МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Савченко А.С.

«___»___ _____ _____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТРА"

ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: «Web-додаток «Система контролю задоволеності співробітників»»

Виконавець: Таран Кирилл Валентинович

Керівник: к.т.н., професор Воронін Альберт Миколайович

Нормоконтролер: _____ Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за галузями)"

ЗАТВЕРДЖУЮ
Завідувач кафедри

___ Савченко А.С.

" " 2020 р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Тарана Кирилла Валентиновича

1. Тема роботи: «Web-додаток «система контролю задоволеності співробітників»»

Затверджена наказом ректора від 25.09.2019 за № 2175/ст.

2. Термін виконання роботи: з 26.09.2019 до 03.02.2020р.

3. Вихідні данні до роботи: Web-додаток, система контролю задоволеності співробітників

4. Зміст пояснювальної записки: вступ, аналітичний огляд і постановка завдання, висновок.

5. Перелік обов'язкового ілюстративного матеріалу: зображення інструментів розробки, зображення етапів та модулів проекту, приклади PHP коду, приклади Yii2 коду, діаграма БД.

6. Календарний план-графік

№ з/п	Завдання	***Термін виконання***	***Підпис керівника***
1.	Проаналізувати літературу та джерела за темою дипломного проекту.	26.09.19 – 09.09.19	
2	Розроблення та затвердження плану дипломного проекту.	10.11.19 – 20.11.19	
3	Провести консультації з науковим керівником щодо створення першого розділу.	21.11.19 – 30.11.19	
4	Розробка розділу 1: Аналіз предметної області	01.12.19 – 10.12.19	
5	Розробка розділу 2: Аналіз та порівняння обраних інструмнтів і технологій для web-додатку	11.12.19 – 20.12.19	
6	Розробка додатку	21.12.19 – 31.12.19	
7	Розробка розділу 3: Створення Web-додатку «Система контролю задоволеності співробітників»	01.01.20 – 10.01.20	
8	Висновки та оформлення пояснювальної записки дипломного проекту.	11.01.20 – 15.01.20	
9	Підписання необхідних документів у встановленому порядку.	15.01.20 – 21.01.20	
10	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	21.01.20 – 03.02.20	

Дата видачі завдання 26.09.19

Керівник дипломної роботи . Воронін А.Н.

Завдання прийняв до виконання_____ Таран К.В.

(підпис випускника) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту роботи «Web-додаток «Система контролю задоволеності співробітниками»» викладена на 94 с., містить 47 рис., 19 літературних джерел.

Ключові слова: WEB-ДОДАТОК, РОЗРОБКА, СИСТЕМА, ЗАДОВОЛЕНІСТЬ СПІВРОБІТНИКІВ

Об'єкт дослідження: створення опитувань задоволеності співробітників

Предмет дослідження: розробка Web-додатку для стоврення опитувань

Мета роботи: створити Web-додаток, здатний конструювати опитування зі різноманітними типами питань та з подальшою можливістю розсилання.

Методи дослідження: аналіз програмного забезпечення для створення і проектування Web-додатку

Отримані результати: реалізовано і введено в експлуатацію Web-додаток, який дозволяє зручно створювати опитування та розсилати їх.

Результати дипломної роботи планується використовувати для подальшого розвитку, додання та розширеня функціоналу.

.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНВ…….7

ВСТУП……...8

РОЗДІЛ 1. Аналіз предметної області……...11

Поняття «Web-додаток» та «Web-сайт»……..11

Основні відмінності web- додатку від web-сайту……...12

Інтерактивність……..12

Інтеграція……...12

Авторизація……13

Переваги та недоліки Web-додатку……..14

Переваги створення Web-додатків……..14

Недоліки створення Web-додатків……..15

Архітектура та принципи роботи типового web-додатку……..16

ВИСНОВОК ДО РОЗДІЛУ 1…….22

РОЗДІЛ 2. Аналіз та порівняння обраних інструментів і технологій для web-додатку…….23

