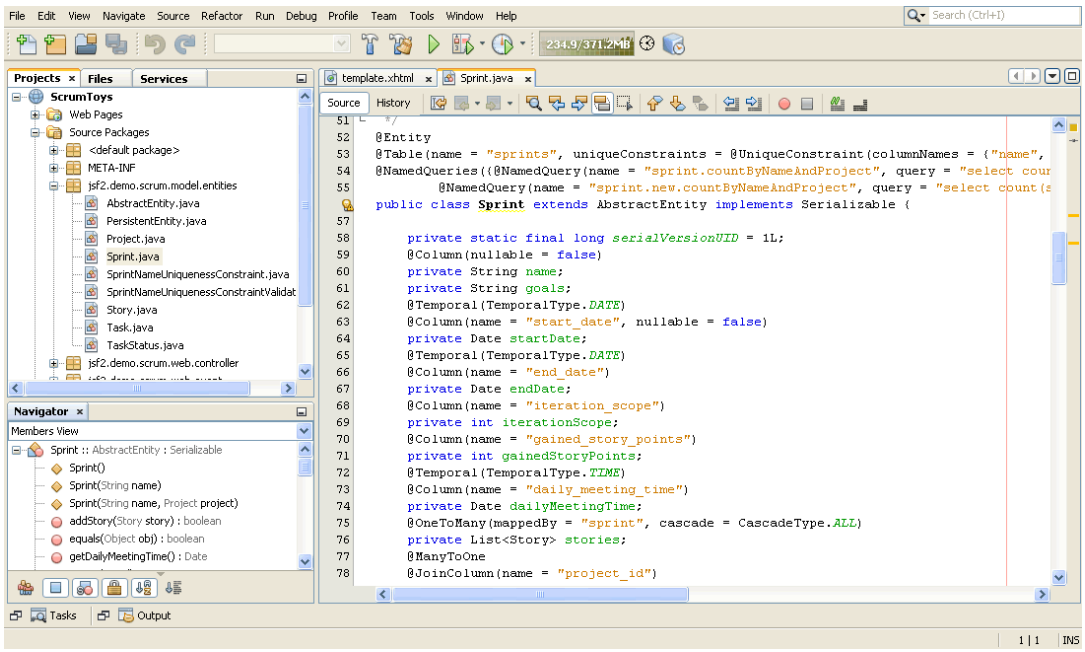

Рисунок 2.2 – Приклад коду Java EE та IDE NetBeans

Для розробки веб-систем в МП Java є окремий випуск – Java EE. Java EE (Enterprise Edition) являє собою широко використовувану платформу, яка містить набір взаємозв'язаних технологій, які істотно скорочують вартість і складність розробки, розгортання багаторівневих серверних додатків, а також управління ними. Платформа Java EE основана на платформі Java SE і надає набір інтерфейсів API (інтерфейсів розробки додатків) для розробки і

запуску портованих, надійних, масштабованих і безпечних серверних додатків[7]. Схема роботи додатків, розроблених за допомогою Java EE зображена на рис. 2.3. Java EE багатоярусні додатки, як правило, вважається три-рівневими додатки, тому що вони розподілені на три локації: клієнтські машини, машини сервера Java EE та машини баз даних. Триступінчасті додатки, що працюють таким чином, розширюють стандартний дворівневий клієнт і серверну модель шляхом розміщення мультипоточного серверу додатків між клієнтським додатком і фоновим зберіганням.

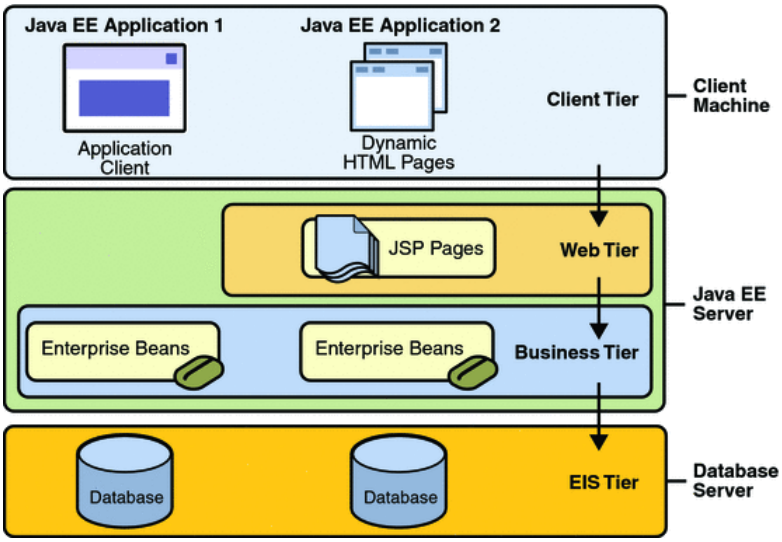

Рисунок 2.3 – Схема роботи додатків, розроблених за допомогою технології Java EE

Веб-додаток Java створює інтерактивні веб-сторінки, що містять різні типи мов розмітки (HTML, XML і т.д.), а також динамічний вміст. Цей вміст зазвичай формується веб-компонентами, наприклад сторінками JavaServer (JSP), сервлетами і компонентами JavaBean, які дозволяють змінювати дані і здійснювати їх тимчасове зберігання, взаємодіяти з базами даних і веб-службами, а також відображати вміст у відповідь на запити клієнтів[7]. Комунікацію на стороні веб-серверу в середині інтернет-додатку, написаного за допомогою технології Java EE зображено на рис. 2.4.

Зображений нижче рис. 2.4 також показує різні елементи, які можуть становити клієнтський рівень. Клієнт може напряму взаємодіяти з бізнес-рівнем, що працює на сервері Java EE, або, як у випадку клієнта, що працює в браузері, перейшовши через сторінки JSP або сервлети, що працюють через веб-рівень[7].

Додаток Java EE використовує клієнт на основі браузера або клієнтські додатки. При вирішенні питання, який з них використовувати, потрібно бути проінформованим про компроміси між збереженням функціональності клієнта та користувача. Чим більше функціональності буде представлено на сервері, тим легше поширювати, розгортати і керувати додатком. Однак, маючи більше функціональних можливостей на стороні клієнта можна веб-систему зробити кращою для сприйняття користувачами[7].

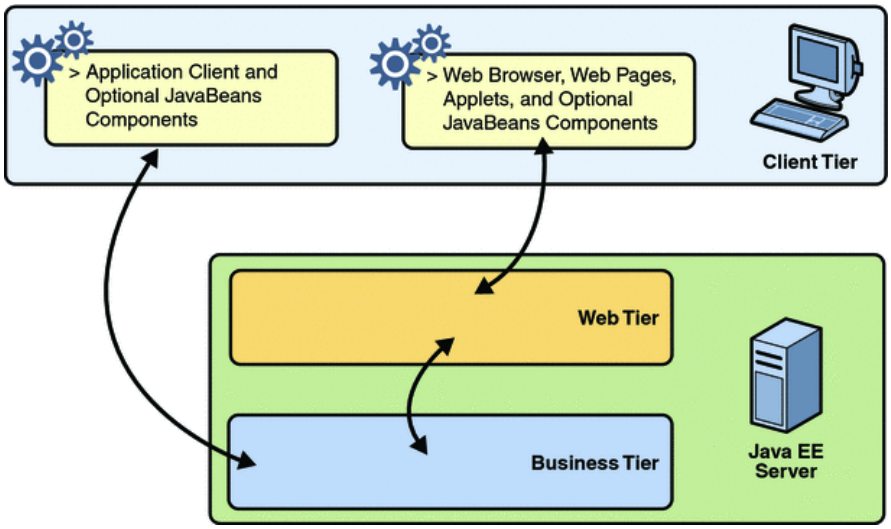

Рисунок 2.4 – Комунікація на інтернет-додатку Java EE на стороні веб-серверу

Так як багато програмних модулів веб-додатків можуть повторюватися або вимагати наявності надлишкового шаблонного коду, то для зменшення кількості спільних дій слід застосовувати веб-платформи. Багато платформи (наприклад, JavaServer Faces) надають бібліотеки для створення шаблонів сторінок і управління сеансами, а також часто забезпечують повторне використання коду.

Підсумовуючи написане вище, можна сказати, що для програмування веб-додатків Java EE надає достойну архітектуру, достатній інструментарій та достатню швидкість роботи розроблених веб-систем.

Проте, додатки написані за допомогою Java EE досить громіздкі, а для розгортання веб-системи на машині сервера повинні бути досить великі системні потреби для її швидкої роботи. Це нормально для дійсно великих веб-проектів, оскільки до них застосовуються більш високі вимоги до надійності, але для невеликих веб-систем це надлишково. Набагато швидше

розробляти прості веб-додатки за допомогою інших технологій, таких як наприклад Python чи PHP.

Також, дуже велику роль при написанні веб-додатків на Java EE грає досвід та знання програміста, тому що без знань правильної архітектури та технології розробки веб-додатків розроблений веб-додаток буде значно поступатись по характеристикам з іншими.

В більшій частині аналогічними проблемами і перевагами наділені мови програмування, що використовують в своїй екосистемі .NET Framework (технологія ASP .NET), тому їх розглядання являється надлишковим. Можна лише додати, що для функціонування технології ASP .NET веб-сервером повинен бути тільки Microsoft IIS, який не являється безкоштовним та коштує чималу суму грошей.

2.2.2 Мова програмування Python

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій[3]. З технічної точки зору web-додаток на Python – повноцінний додаток, завантажений в пам'ять, що володіє своїм внутрішнім станом, збережуваним від запиту до запиту[8].

Приклад коду веб-додатка на Python та IDE Microsoft Visual Studio зображені на рис. 2.5.

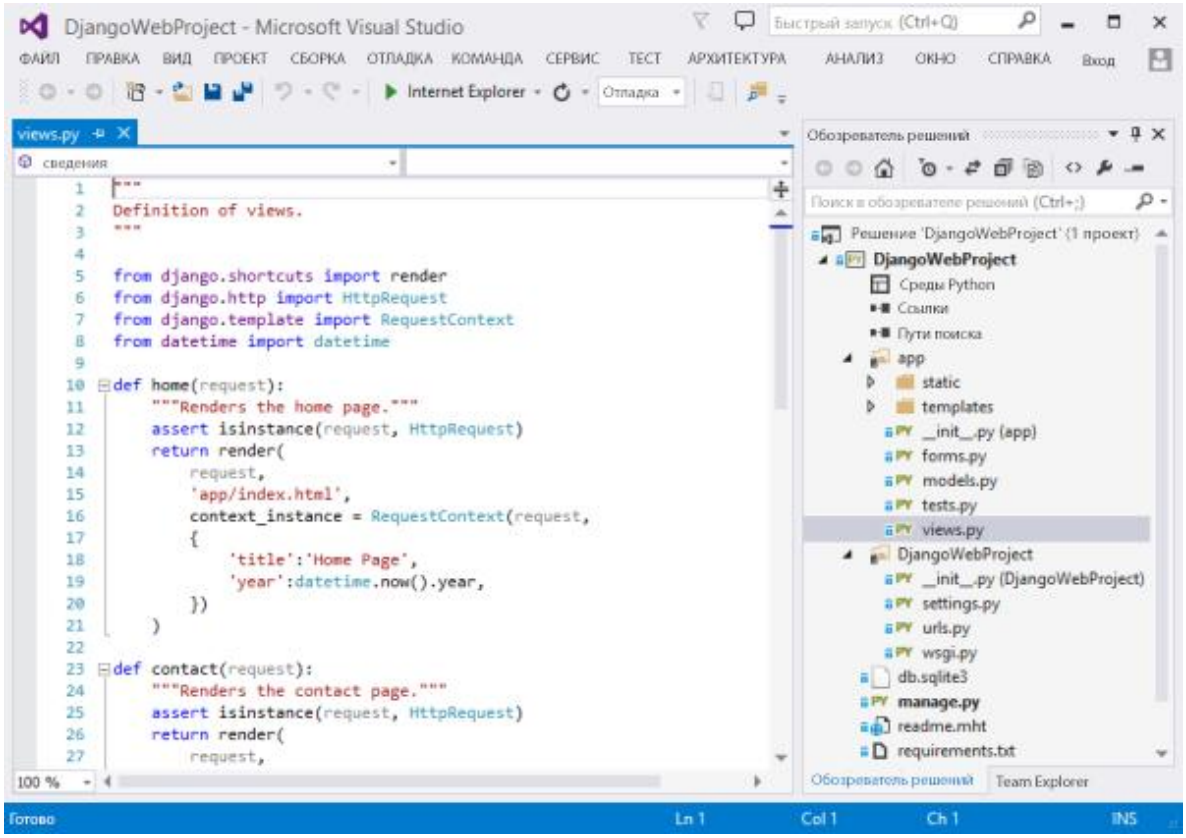

Рисунок 2.5 – Приклад коду веб-додатка на Python та IDE Microsoft Visual Studio

Кожен HTTP-запит в МП Python обробляється в окремому потоці (це ж справедливо і для процесів). При цьому кожен потік за час свого життя обробляє безліч HTTP-запитів, зберігаючи свій стан (тобто значення змінних рівня модулів) від запиту до запиту. У цьому полягає чи не основна відмінність від моделі обробки HTTP-запитів в PHP, де кожен запит приходить в нове, щойно проініціалізоване оточення і завантаження програми необхідно виконувати кожен раз заново[8].

Python для розробки інтернет-додатків пропонує декілька фреймворків:

- Django;
- Bottle;
- Flask;
- Pyramid.

Наведені фреймворки різняться функціоналом та вбудованими модулями, але в загальному випадку вони дуже схожі.

Хоча МП Python і дозволяє зручно розробляти інтернет-додатки вона також має й ряд недоліків, які роблять її не такою привабливою для вибору в якості основної мови розробки даної веб-системи:

- низька швидкодія (через те, що Python не застосовує в своїй роботі JIT-компілятор);
- неможливість модифікації вбудованих класів та функцій;
- мала кількість сторонніх бібліотек функцій.

Недостатня кількість документації, важке розуміння розробленого коду та необхідність користуватись вище визначеними фреймворками для створення веб-додатків робить МП Python не привабливою для використання при розробці даної веб-системи онлайн-портфоліо «Nodesi».

2.2.3 Мова програмування PHP

PHP – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP – проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта[3].

Приклад коду мови програмування PHP та IDE PhpStorm від компанії JetBrains зображені на рис. 2.6.

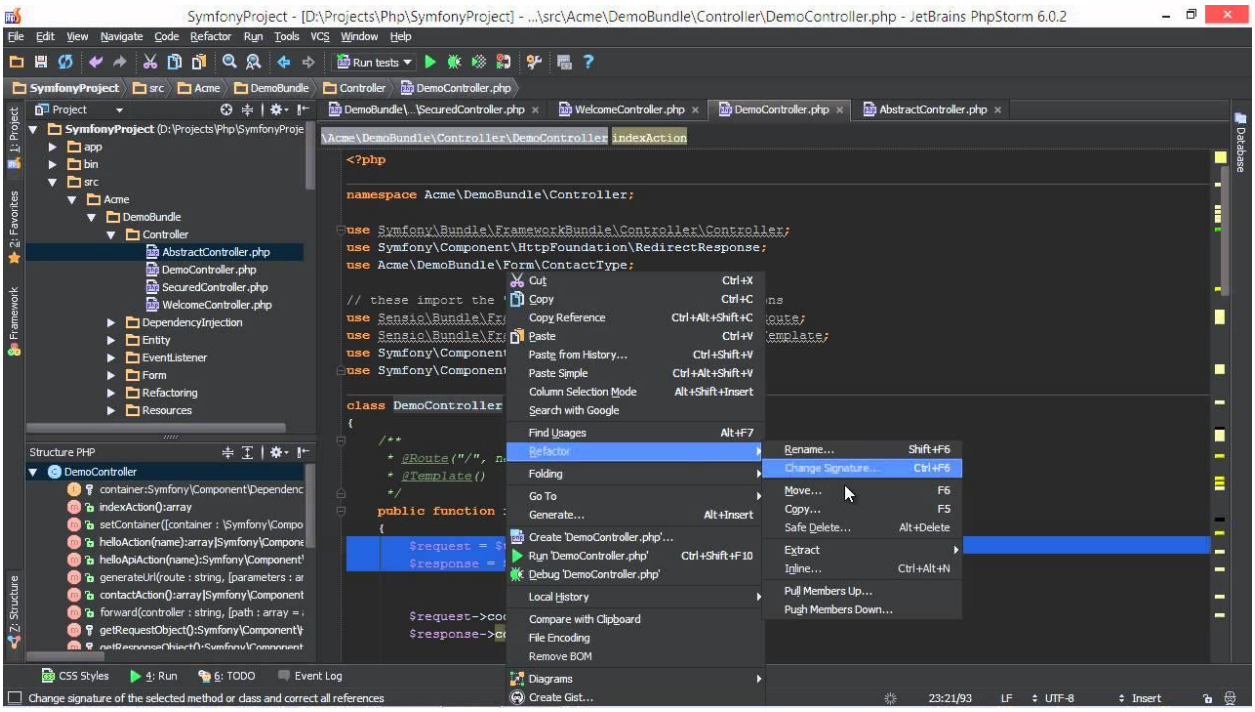

Рисунок 2.6 – Приклад PHP-коду та IDE JetBrains

Розглядаючи різні аспекти використання PHP, можна виділити такі основні її переваги:

- традиційність
- простота
- ефективність
- безпека
- гнучкість

Традиційність. Синтаксис і конструкції PHP включають багато елементів, які є у мовах програмування C, Perl, Pascal. PHP є мовою з універсальним синтаксисом і водночас пристосована до веб-програмування.

Простота. PHP може бути вбудована безпосередньо в html-код сторінок, які коректно обробляються PHP-інтерпретатором. PHP містить величезну кількість різних функцій, що позбавляє нас необхідності писати багаторядкові скрипти для виконання простого завдання. Головне для розробника – правильно вибрати функцію відповідно до конкретного завдання. Крім того, не потрібно завантажувати бібліотеки, вказувати спеціальні параметри компіляції.

PHP містить ряд готових бібліотек для роботи із популярними базами даних.

PHP надає засоби для покращення безпеки:

1) Засоби безпеки системного рівня. PHP можна налаштувати так, щоб вона забезпечувала максимальну свободу дій і безпеку. PHP може працювати в безпечному режимі (safe mode), який обмежує можливості застосування PHP користувачами. Наприклад: максимальний час виконання та використання пам’яті.
2) Засоби безпеки рівня програми. PHP включає надійні механізми шифрування. PHP також сумісний з багатьма додатками інших розробників, що дозволяє легко інтегрувати його з захищеними технологіями електронної комерції. Вихідний код PHP не можна переглянути у браузері, оскільки він виконується на сервері.

Гнучкість. PHP використовується не лише у поєднанні з HTML, але й із JavaScript, WML, XML та іншими мовами програмування. PHP-код може передаватися будь-яким браузерам і пристроям, в тому числі стільниковим телефонам, портативним комп’ютерам. PHP-код можна виконувати в режимі командного рядка[9].

PHP працює на різних web-серверах (Apache, Netscape Enterprise Server, Microsoft IIS, Stronghold, Zeus) і платформах (UNIX, Solaris, FreeBSD, Windows 95/98/NT/2000/XP/2003).

Слід також сказати про недоліки МП PHP:

- має слабкі засоби для роботи з винятками;
- глобальні параметри конфігурації впливають на базовий синтаксис мови, що ускладнює настройку сервера і розгортання додатків;
- об'єкти передаються за значенням, що бентежить багатьох програмістів, які звикли до передачі об'єктів по посиланню, як це робиться в більшості інших мов.

Утиліта PHPMyAdmin дозволяє працювати з сервером баз даних MySQL. Основні можливості phpMyAdmin:

- створювати базу даних;
- створювати таблиці в базі даних;
- додавати, видаляти і редагувати дані в таблицях;
- здійснювати пошук даних;
- встановлювати привілеї на базу даних, таблицю;
- відновлювати базу даних та багато іншого.

Тобто, phpMyAdmin дозволяє робити початкові налаштування бази даних і її вмісту [9].

Web-утиліта PhpMyAdmin зображена на рис. 2.7.

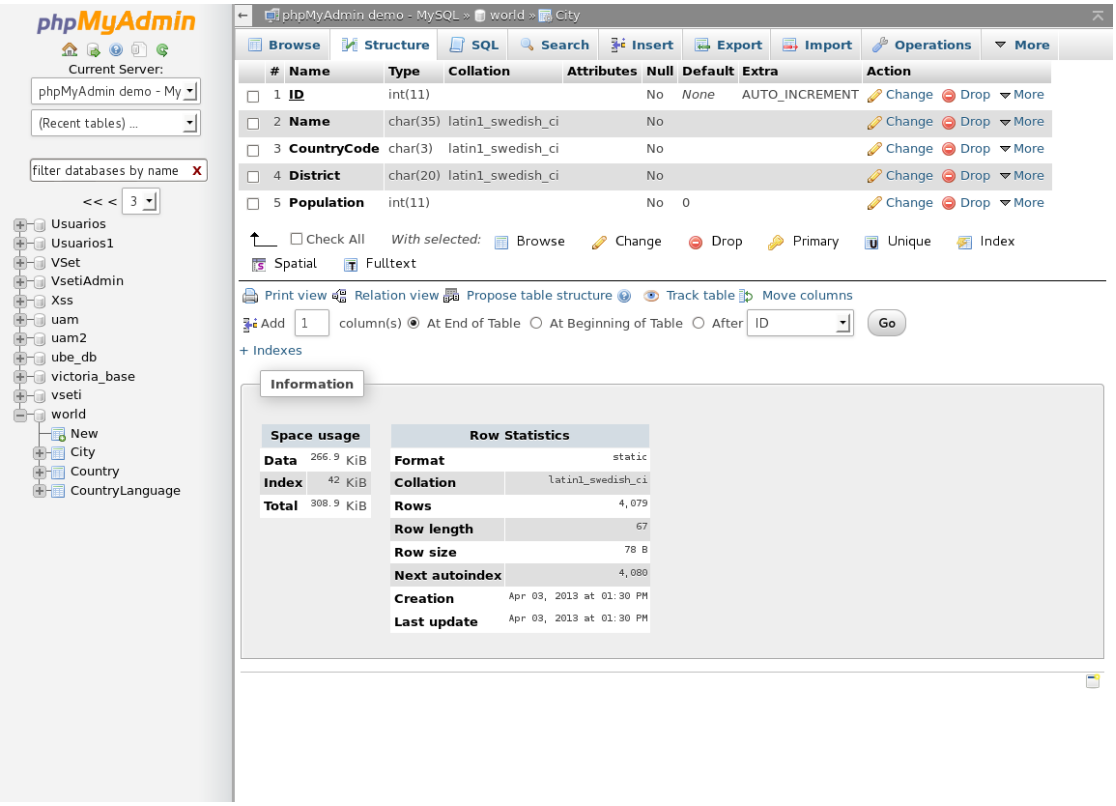

Рисунок 2.7 – Утиліта PhpMyAdmin

Підсумовуючи написане вище, мову програмування PHP можна обрати в якості основної для розробки даного веб-сервісу онлайн-портфоліо «Nodesi», оскільки підведені підсумки по недолікам показують, що вони не значні і на досягнення цілі (створення веб-орієнтованої системи онлайн-портфоліо «Nodesi» не впливають.

2.3 Вибір СКБД для управління контентом веб-системи

При написанні програм в любій сфері, будь то розробка веб-систем, чи мобільних додатків, або десктопних додатків, неможливо обійтись без використання баз даних, оскільки навіть невеликі системи потребують зберігання та відтворення даних.

Бази даних – це спеціально розроблене сховище для різних типів даних. Кожна база даних має певну модель (реляційна, документно-орієнтована), яка забезпечує зручний доступ до даних. Системи керування базами даних

(СКБД) – спеціальні додатки (або бібліотеки) для управління базами даних різних розмірів і форм[10].

На даний момент існують два основних класи СКБД:

1) реляційні;
2) нереляційні.

Перші використовують в роботі мову SQL (Structured Query Language – мова структурованих запитів) та роблять опір на відносини (relations, звідси і назва - реляційні), а інші використовують власну реалізацію CRUD (Create, Read, Update, Delete) функцій, та ще носять назву NoSQL бази даних.

Дані веб-системи онлайн-портфоліо «Nodesi» мають строгу і стабільну структуру, а тому для їх оперуванням слід зробити вибір на користь реляційних СКБД.

СКБД повинна забезпечувати реляційну модель роботи з даними. Сама модель має на увазі певний тип зв'язку між сутностями з різних таблиць. Щоб зберігати і працювати з даними, такий тип СКБД повинен мати певну структуру (таблиці). У таблицях кожен стовпець може містити дані різного типу. Кожен запис складається з безлічі атрибутів (стовпців) і має унікальний ключ, що зберігається в тій же таблиці – всі ці дані взаємопов'язані між собою, як описано в реляційної моделі[10].

Розглянемо три реляційні СКБД:

1) MySQL;
2) PostgreSQL;
3) Microsoft SQL Server.

2.3.1 СКБД PostgreSQL

PostgreSQL – вільна об’єктно-реляційна СКБД.

Існує в реалізації для більшості UNIX-подібних систем, а також для сімейства ОС Windows. Базується на мові SQL та підтримує більшість можливостей стандарту SQL:2011[3].

Оскільки PostgreSQL базується на мові SQL, то вона і являється строго типізованою СКБД, тобто, при проектуванні бази даних повинна дотримуватись строга схема таблиць з певними типами даних, які зберігаються в полях таблиці.

Серед реляційних баз даних славиться значною швидкістю, надійними механізмами транзакцій, реплікації та є легко розширюваною.

Для роботи з цією СКБД потрібно розвернути власне сервер PostgreSQL, а для роботи з клієнтського додатку можна скористатись або консолью, або одним з графічних додатків, наприклад PG Lightning Admin, зображено на рис. 2.8.