HTML……..23

CSS……...26

JavaScript…….28

jQuery……...30

PHP……..31

Фреймворки у веб-розробці……..38

Поняття фреймворку, веб-фреймворку……...38

Важливість фреймоврків у розробці……...39

Топ-10 фреймворків PHP……..41

Фреймворк Yii……...48

IDE та текстові редактори……49

Open Server……..52
ВИСНОВОК ДО РОЗДІЛУ 2…….56
РОЗДІЛ 3. Створення web-додатку «система контролю задоволеності співробітників»……...57
Постановка задачі……..57
Підготовка до розробки додатку……..58
Розгортання та налаштування фреймворку Yii2 на проекті……..61
Проектування БД……67
Опис готового проекту……..71
ВИСНОВОК ДО РОЗДІЛУ 3…….80
ВИСНОВКИ…….81
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……...82
Додаток А…….84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ООП – об’єктно-орієнтоване програмування;

HTML – HyperText Markup Language;

CSS – Cascade Style Sheets;

HTTP – HyperText Transfer Protocol;

JS – JavaScript;

HTTP – протокол передачі даних;

ERP - enterprise resource planning;

CRM - Customer Relationship Management;

URL - Uniform Resource Locator;

DNS - Domain Name System;

IP - Internet Protocol;

TCP - Transmission Control Protocol;

IDE – Інтегроване середовище розробки (Integrated Development Environment);

PHP - Personal Home Page;

Yii – yes it is;

ВСТУП

Одним з показників, який достовірно відображає рівень конкурентоспроможності підприємств і організацій є продуктивність праці. Сукупність факторів, що впливають на продуктивність умовно можна розділити на два рівня. Фактори рівня господарюючого суб'єкта: рівень використання кадрового потенціалу, рівень техніки і технологій, наявність сегмента ринку, якість менеджменту. До зовнішніх факторів по відношенню до суб'єкта господарювання, можна віднести: зміна політичної, соціально-економічної ситуації, особливості податково-бюджетної, грошово-кредитної та інвестиційної політики держави, форс-мажорні обставини на фінансовому ринку та ін. Значення факторів цих рівнів в збільшенні продуктивності праці, безсумнівно велике.

Однак, в останні роки, співвідношення факторів, що обумовлюють зростання продуктивності праці, істотно змінилося в бік підвищення значення людських ресурсів в досягненні ефективності виробничої діяльності. Ефективне управління кадрами, спрямоване на досягнення цілей організації, має на увазі вміле використання праці, досвіду, інтелекту і мотивів поведінки людей з урахуванням їх задоволеності працею.

Висока задоволеність роботою характеризується переважанням позитивного, конструктивного ставлення до роботи з боку співробітників, яке проявляється в старанності, високого ступеня відповідальності за виконувану роботу, прагненні зробити її якомога краще, дотримання норм поведінки і вимог організації, співробітництво, взаємодопомогу, бажанні підвищити свій професійний рівень та ін.

Незадоволеність роботою проявляється в низьких результатах праці, великій текучки персоналу, його нестабільності, високому рівні захворюваності, порушення норм поведінки, в тому числі правил техніки безпеки, зниженні трудової активності і т.д.

Прийнято вважати, що поповнити персонал своєї компанії якісними, кваліфікованими працівниками – велика вдача, але це всього лише мала дещиця необхідних зусиль. Потрібно як слід постаратися, щоб такі фахівці надовго залишалися на підприємстві. Для досягнення цієї мети дуже важливо зрозуміти, наскільки персонал організації лояльний до існуючих в компанії умов праці, наскільки співробітники задоволені своєю зарплатою, соцпакетом і атмосферою відносин на підприємстві. Якщо результати оцінки показують високий індекс задоволеності працівників, то для підприємства забезпечено і стабільний фінансовий результат, і низький рівень стресу робітників, що також позитивно позначається на бізнес-процесах всієї організації. Оцінка задоволеності персоналу надає в розпорядження роботодавця один з головних показників, який характеризує ефективність роботи всього підприємства.

Для отримання оцінки задоволеності менеджери, та керівники організацій проводять дослідження задоволеності працею. Дослідження можна поділити на п’ять етапів:

1. Визначення мети дослідження.
2. Складання листів опитування.
3. Проведення опитування співробітників, заповнення анкет.
4. Визначення результатів дослідження. Аналіз рентабельності діючої системи мотивації.
5. Інформування керівництва та персоналу про результати дослідження. Висновки та рекомендації.

Перша вимога для ефективного опитування - конфіденційність. Якщо у співробітника виникне найменший сумнів у безпеці своєї участі, можна не розраховувати на отримання достовірних результатів. Друга вимога - технічна можливість швидко і без помилок обробити отримані дані. Оптимальний варіант організації опитування - за допомогою інтернет-технологій. Заповнення і аналіз паперових бланків потребує значного обсягу часу.