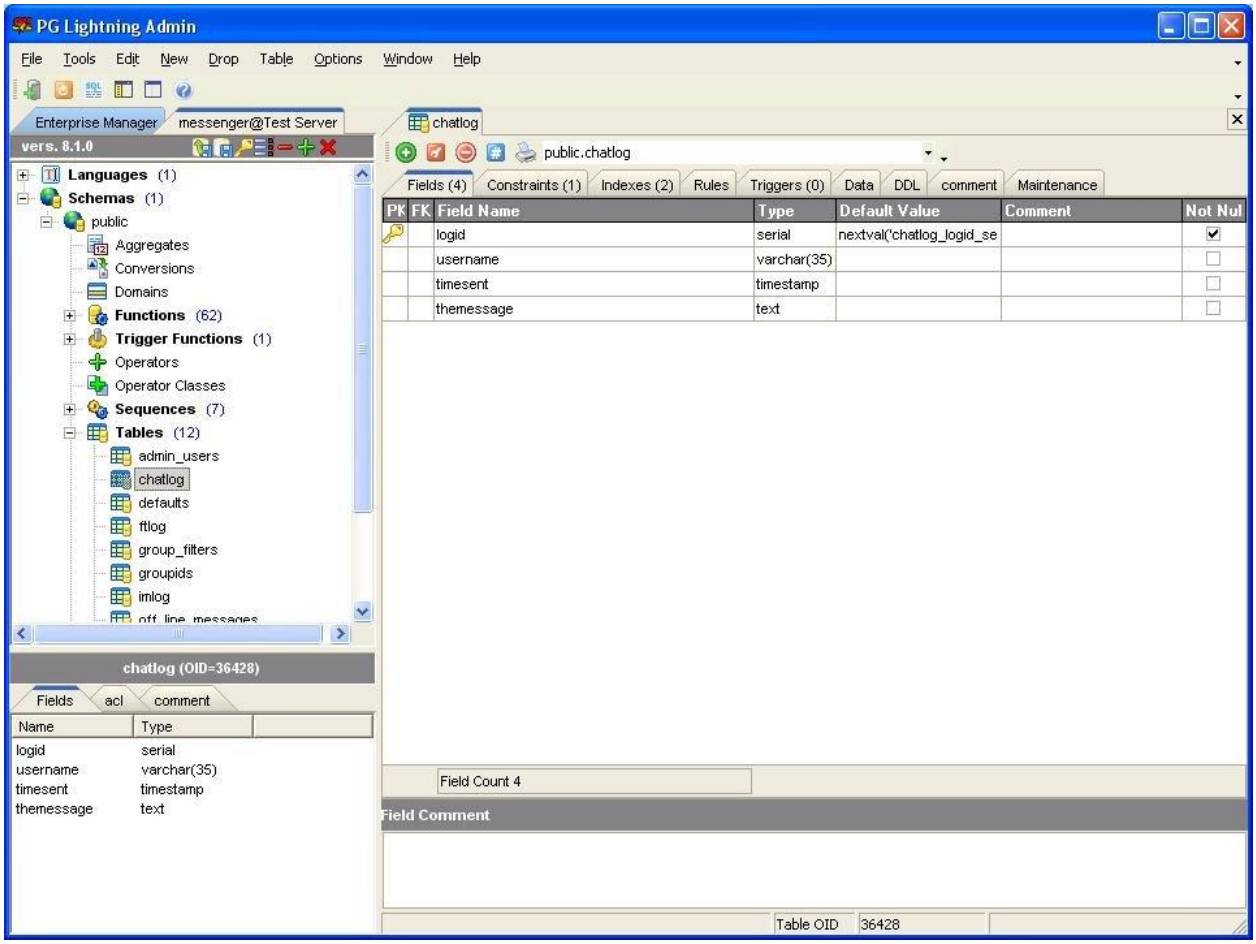

Рисунок 2.8 – Програма для керування СКБД PostgreSQL PG Lightning Admin

Переваги СКБД PostgreSQL:

- для роботи з даними використовується відома багатьом мова SQL;
- витримує великі навантаження на сервер PostgreSQL;
- здатна вмістити в себе дуже велику кількість даних;
- використання відношень (relations) полегшує роботу з структурованими даними.

Недоліки СКБД PostgreSQL:

- необхідність дотримуватись строгої структури таблиць та даних;
- мала швидкість роботи з вставкою/читанням даних в порівнянні з NoSQL рішеннями СКБД;

- використання відношень (relations) погіршує роботу з безструктурними даними.

Коли використовувати PostgreSQL[10]:

- цілісність даних – коли надійність і цілісність даних – ваші вимоги, PostgreSQL буде, мабуть, найкращим вибором;
- складні, призначені для користувача процедури – якщо необхідно використовувати призначені для користувача процедури, то PostgreSQL має вбудовану підтримку для них;
- інтеграція – якщо в майбутньому планується перехід на платні СКБД, наприклад Oracle, то зробити це з PostgreSQL буде досить просто в порівнянні з іншими безкоштовними СКБД;
- складна структура даних – в порівнянні з іншими відкритими СКБД PostgreSQL надає більше можливостей для створення складних структур даних без необхідності жертвувати якими-небудь аспектами.

Коли не слід використовувати PostgreSQL[10]:

- швидкість – якщо швидке читання для вас єдиний фактор, то варто придивитися до інших СКБД;
- просте налаштування – якщо не потрібна цілісність даних, відповідність ACID або складні структури даних, то налаштування PostgreSQL може добряче попсувати нерви;
- реплікація – якщо не потрібно витратити час і енергію на те, що можна було б з легкістю зробити, використовуючи MySQL, то напевно простіше було б на ньому і залишитися.

2.3.2 СКБД Microsoft SQL Server

Microsoft SQL Server – система керування базами даних, розроблена корпорацією Microsoft.

Основна використовувана мова запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO по структурованій мові запитів SQL з розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СКБД в цьому сегменті ринку[3].

Робота з MS SQL та додаток для роботи з нею Microsoft SQL Server Manager зображені на рис. 2.9.

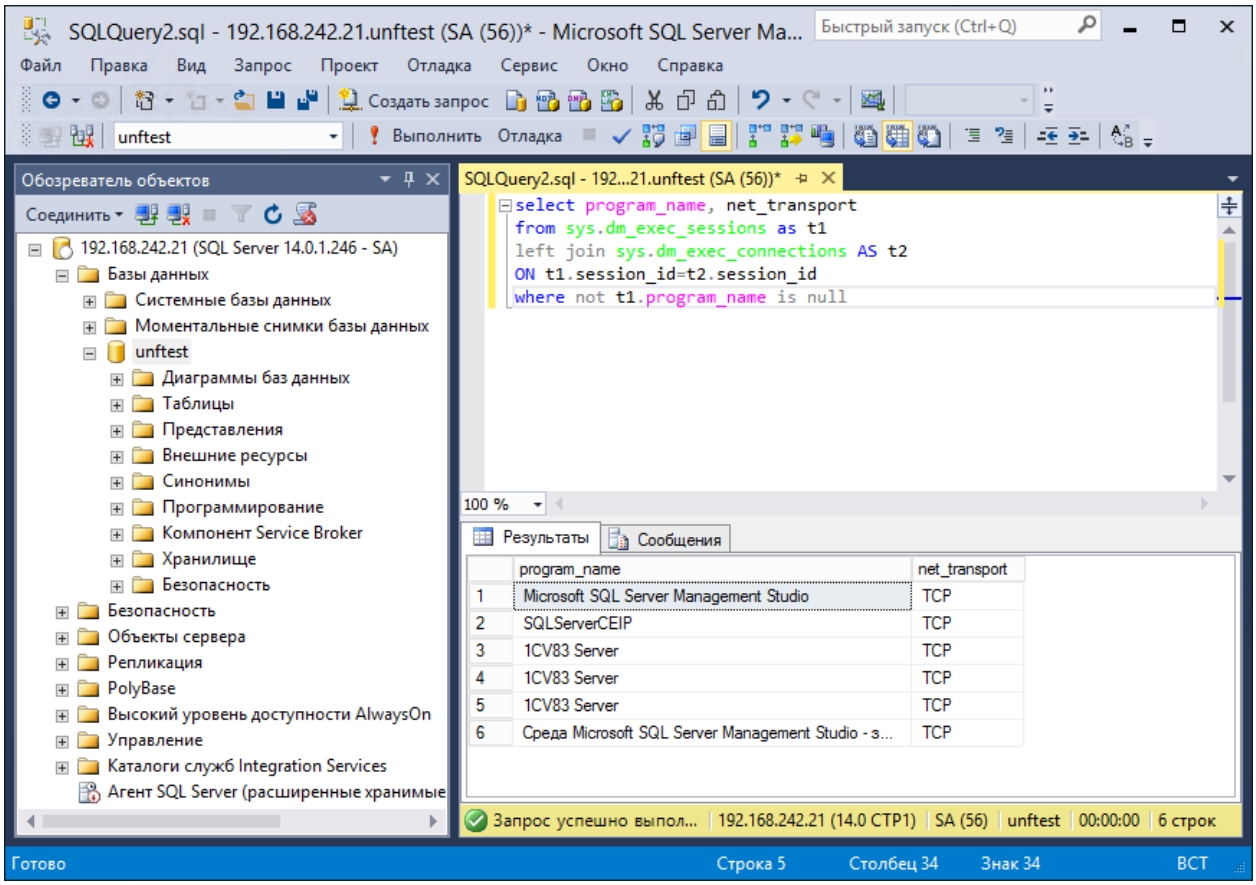

Рисунок 2.9 – Робота з СКБД MS SQL та програмний засіб для роботи з нею Microsoft SQL Server Manager

Дана СКБД має багато переваг, порівняно з іншими:

- легка масштабованість;
- чудова продуктивність;
- висока швидкодія;
- просто використання;
- використовувана мова T-SQL надає додаткові переваги при роботі;
- проста в роботі.

Проте, вона має три великих недоліки:

1) СКБД MS SQL може розгортатись та працювати тільки під управлінням ОС Windows, тобто являється моноплатформенною;
2) хостинг-сервіси для надання послуг по використанню даної СКБД беруть додаткові кошти;
3) комерційне використання СКБД MS SQL, як і майже всі інші продукти компанії Microsoft, являється платним і коштує чималих грошей.

2.3.3 СКБД MySQL

MySQL – вільна реляційна система керування базами даних. Розробку та підтримку MySQL здійснює корпорація Oracle, яка отримала права на торгівельну марку разом з поглиненою Sun Microsystems, яка раніше придбала шведську компанію MySQL AB[3].

MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені (remote) клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СКБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СКБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СКБД MySQL постійно з'являються нові типи таблиць[3].

Оскільки СКБД MySQL являється дуже поширеною, то вона має API-інтерфейс для роботи з нею з-під різних мов програмування, таких як: Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, Python, Ruby та ін.

MySQL є клієнт-серверної системою, що включає багатопотоковий SQL-сервер, що підтримує різні платформи, кілька клієнтських програм і бібліотек, інструменти адміністрування і широкий діапазон програмних інтерфейсів додатків (API-інтерфейсів) [3].

MySql – невеликий компактний багатопотоковий сервер баз даних. MySql характеризується великою швидкістю, стійкістю і легкістю у використанні. Коротко перерахуємо основні достоїнства MySql:

- підтримується необмежене число користувачів, що одночасно працюють з базою даних.
- кількість рядків у таблицях може досягати 50 млн.
- швидке виконання команд;
- проста і ефективна система безпеки.

MySql дійсно дуже швидкий сервер, але для досягнення цього розробникам довелося пожертвувати деякими вимогами до реляційних СКБД. Тому в MySql відсутні:

- підтримка вкладених запитів;

- підтримка зовнішніх ключів;
- підтримка тригерів і збережених процедур;
- підтримки представлень view.

MySQL – кросплатформена система. Її можна використовувати практично у всіх сучасних операційних системах, у тому числі Windows, Linux, Mac OS, Solaris та ін. Так само має безліч програмних інтерфейсів (API), завдяки яким до бази даних MySQL можуть підключатися додатки, створені за допомогою C / C ++, Java, Perl, PHP, Python, Tcl, ODBC, NET і Visual Studio. MySQL має відмінні технічні характеристики: многопоточність, багатокористувацький доступ, швидкодія, масштабованість. MySQL має розвинену систему забезпечення безпеки та розмежування доступу на основі системи привілегії. MySQL є реляційною СКБД.

Працювати з MySQL можна як через графічну web-утиліту PhpMyAdmin (рис. 2.7), так і через консоль (рис. 2.10).

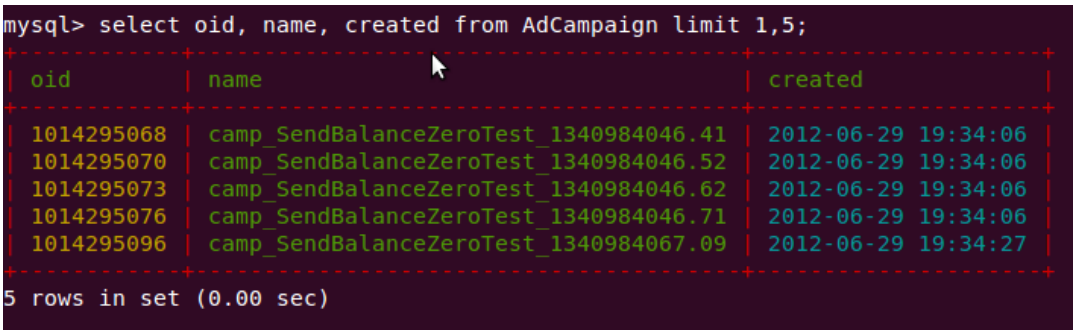

Рисунок 2.10 – Робота з MySQL через консоль

Хоча це не означає, що з СКБД MySQL можна працювати тільки так, існує безліч інших інструментів для роботи з нею.

2.4 Веб-сервер Apache HTTP Server

Apache – веб-сервер з відкритим сирцевим кодом. Він являється кросплатформенним програмним засобом, підтримує ОС Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Основними достоїнствами веб-серверу Apache вважаються надійність і гнучкість конфігурації. Він дозволяє підключати зовнішні модулі для

надання даних, використовувати СКБД для аутентифікації користувачів, модифікувати повідомлення про помилки і т. д. Підтримує IPv6.

Ядро Apache включає в себе основні функціональні можливості, такі як обробка конфігураційних файлів, протокол HTTP і система завантаження модулів. Ядро (на відміну від модулів) повністю розробляється Apache Software Foundation, без участі сторонніх програмістів.

Архітектура веб-сервера Apache HTTP Server зображена на рис. 2.11.

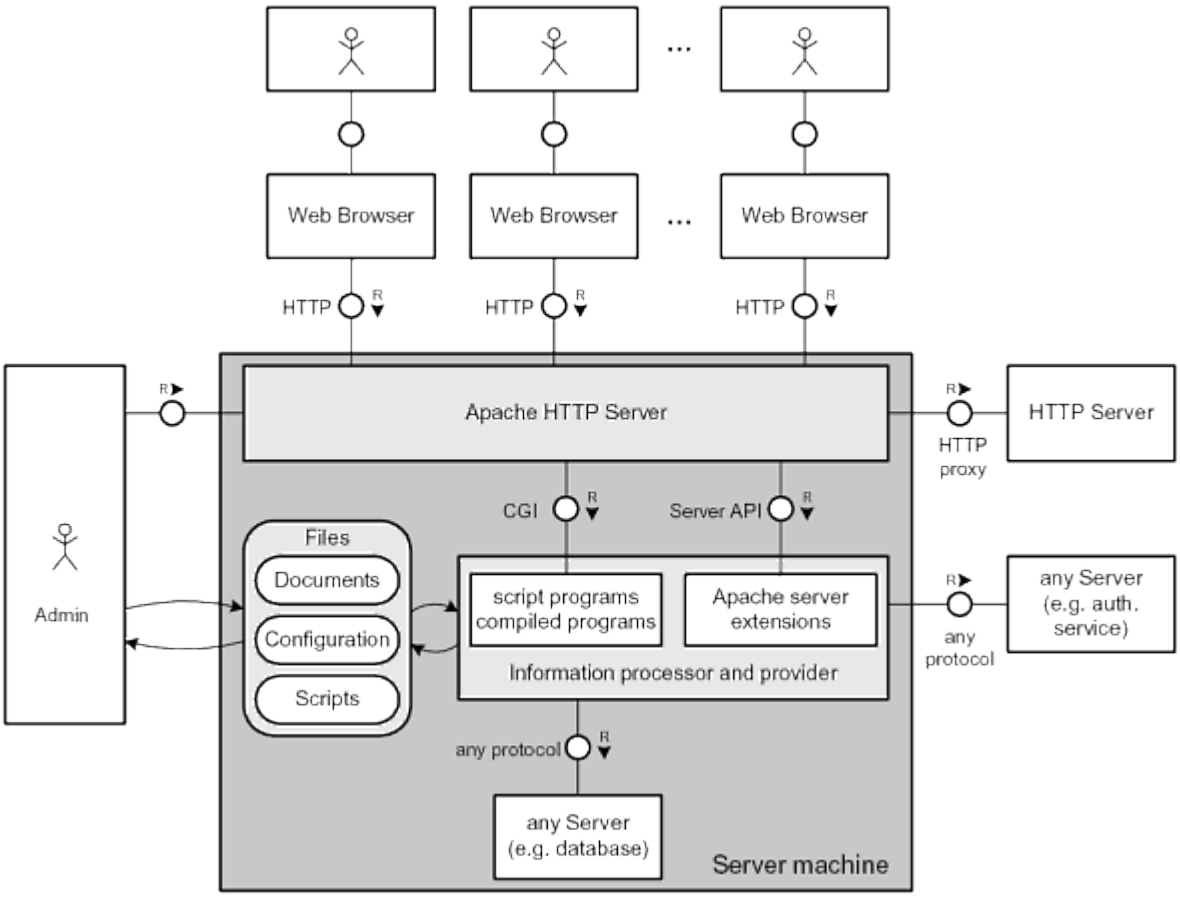

Рисунок 2.11 – Архітектура веб-серверу Apache HTTP Server

Apache HTTP Server підтримує модульність. Існує більше 500 модулів, що виконують різні функції. Частина з них розробляється командою Apache Software Foundation, але основна кількість – окремими open source-розробниками.

Модулі можуть бути як включені до складу сервера в момент компіляції, так і завантажені динамічно, через директиви конфігураційного файлу.

У модулях реалізуються такі речі, як:

- підтримка мов програмування.
- додавання функцій.
- виправлення помилок або модифікація основних функцій.
- посилення безпеки.

Частина веб-додатків, наприклад панелі керування ISPmanager і VDSmanager реалізовані у вигляді модуля Apache[3].

3 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

3.1 Загальні вимоги до інформаційної системи

До веб-орієнованої системи онлайн портфоліо для стабільної і зручної роботи користувачів та самої системи пред’являється ряд вимог. Система повинна забезпечувати:

- зручну систему введення даних користувача;
- виведення на екран дисплея і друкувальний пристрій необхідної інформації по запиту користувача в зручному для перегляду вигляді;
- передачу даних по мережі;
- обробку інформаційних запитів користувача з використанням;
- захист інформації від несанкціонованого доступу шляхом введення системи паролів;
- зручну систему листування серед користувачів.

Перш ніж перейти до проектування бази даних, необхідно визначити основні функціональні вимоги, які пропоновані до системи, тобто визначити діапазон завдань системи та програми бази даних та складу її користувачів.

3.2 Функціональні можливості користувачів системи «Nodesi»

Для адекватної роботи користувачів веб-орієнтованої системи «Nodesi». Розмежування прав доступу в веб-орієнтованій системі «Nodesi» здійснюється шляхом дозволу / заборони на редагування та перегляд певної інформації певної групи користувачів.

В веб-орієнтованій системі онлайн-портфоліо «Nodesi» функціонують чотири групи користувачів:

1) «Користувач-гість»;
2) «Користувач-дизайнер»;
3) «Користувач-роботодавець»;
4) «Користувач-модератор»;

Кожен з наведених типів користувачів має свої певні недоліки і переваги перед іншим типом користувачів. Одні можуть викладати та оцінювати творчі роботи учасників, а інші мають адміністраторські функції: блокування певних робіт, блокування користувачів тощо.

Базовий тип користувача – «користувач-гість» має лише можливості перегляду робіт та загальні можливості роботи з веб-системою. Функціональні можливості даного типу користувача наведені на рис. 3.1.

Рисунок 3.1 – Функціональні можливості користувача-гостя

Права користувача-гостя видаються відразу при переході людиною на веб-сайт системи.

Наступний тип користувача веб-орієнтованої системи онлайн-портфоліо «Nodesi» – користувач-дизайнер має більш широкі можливості, порівняно з попереднім типом. Його функціональні можливості наведені на рис. 3.2.

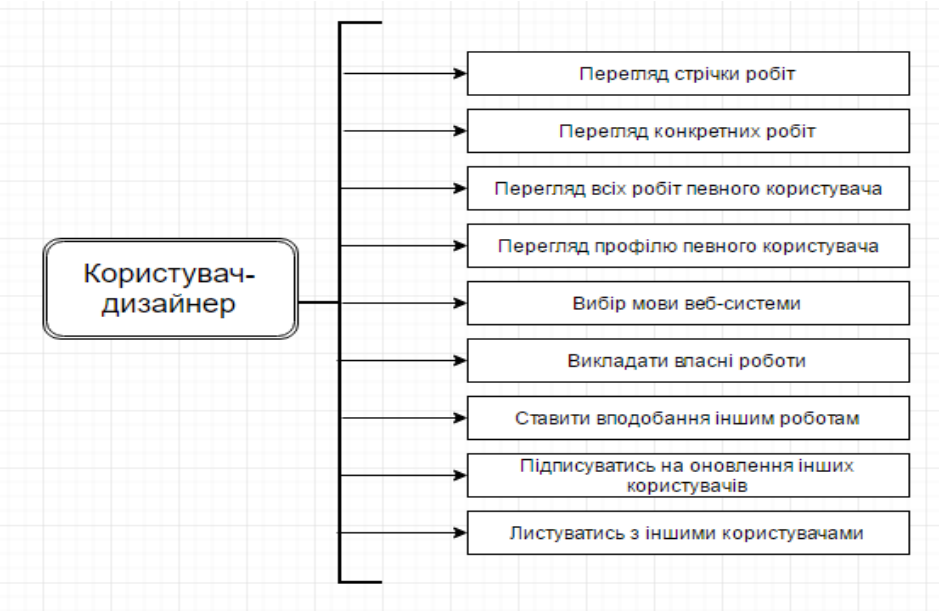

Рисунок 3.2 – Функціональні можливості користувача-дизайнера

Отримати переваги даного типу користувача можна пройшовши реєстрацію, або, якщо такий аккаунт вже існує – пройти процедуру авторизації.

Як видно з рис. 3.2. функціональні можливості дещо розширились, порівняно з попереднім типом користувача. Додались можливості більш точніше відображати власні вподобання, ставити вподобання (англ. Like) певним роботам, проводити листування з іншими користувачами тощо.

Користувач-роботодавець має трішки «врізані» можливості відносно користувача-дизайнера, проте це підтверджує його статус роботодавця та дає іншим користувачам зрозуміти, що їм пише саме роботодавець, а не фейк (англ. Fake).

Функціональні можливості користувача-роботодавця наведені на рис. 3.3.

Рисунок 3.3 – Функціональні можливості користувача-роботодавця

Наступний і останній тип користувачів, який має адміністраторські права та може впливати на інших користувачів – «користувач-модератор». Даний тип користувачів необхідний для того, аби певні люди, працюючі на веб-орієнтовану систему онлайн-портфоліо «Nodesi» могли відстежувати неправомірні дії інших користувачів (наприклад, коли дехто завантажує чужі роботи – плагіат чи завантажує неправомірний матеріал тощо).

Даний тип користувацького аккаунту не можна получити, пройшовши процедуру реєстрації, він назначається тільки власниками веб-сервісу.

Функціональні можливості даного типу аккаунту користувачів зображені на рис. 3.4.

Рисунок 3.4 – Функціональні можливості користувача-модератора

Після визначення типів користувачів та їх функціональні можливості в веб-орієнтованій системі онлайн-портфоліо «Nodesi» можна перейти до проектування бази системи.

3.3 Проектування бази даних веб-орієнтованої системи «Nodesi»

Обрана архітектура веб-орієнтованої системи, що проектується, реалізує архітектуру «клієнт-сервер» та передбачає, що БД може бути розміщена як на сервері локальної комп'ютерної мережі так і на видаленому (remote) сервері. На робочих станціях розміщують клієнтську програму, що формує і відсилає запити віддаленого серверу з БД, який забезпечує виконання запиту і видачу клієнтові результату. Вся обробка інформації відбувається на віддаленому сервері.

Основним компонентом реляційних БД є таблиця. Таблиця використовується для структуризації та зберігання інформації. У реляційних БД кожна клітинка таблиці містить одне значення. Крім того, всередині однієї БД існують взаємозв'язки між таблицями, кожна з яких задає спільне користування даними таблиці[3].

Визначивши функціональні вимоги до web-системи компанії Carrot Studio, використовуючи моделювання SQL / DDL, спроектована база даних,

що реалізує модель «Сутність – Зв'язок». При розробці системи була створена база даних, за допомогою СКБД MySQL. Визначено основні сутності проектованої системи. Сутність – це суб'єкт, місце, річ, подія або поняття, що містять інформацію. Точніше, сутність – це набір (об'єднання) об'єктів, званих екземплярами. Кожен екземпляр сутності володіє набором характеристик. У логічної моделі БД всі ці характеристики називаються атрибутами сутності[11].

Перший етап процесу проектування бази даних називається концептуальним проектуванням бази даних. Він полягає у створенні концептуальної моделі даних предметної області. Ця модель даних створюється на основі функціональних вимог користувачів. Концептуальне проектування бази даних абсолютно не залежить від таких подробиць її реалізації, як тип обраної СКБД, набір створюваних прикладних програм, використовувані мови програмування, тип обраної обчислювальної платформи, а також від будь-яких інших особливостей фізичної реалізації. Концептуальне проектування – створення концептуального уявлення бази даних, що включає визначення типів найважливіших сутностей та існуючих між ними зв'язків і атрибутів. Послідовність етапів проектування концептуальної моделі даних: визначення сутностей; визначення взаємозв'язків між сутностями; визначення атрибутів сутностей; завдання первинних і альтернативних ключів[12].

3.3.1 Опис таблиць бази даних веб-системи «Nodesi»

Для бази даних даної веб-системи визначені наступні 7 сутностей:

1) користувачі;
2) тип користувача;
3) роботи користувача;
4) тип робіт користувача;
5) теги роботи користувача;
6) коментарі до роботи користувача;
7) статистика профілю користувача.

Кожна сутність містить первинний ключ, призначений для унікальної ідентифікації екземпляра сутності. При створенні сутності необхідно виділити групу атрибутів, які потенційно можуть стати первинним ключем, потім зробити відбір атрибутів для включення до складу первинного ключа. Первинний ключ повинен бути підібраний таким чином, щоб за значеннями

атрибутів, в нього включених, можна було точно ідентифікувати примірник сутності, крім того ніякий з атрибутів первинного ключа не повинен мати нульове значення. Значення атрибутів первинного ключа не повинні змінюватися. Якщо значення змінилося, значить, це вже інший примірник сутності[12].

При виборі первинного ключа можна внести в сутність додатковий атрибут і зробити його ключем. Так, для визначення первинного ключа часто використовують унікальні номери, які можуть автоматично генеруватися системою при додаванні екземпляра сутності в БД. Застосування унікальних номерів полегшує процес індексації та пошуку в БД.

Список всіх сутностей бази даних веб-орієнтованої системи онлайн-портфоліо подані в вигляді табл. 3.1.

Таблиця 3.1 – Список всіх сутностей БД веб-системи «Nodesi»

User	Користувачі системи
User_type	Тип користувача системи
Works	Роботи користувачів
Work_type	Тип робіт користувачів
Tag	Теги до роботи користувачів
Comments	Коментарі до роботи користувачів
Statistics	Статистика користувача

Тепер визначимо атрибути для кожної сутності, а також поставимо для всіх сутностей первинні ключі. Відомості про таблиці бази даних представлені нижче у вигляді таблиць (табл. 3.2 – 3.8).

Таблиця 3.2 – Типи користувачів веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Name	Назва типу	varchar(250)

Таблиця 3.3 – Користувачі веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Login	Логін користувача	varchar(250)
3	Password	Пароль користувача	varchar(256)

4	Name	ФІО користувача	text
5	Description	Опис аккаунту користувача	longtext
6	Acc_type_FK	Тип аккаунту користувача	int(10)

Таблиця 3.4 – Роботи користувачів веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Title	Назва роботи	varchar(256)
3	Description	Опис роботи	text
4	Imgs	Картинки роботи	text
5	Likes	Кількість вподобань роботи	int(10)
6	Watchs	Кількість переглядів роботи	int(10)
7	Comments_FK	Коментарі роботи	int(10)
8	Author_FK	Автор роботи	int(10)
9	Work_type_FK	Тип роботи	int(10)

Таблиця 3.5 – Тип робіт користувачів веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Name	Назва типу роботи	

Таблиця 3.6 – Коментарі до робіт веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Text	Текст коментаря	text
3	Add_date	Дата додавання коментаря	datetime
4	User_FK	Користувач, залишивший коментар	int(10)

Таблиця 3.7 – Теги до робіт веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Name	Назва тегу	varchar(256)

Таблиця 3.8 – Статистика користувача веб-системи «Nodesi»

№	Поле	Атрибут	Тип
1	ID	Унікальний ідентифікатор	int(10)
2	Data	Данні статистики	longtext
3	User_FK	Статистика користувача	int(10)

Створену базу даних для реалізації інтернет-додатку для компанії можна вважати нормалізованою, так як її таблиці знаходяться в третій нормальній формі. Тобто не ключові стовпці в таблиці не залежать від інших не ключових стовпців, а залежать тільки від первинного ключа. Тому, таблиці також знаходяться в першій і другій нормальній формі.

Перша нормальна форма забороняє повторювання стовпців, забороняє множинні стовпці і вимагає визначити первинний ключ для таблиці. Друга нормальна форма вимагає, щоб не ключові стовпці таблиць залежали від первинного ключа в цілому, але не від його частини.

Відношення таблиць і взаємодія бази даних веб-орієнтованої системи онлайн-портфоліо «Nodesi» в цілому зображена на рис. 3.5.

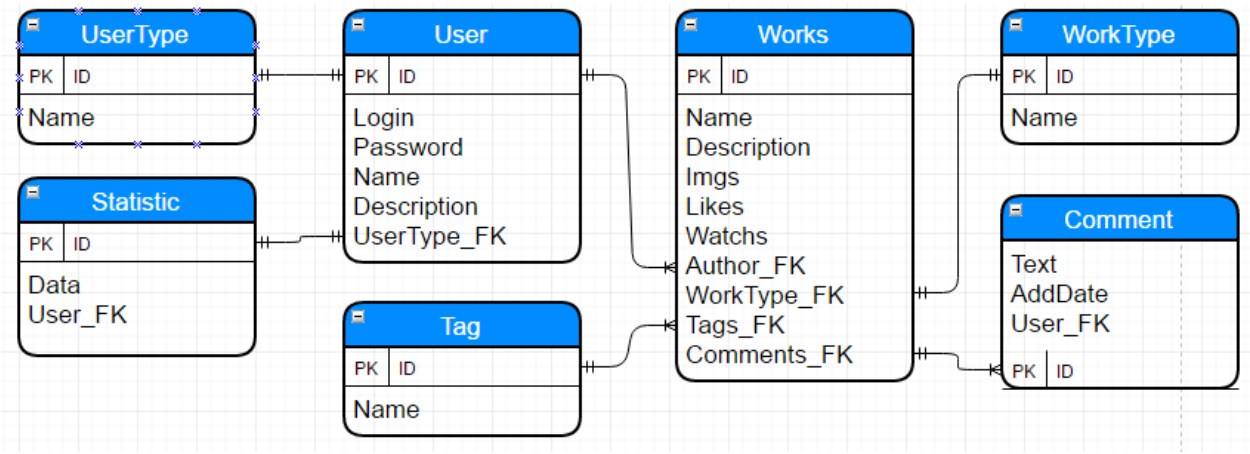

Рисунок 3.5 – Відношення таблиць БД веб-системи «Nodesi»

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Схема роботи веб-орієнтованої системи «Nodesi»

Веб-додаток складається з клієнтської і серверної частин, тим самим реалізуючи технологію «клієнт-сервер».

Клієнтська частина реалізує інтерфейс користувача, формує запити до сервера і обробляє відповіді від нього.

Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу інтернет з використанням протоколу HTTP.

Архітектура клієнт-сервер застосовується у великій кількості мережевих технологій, що використовуються для доступу до різних мережевих сервісів.

Схема роботи веб-орієнтованої системи онлайн-портфоліо «Nodesi» зображена на рис. 4.1.

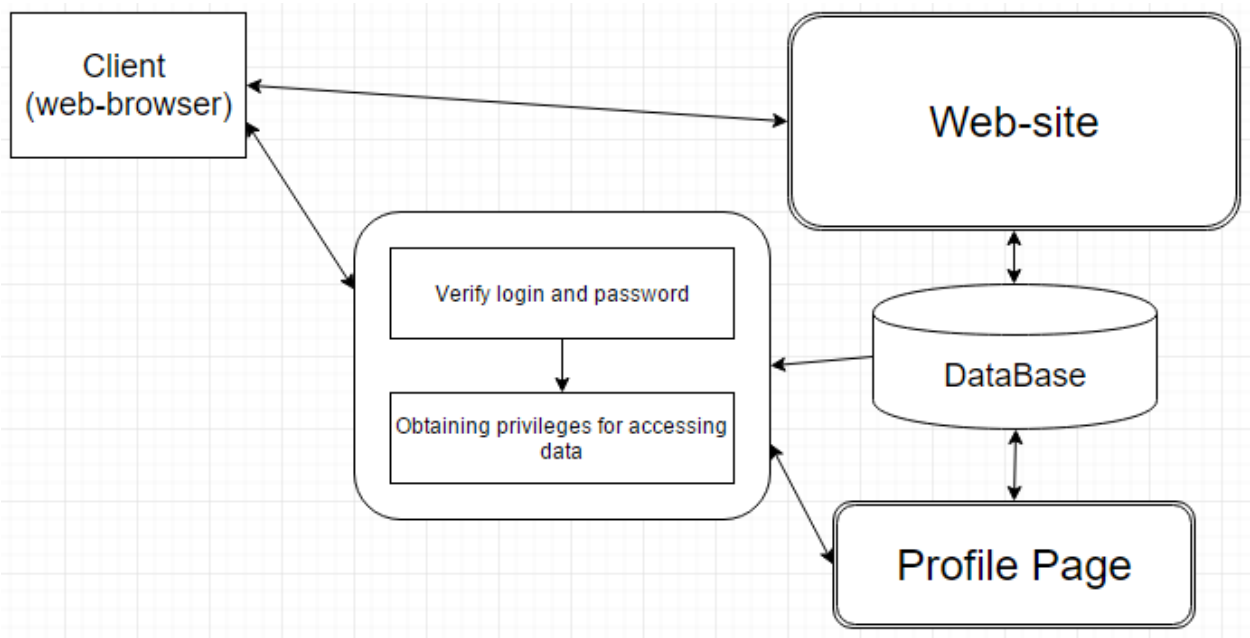

Рисунок 4.1 – Схема роботи веб-системи «Nodesi»

4.2 Реалізація структури веб-орієнтованої системи онлайн-портфоліо

Для зручної роботи будь-якої веб-орієнтованої системи вона повинна мати зручне маню навігації.

Меню навігації – це критично важливий елемент будь-якого веб-сайту, тому до нього потрібно поставитися з усією серйозністю. Візуально воно може бути будь-яким: вертикальним, горизонтальним, складеним з іконок або «плаваючим», але є одна обов'язкова умова – воно повинно допомагати відвідувачеві чи користувачеві знаходити потрібні розділи, служити йому путівником і опорою[13].

Окрім меню навігацію основними деталями, що впливають на враження користувача є наявність зображень і організація елементів на сторінці.

Для розробки дизайну використані принципи, визначені в [14], а саме: проста навігація, дизайнерські рішення колірної гами, центральне вирівнювання, виділення областей кольором, великий текст, яскраві кольори, анімація, яка не відволікає від основного заняття.

З наведених вище даних можна розробити шаблон для веб-сторінок системи. Шаблон головної сторінки системи зображений на рис. 4.2.

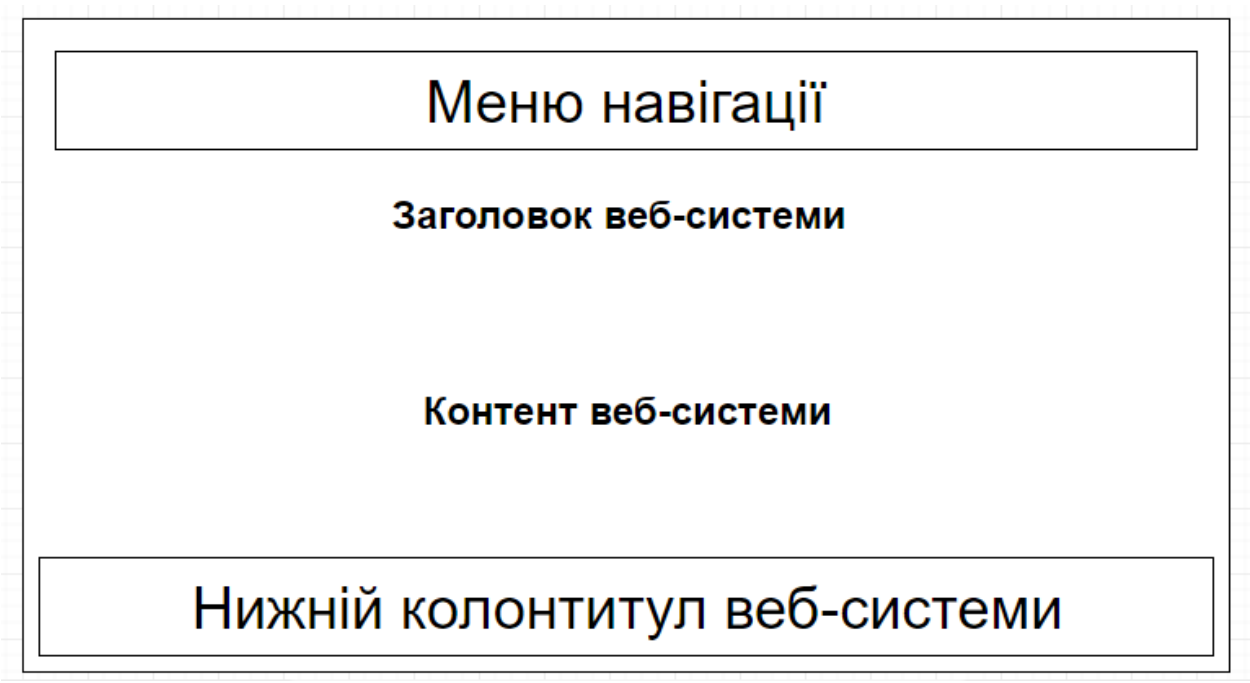

Рисунок 4.2 – Шаблон головної сторінки веб-орієнтованої системи «Nodesi»

Для інших сторінок веб-орієнтованої системи онлайн-портфоліо «Nodesi» розроблено дуже схожий, але трішки інакший шаблон сторінок.

На інших сторінках веб-системи «Заголовок системи» опущений та винесений в вигляді логотипу в лівий верхній кут веб-сторінки.

Загалом, інші сторінки дещо відрізняються одна від одної, але мають загальні особливості: верхнє меню навігації, контент в середині сторінки та нижній колонтитул (footer).

Відкрита сторінка з певною роботою завантажується без переходу на нову сторінку веб-системи, а за допомогою технології Ajax завантажує її на тій самій сторінці робіт в вигляді модального вікна.

Шаблон інших сторінок веб-орієнтованої системи онлайн-портфоліо «Nodesi» зображений на рис. 4.3.

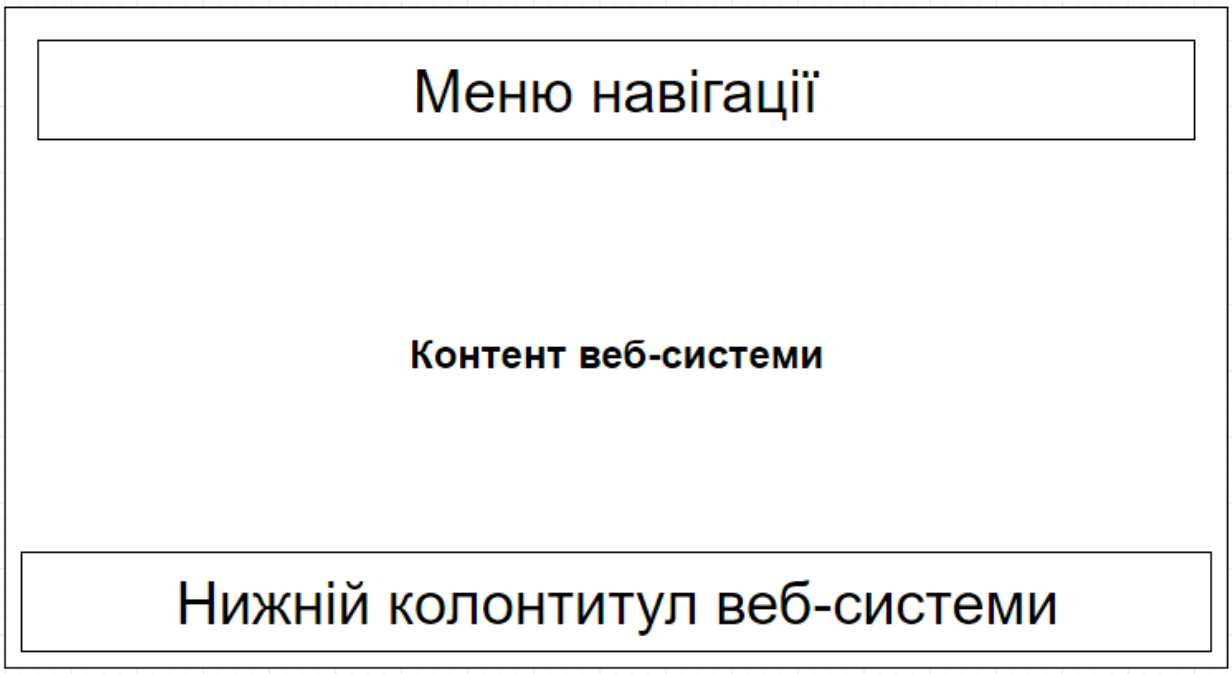

Рисунок 4.3 – Шаблон інших сторінок веб-системи «Nodesi»

4.3 Розробка інтерфейсу користувача веб-системи

Притримуючись шаблону головної сторінки та принципів дизайну, визначених в підрозділі 4.1 була розроблена головна сторінка веб-орієнтованої системи онлайн-портфоліо «Nodesi».

Всі сторінки даної веб-системи онлайн-портфоліо містять зручне меню навігації зверху сторінки, яке дозволяє перейти до деяких розділів системи:

1) головна сторінка;
2) власне портфоліо;
3) топ дизайнерів системи;
4) профіль;

5) список сповіщень від системи та список оновлень користувачів, на які підписаний поточний користувач;
6) список повідомлень користувачеві;
7) кнопка дії з аккаунтом.

Головна сторінка зображена на рис. 4.4.

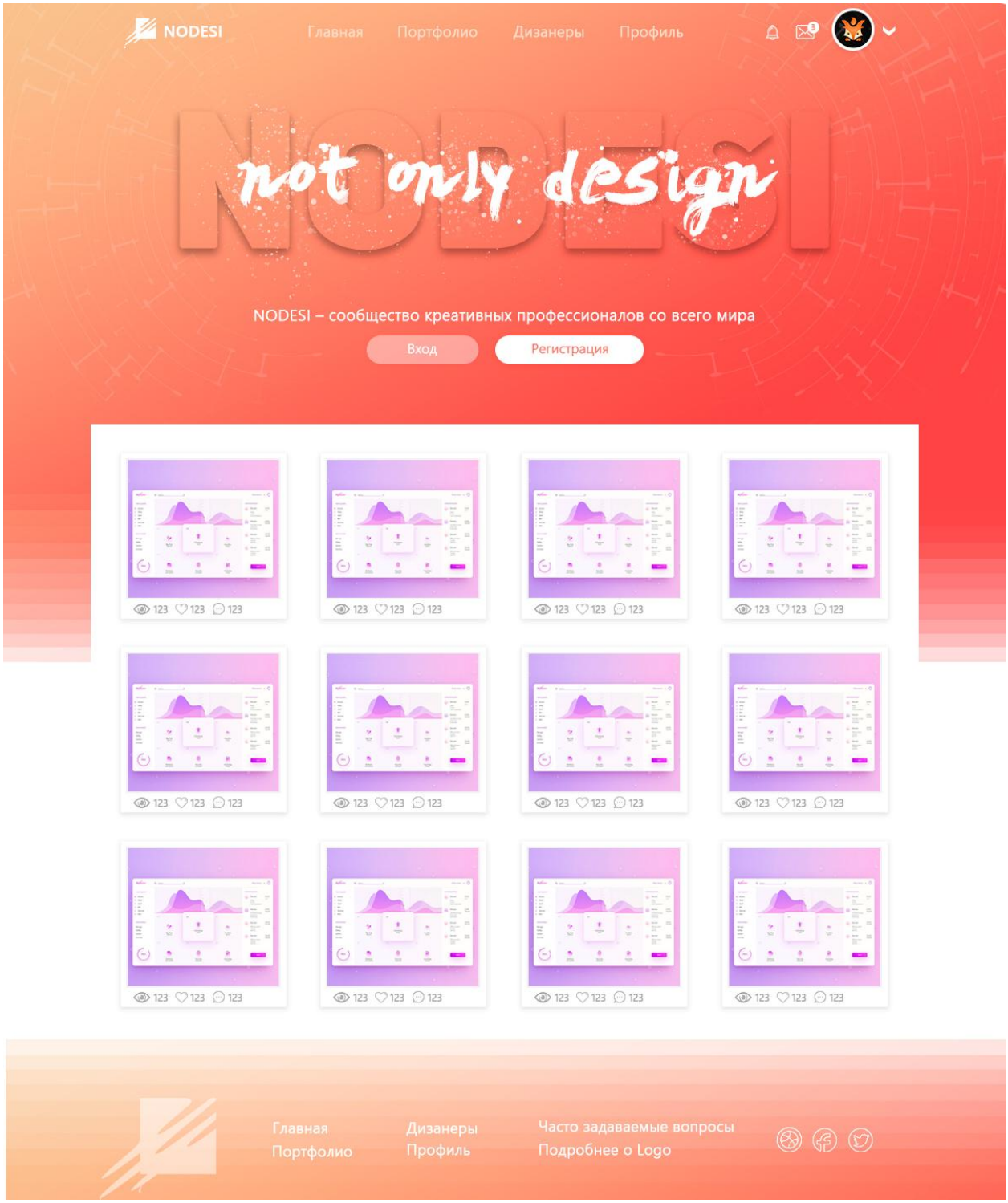

Рисунок 4.4 – Головна сторінка веб-орієнтованої системи «Nodesi»

Як видно з наведеного рисунку головна сторінка повністю відповідає приведеному на рис. 4.2 шаблону головної сторінки. Контентом даної веб-сторінки служать роботи користувачів, які були останніми завантажені до системи, або роботи користувачів, на які підписаний ввійшовший в системи користувач.

Інші сторінки веб-системи реалізують шаблон зображений на рис. 4.3, наприклад сторінка профілю. Вона зображена на рис. 4.5.

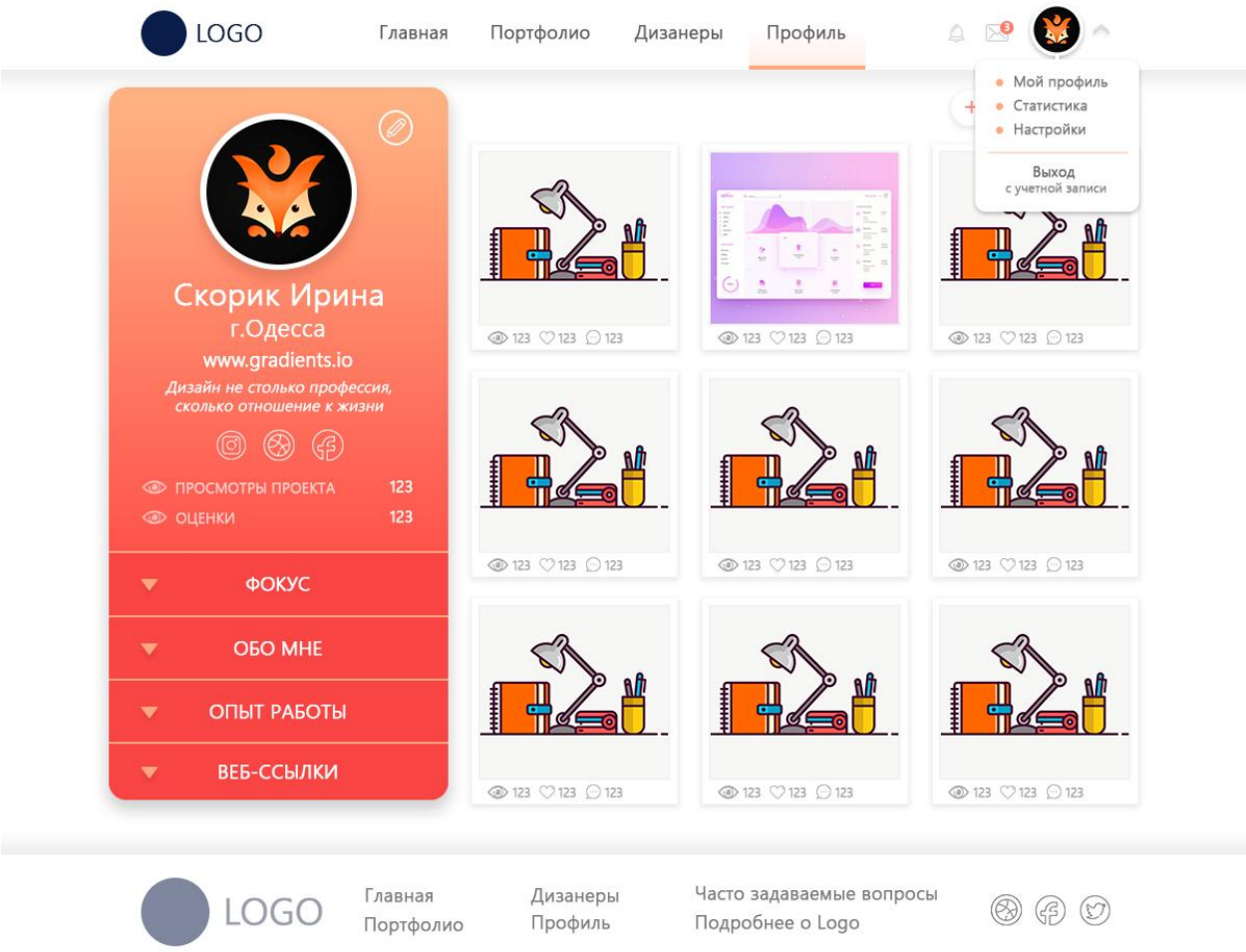

Рисунок 4.5 – Сторінка профілю користувача

Розроблена веб-орієнтована система відповідає всім сучасним стандартам розробки веб-додатків, реалізує та відповідає шаблону проектування MVC, має відповідний до стандартів, визначених в [14] дизайн, надає зручні можливості для роботи користувачів, та реалізує всі необхідні функції, визначені в підрозділі 3.2.

ВИСНОВКИ

В результаті дипломного проектування було розроблені веб-орієнтовану систему онлайн-портфоліо «Nodesi».

Актуальність реалізації даної системи полягає в необхідності вирішення проблем дизайнерів по пошуку роботи та роботодавців, які займаються наданням послуг, так чи інакше пов’язаних з дизайном (рекламні компанії, розробка веб-систем тощо), в пошукові талановитих співробітників.

Значними перевагами розробленої веб-системи перед іншими, розглянути являються її повна безкоштовність, інтеграції з соціальними мережами та зручна, для вітчизняних користувачів, мова самої системи.

За допомогою розробленої веб-орієнтованої системи можна:

- легко і швидко створити свою персональну сторінку портфоліо;
- оцінювати роботи інших користувачів і дивитися свої оцінки;
- публікувати необмежену кількість робіт;
- підключити соціальні мережі і ділиться своїми роботами там;
- роботодавцю вибирати по роботах співробітника.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке фріланс? Як заробити фрілансом? [Електронний ресурс] – Режим доступу: http://rada.lviv.ua/2277-scho-take-frilans-yak-zarobyty-frilansom/
2. Сімейство блогів «Tumblr» [Електронний ресурс] – Режим доступу: https://www.tumblr.com/
3. Википедия – свободная энциклопедия [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org
4. Онлайн портфоліо «Behance» [Електронний ресурс] – Режим доступу: https://www.behance.net
5. Спільнота дизайнерів «Dribbble» [Електронний ресурс] – Режим доступу: https://dribbble.com
6. Dribbble социальная сеть для дизайнеров [Електронний ресурс] – Режим доступу: http://www.cossa.ru/152/25904/
7. The Java EE 5 Tutorial [Електронний ресурс] – Режим доступу: https://docs.oracle.com/cd/E19575-01/819-3669/6n5sg7arb/index.html
8. Web-разработка на Python глазами PHP-программиста [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/243961/
9. PHP – найбільш популярна мова для веб-програмування [Електронний ресурс] – Режим доступу: http://asaweb.com.ua/ua/php-5/
10. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных [Електронний ресурс] – Режим доступу: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/
11. MySQL 5.0. Библиотека программиста – СПб.: Питер, 2010. – 253 с.: ил.
12. Дейт К. Дж. Введение в системы баз данных. – СПб.: Вильямс, 2005. – 1316 с.: ил.
13. Нельсен Якоб. Веб дизайн. – СПб.: Питер, 2013. – 504 с.: ил.
14. Уэйншенк Сьюзан. 100 главных принципов дизайна. Как удержать внимание. – СПб.: Питер, 2012. – 272 с.: ил.

Д О Д А Т К И

ДОДАТОК А СХЕМА ФУНКЦІОНУВАННЯ ВЕБ-СИСТЕМИ

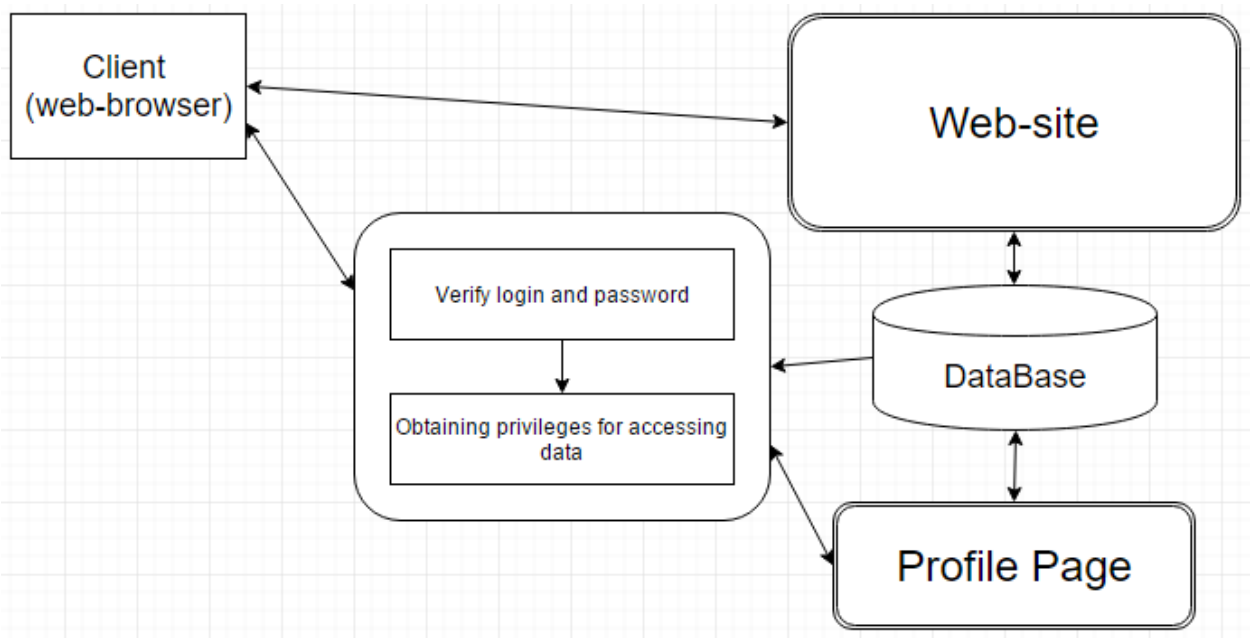

Рисунок А.1 – Схема функціонування веб-орієнтованої системи «Nodesi»

53

ДОДАТОК Б ГРАФІЧНЕ ЗОБРАЖЕННЯ БАЗИ ДАНИХ СИСТЕМИ

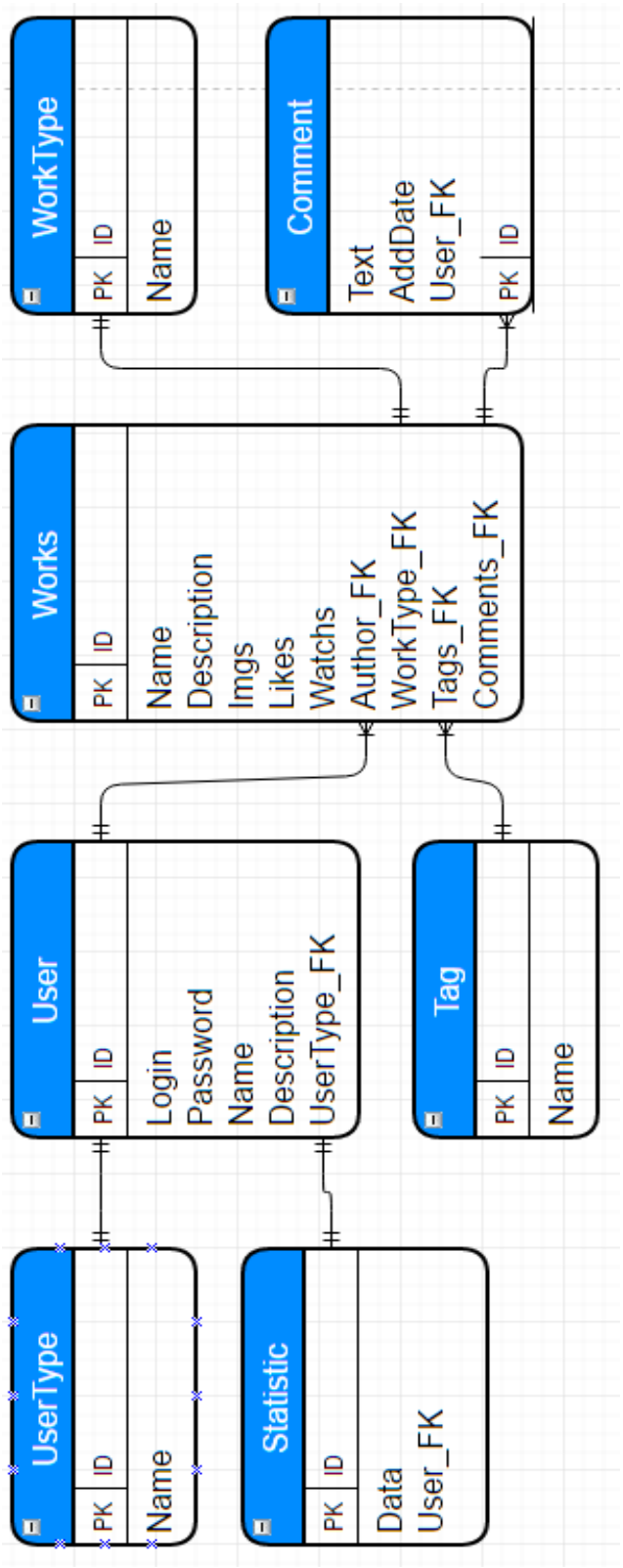

Рисунок Б.1 – Графічне зображення відношень таблиць бази даних системи

ДОДАТОК В ОСНОВНІ КОДИ ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ

IncludesFn.php

```
<?php
class IncludesFn
{
    public static function printHeader($title, $bodyClass = null, $head
= null)
    {
        $header = <<<EOF
<!DOCTYPE html>
<html>
<head>

    <title>{$title}</title>

    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-
scale=1"/>

    <link rel="stylesheet" href="/Libs/FrontEnd/css/eric-meyer-
reset.css" title="no title"/>
    <link rel="stylesheet" href="/Libs/FrontEnd/bootstrap-
3.3.7/css/bootstrap.min.css" media="screen" title="no title" charset="utf-
8"/>
    <link rel="stylesheet" href="/Libs/FrontEnd/css/fonts.css" title="no
title"/>
    <link rel="stylesheet" href="/Libs/FrontEnd/css/main.css" title="no
title"/>
    <link rel="stylesheet" href="/Libs/FrontEnd/css/font-awesome.css"/>

    <link rel="shortcut icon" type="image/png"
href="/Images/Icons/favicon.png"/>

    {$head}
</head>
<body class="{$bodyClass} meScroll-mini">
EOF;
        print($header);

    }

    public static function ReturnRating($int) // From 0 To 100
    {
        $countFull = (int)($int / 20);
        $countHalf = 0;

        if($int %20 != 0)
        {
            $countHalf = 1;
        }

        $countEmpty =  5 - (int)($countFull + $countHalf);

        $arrayRating = [$countFull, $countHalf, $countEmpty];

        $htmlRating = "";

        foreach ($arrayRating as $key => $value)
        {
```

```
                for ($i = 0; $i < $value; $i++)
                {
                    if($key == 0)
                    {
                        $htmlRating .= '<i class="fa fa-star" aria-
hidden="true"></i>';
                    }
                    else if($key == 1)
                    {
                        $htmlRating .= '<i class="fa fa-star-half-o" aria-
hidden="true"></i>';
                    }
                    else
                    {
                        $htmlRating .= '<i class="fa fa-star-o" aria-
hidden="true"></i>';
                    }
                }
            }

            return $htmlRating;
        }

        public static function ReturnIconCategory($name)
        {
            switch ($name)
            {
                case ("web"):
                    return "/Images/Icons/web-03.svg"; break;
            }
        }

        public static function printMenuCategory($sql, $url, $name, $default
= null, $prefix = null)
        {
            $category = R::getAll($sql);

            $li = "";

            foreach ($category as $value)
            {
                $href = $value[$url];
                $title = $value[$name];

                if($href == $default)
                {
                    $li .= '<li><a href="' . $prefix . $href . '"
class="active"><i class="fa fa-check-circle-o" aria-hidden="true"></i> ' .
Languages::Translate($title) . '</a></li>';
                    continue;
                }

                $li .= '<li><a href="' . $prefix . $href . '"><i class="fa
fa-circle-o" aria-hidden="true"></i> ' . Languages::Translate($title) .
'</a></li>';
            }
            $menu = <<<EOF
                <ul>
                    {$li}
                </ul>
      EOF;
            return $menu;
        }
```

```
          public static function printMenu($url, $menuName, $className = null,
$shell = null, $print = null)
          {
              $language = new Languages();
              $menu = "";

              $arrayMenu = [
                  "/" => [
                      "text" => 'dashboard_menu_home_page',
                      "class" => "home-mu"
                  ],
                  "/services/" => [
                      "text" => 'dashboard_menu_services',
                      "class" => "services-mu"
                  ],
                  "/aboutus/" => [
                      "text" => 'site_about',
                      "class" => "aboutus-mu"
                  ],
                  "/blog/" => [
                      "text" => 'dashboard_menu_blog',
                      "class" => "blog-mu"
                  ],
                  "/portfolio/" => [
                      "text" => 'dashboard_menu_portfolio',
                      "class" => "portfolio-mu"
                  ],
                  "/contacts/" => [
                      "text" => 'dashboard_menu_contacts',
                      "class" => "contacts-mu"
                  ],
              ];

      //        "/shop/" => 'dashboard_menu_template_shop',

              if($menuName == "base")
              {
                  $li = "";

                  foreach ($arrayMenu as $key => $value)
                  {
                      if($key == $url)
                      {
                          $li .= '<li><a href="' . $key . '" class="active ' .
$value["class"] . '">' . Languages::Translate($value["text"]) . '</a></li>';
                          continue;
                      }

                      $li .= '<li><a href="' . $key . '" class="' .
$value["class"] . '">' . Languages::Translate($value["text"]) . '</a></li>';
                  }
                  $menu = <<<EOF
                      <ul>
                          {$li}
                      </ul>
      EOF;
              }

      //        $menu = <<<EOF
      //<ul>
      //    <li><a href="/">{$language-
>Translate("dashboard_menu_home_page")}</a></li>
```

```
      //    <li><a href="/services/" class=active>{$language-
>Translate("dashboard_menu_services")}</a></li>
      //    <li><a href="/blog/">{$language-
>Translate("dashboard_menu_blog")}</a></li>
      //    <li><a href="/portfolio/">{$language-
>Translate("dashboard_menu_portfolio")}</a></li>
      //    <li><a href="/contacts/">{$language-
>Translate("dashboard_menu_contacts")}</a></li>
      //</ul>
      //EOF;

              if($shell)
              {
                  $languagesSwitch = Languages::LanguageSwitch("in");

                  $menu = <<<EOF
                  <div class="header-any-page clearfix {$className}">
                      <div class="head-logo">
                          <a href="/"><div class="link-go-home"><img
alt="Carrot" src="/Images/Home/logoBold-03.svg"></div></a>
                      </div>
                      <div class="head-menu">
                          <button><i class="fa fa-bars" aria-
hidden="true"></i></button>
                          {$menu}
                      </div>
                      <div class="head-user">
                          <div class="dropdown switch-language contacts-drop
in">
                            <button class="login-button clear-button"
type="button" data-toggle="dropdown">
                            <i class="fa fa-phone" aria-hidden="true"></i>
                            </button>
                            <ul class="dropdown-menu dropdown-menu-right
shadow-none">
                              <li><strong>PъPsPSC,P°PєC,C‹</strong></li>
                              <li><a href="tel:380487000393"><i class="fa fa-
phone" aria-hidden="true"></i> +38 (048) 7000 393</a></li>
                              <li><a href="tel:380949523393"><i class="fa fa-
phone" aria-hidden="true"></i> +38 (094) 95 23 393</a></li>
                              <li><a href="tel:380932572284"><i class="fa fa-
mobile" aria-hidden="true"></i> +38 (093) 25 72 284</a></li>
                              <li><a href="tel:380976215379"><i class="fa fa-
mobile" aria-hidden="true"></i> +38 (097) 62 15 379</a></li>
                              <li><a href="mail:manager@std-carrot.com"><i
class="fa fa-envelope" aria-hidden="true"></i> manager@std-
carrot.com</a></li>
                            </ul>
                          </div>
                          {$languagesSwitch}
                          <a href="/login/"><button class="login-button clear-
button"><span><i class="fa fa-user" aria-
hidden="true"></i></span></button></a>
                      </div>
                      <div class="menu-services">
                          <ul>
                              <li><strong><a href="/services/web">{$language-
>Translate("site_websites")}</a></strong></li>
                              <li><a href="/services/landing">Landing
Page</a></li>
                              <li><a href="/services/shop">P□PSC,PµCЪPSPµC,
PjP°PiP°P·PëPS</a></li>
                          </ul>
```

```
                            <ul>
                                <li><strong><a href="/services/web">{$language-
>Translate("site_support_websites")}</a></strong></li>
                            </ul>
                            <ul>
                                <li><strong><a href="/services/apps">{$language-
>Translate("site_applications")}</a></strong></li>
                            </ul>
                            <ul>
                                <li><strong><a href="/services/ab">{$language-
>Translate("site_ab_tests")}</a></strong></li>
                            </ul>
                            <ul>
                                <li><strong><a href="/services/outdoor-
advertising">{$language-
>Translate("site_outdoor_advertising")}</a></strong></li>
                            </ul>
                            <ul>
                                <li><strong><a
href="/services/polygraphy">{$language-
>Translate("site_polygraphy")}</a></strong></li>
                            </ul>
                        </div>
                    </div>
        EOF;
                }

                if($print)
                {
                    return print($menu);
                }
                return $menu;
            }
        }

        BF.php

        <?php
        class BF
        {
            public function CheckUserInSystem($login, $password)
            {
                $check = R::getRow("SELECT * FROM carrot_users WHERE
carrot_users_login = ? AND carrot_users_password = ?",
[BF::GeneratePass($login), BF::GeneratePass($password)]);

                if(count($check) > 0)
                {
                    return 1;
                }

                return 0;
            }

            public static function ReturnInfoUser($return)
            {
                if(BF::CheckUserInSystem($_SESSION["login"],
$_SESSION["password"]) == 1)
                {
                    $id = R::getRow("SELECT carrot_users_id, carrot_users_name,
carrot_users_permissions  FROM carrot_users WHERE carrot_users_login = ?",
[BF::GeneratePass($_SESSION["login"])]);
```

```
                return $id[$return];
            }

            return false;
        }

        public function RedirectUser($redirectTo, $needResult) // Redirect
To, if result 1, Or Need Result, if result 0
        {

            if($needResult == 1)
            {
                if(BF::CheckUserInSystem($_SESSION["login"],
$_SESSION["password"]) == 1)
                {
                    header("Location: /" . $redirectTo);
                }
            }
            else
            {
                if(BF::CheckUserInSystem($_SESSION["login"],
$_SESSION["password"]) == 0)
                {
                    header("Location: /" . $redirectTo);

                    die();
                }
            }
        }

        public static function LoginUser($login, $password)
        {
            $_SESSION["login"] = $login;
            $_SESSION["password"] = $password;
        }

        public static function QuitUser()
        {
            unset($_SESSION["login"]);
            unset($_SESSION["password"]);
        }

        public static function GeneratePass($text)
        {
            $text = "_23_asd_" . $text . "_asd_324";
            $password = md5($text);
            return $password;
        }

        public static function IncludeScripts($array)
        {
            $script = "";

            foreach($array as $value)
            {
                $script .= <<<EOF
      <script type="text/javascript"
src="/Libs/FrontEnd/{$value}.js"></script>
      EOF;
            }

            return print($script);
        }
```

```
        public static function IncludeStyles($array)
        {
            $style = "";

            foreach($array as $value)
            {
                $style .= <<<EOF
    <link rel="stylesheet" href="/Libs/FrontEnd/{$value}.css">
    EOF;
            }

            return print($style);
        }

        public static function CreateLikeQuery($arrayWithColumns,
$searchText)
        {
            $query = "";

            $explodeText = explode(" ", $searchText);
            $countWords = count($explodeText);
            $countColumns = count($arrayWithColumns);

            for ($i = 0; $i < $countWords; $i++)
            {
                $word = $explodeText[$i];

                $text = "";

                for ($y = 0; $y < $countColumns; $y++)
                {
                    $value = $arrayWithColumns[$y];

                    if($y != $countColumns - 1 && $countColumns != 1)
                    {
                        $text .= "{$value} LIKE '%{$word}%' OR ";

                        continue;
                    }
                    else if($countColumns == 1)
                    {
                        $text .= "{$value} LIKE '%{$word}%'";

                        continue;
                    }

                    $text .= "{$value} LIKE '%{$word}%'";
                }

                if($i != $countWords - 1 && $countWords != 1)
                {
                    $query .= "({$text}) AND ";

                    continue;
                }
                else if($countWords == 1)
                {
                    $query .= $text;

                    continue;
                }
```

```
                $query .= "({$text})";
            }

            return $query;
        }

        public static function ClearText($data)
        {
            return html_entity_decode(html_entity_decode($data));
        }

        public static function ClearCode($data, $type = null, $from =
"array")
        {
            $data = BF::CheckFrom($data, $from);

            switch($type) {
                case("int"):
                    $data = intval($data);
                    break;
                case("float"):
                    $data = floatval($data);
                    break;
                case("bool"):
                    $data = boolval($data);
                    break;
                case("double"):
                    $data = doubleval($data);
                    break;
                case("str"):
                    $data = htmlentities(strval($data));
                    break;
                default:
                    $data = intval($data);
                    break;
            }
            return htmlentities($data);
        }

        public static function CheckFrom($var, $from = "array")
        {
            switch ($from)
            {
                case("array"):
                    if(isset($var))
                    {
                        return $var;
                    }
                    break;
                case("post"):
                    if(isset($_POST[$var]) && $_POST[$var] != null)
                    {
                        return $_POST[$var];
                    }
                    break;
                case("get"):
                    if(isset($_GET[$var]) && $_GET[$var] != null)
                    {
                        return $_GET[$var];
                    }
                    break;
                case("cookie"):
                    if(isset($_COOKIE[$var]) && $_COOKIE[$var] != null)
```

```
                    {
                        return $_COOKIE[$var];
                    }
                    break;
                default:
                    return false;
            }

            return false;
        }

        public static function ReturnStatusAccount($countProjects,
$countTasks)
        {
            if( ($countProjects >= 0 && $countProjects <= 3) && ($countTasks
>= 0 && $countTasks <= 5) )
            {
                return ["dashboard_account_status_basic", 0];
            }
            else if( ($countProjects > 3 && $countProjects <= 5) &&
($countTasks > 5 && $countTasks <= 8) )
            {
                return ["dashboard_account_status_medium", 5];
            }
            else if($countProjects > 6 && $countTasks > 8)
            {
                return ["dashboard_account_status_pro", 15];
            }

            return false;
        }

        public static function UploadFile($name, $path)
        {
            $dataInfo = [];

            $target_dir = $_SERVER["DOCUMENT_ROOT"] . $path;
            $target_file = $target_dir . basename($_FILES[$name]["name"]);

            $dataInfo["fileInfo"] = $target_file;

            $uploadOk = 1;
            $imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
            // Check if image file is a actual image or fake image
            $check = getimagesize($_FILES[$name]["tmp_name"]);

            if($check !== false) {
                $dataInfo["imageMime"] = $check["mime"];
                $uploadOk = 1;
            } else {
                $dataInfo["imageMime"] = "File is not an image.";
                $uploadOk = 0;
            }
            // Check if file already exists
            if (file_exists($target_file)) {
                $dataInfo["imageExists"] = "Sorry, file already exists.";
                $uploadOk = 0;
            }
            // Check file size

            $dataInfo["imageSize"] = $_FILES[$name]["size"];

            if ($_FILES[$name]["size"] > 1000000) {
```

```
                $uploadOk = 0;
            }
            $dataInfo["imageType"] = "Support";

            // Allow certain file formats
            if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
                && $imageFileType != "gif" ) {
                $dataInfo["imageType"] = "Sorry, only JPG, JPEG, PNG & GIF
files are allowed.";
                $uploadOk = 0;
            }
            // Check if $uploadOk is set to 0 by an error
            if ($uploadOk == 0) {
                $dataInfo["imageUploaded"] = "Sorry, your file was not
uploaded.";
                $dataInfo["imageStatus"] = false;
            // if everything is ok, try to upload file
            } else {
                if (move_uploaded_file($_FILES[$name]["tmp_name"],
$target_file)) {
                    $dataInfo["imageUploaded"] = "The file has been
uploaded.";
                    $dataInfo["imageUploadedName"] = basename(
$_FILES[$name]["name"]);
                    $dataInfo["imageStatus"] = true;
                } else {
                    $dataInfo["imageUploaded"] = "Sorry, there was an error
uploading your file.";
                    $dataInfo["imageStatus"] = false;
                }
            }

            return $dataInfo;
        }

        public static function ReturnCondition($data, $var1, $var2)
        {
            if($var1 == $var2)
            {
                return $data;
            }

            return false;
        }

        public static function ReturnPercent($parent, $child)
        {
            if($parent == 0)
            {
                return 0;
            }

            return intval($child * 100 / $parent);
        }

        public static function NotFound()
        {
            header('HTTP/1.1 404 Not Found');
            header("Status: 404 Not Found");
            header('Location: /404/');

            die();
```

```
    }

}
```

Bootstrap.php

```
<?php
session_start();
require_once($_SERVER["DOCUMENT_ROOT"] . "/App/Config/Define.php");

foreach (ReturnListFiles(DIR_APP . "Library/Sweane/") as $path)
{
    require_once $path;
}

if(!BF::ClearCode("language", "str", "cookie"))
{
    setcookie("language", "ru", time() + 86400, "/");
}

require_once(DIR_APP . "Core/Controller.php");
require_once(DIR_APP . "Core/Model.php");
require_once(DIR_APP . "Core/Route.php");
require_once(DIR_APP . "Core/View.php");

function ReturnListFiles($dir)
{
    $message = [];

    if(is_dir($dir))
    {
        if($listDir = opendir($dir))
        {
            while(($file = readdir($listDir)) !== false)
            {
                if(file_exists($dir . $file))
                {
                    if(filetype($dir . $file) == "file")
                    {
                        array_push($message, $dir . $file);
                    }
                }
            }
            closedir($listDir);
        }
    }

    return $message;
}

$route = new Route();
$route->Init();
```