Метою написання кваліфікаційної роботи магістра є створення системи

контролю задоволеності співробітників, тобто web-додатку для конструювання опитувань, їх розсилання та обробку отриманих даних.

Об’єктом дослідження є створення опитувань задоволеності працівників.

Предметом дослідження є створення web-додатку.

Для створення серверної частини web-додатку було використано фреймворк Yii2, який написано мовою програмування PHP. Для клієнтської частини використовувались: мова розмітки HTML, мова опису зовнішнього виду документа, CSS, а також мова програмування JavaScript, зокрема бібліотека jQuery. Для роботи з базою даних додаток буде використовувати систему керування базами даних MySQL.

Веб-додаток складатиметься с адмін-панелі в якій можна конструювати опитування, налаштовувати розсилання опитувань і дивитися статистику (для менеджерів чи керівників) та з перегляду\проходження сконструйованих опитувань (для працівників).

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття «Web-додаток» та «Web-сайт»

Веб-додаток - це будь-яка комп'ютерна програма, яка виконує певну функцію, використовуючи веб-браузер у якості свого клієнта, в незалежності від пристрою та платформи, на яких відкрито браузер. Додаток може бути таким же простим, як дошка оголошень або контактна форма на веб-сайті, або настільки ж складний, як текстовий процесор або багатокористувацький мобільний ігровий додаток, який ви завантажуєте на свій телефон.

Веб-сайт - це група глобально доступних взаємопов'язаних веб-сторінок, які мають єдине доменне ім’я. Він може бути розроблений і підтримуватися особою, бізнесом або організацією. Веб-сайт має на меті служити різноманітним цілям. Приклад: сайти-візитки, блоги, новостні портали та ін. Основні функції веб-сайту:

1. Ефективний спосіб демонструвати свої продукти та послуги.
2. Розробка сайту допоможе створити соціальний доказ.
3. Допомагає у брендінгу бізнесу.
4. Допомагає досягти ділових цілей.
5. Дозволяє розширити підтримку клієнтів

1.2. Основні відмінності web- додатку від web-сайту

1.2.1. Інтерактивність

Перша відмінність - це різні степені взаємодії із сторінкою. У той час, як веб-сайт містить тексти та візуальний контент, з яким користувач не може взаємодіяти, веб-додатки надають користувачеві можливість не лише читати, а й змінювати, додавати інформацію на сторінках.

Інтернет-магазин, який дозволяє користувачам купувати товари, шукати їх у каталозі, можна назвати веб-додатком. Ще один цікавий приклад - це соціальні мережі. Вони включають у себе функції блогу, чати, контент на вибір користувача та можливість ділитися цим контентом.

Сьогодні велика кількість веб-сайтів мають інтерактивність. Тому що користувачеві це подобається. Для цього власники веб-сайтів додають на свої веб-сайти невеликі веб-програми.

На сайті деяких ресторанів є доступ до Google Maps, який допомагає користувачеві знайти дорогу до ресторану. У той ж час, більшість сайтів більш інформативні ніж інтерактивні. Таким чином відвідувачі сайтів більш захоплені переглядом, читанням або прослухуванням інформаціїї. А відвідувачі веб-додатку направлені на взаємодію з користувачами.

1.2.2. Інтеграція

Інтеграція - це процес, що містить об'єднання простих компонентів у одному складному. Розробники можуть інтегрувати веб-додатки та сайти з програмами, включаючи ERP, CRM. Однак у більшості випадків інтеграція відбувається саме з веб-додатками, тому що їх складним функціям дуже часто потрібна інформація з інших систем. Найпопулярніший вид інтеграції в

електронній комерції - це інтеграція веб-додатків із системою управління взаємодією з клієнтами (CRM). Це допомагає зберігати дані про покупців, інформацію про замовлення та покращити продажі в цілому. Завдяки інтеграції, інформація про користувачів, автоматично збирається та зберігається в CRM системі. Така інтеграція може дозволити команді відділу продажів дізнатися більше про поведінку клієнтів і ефективніше працювати з негативними відгуками. Доступна можливість взаємодії з інформацією про клієнта може принести збільшення в продажі та вдосконалити процес роботи в Інтернет-магазині.

У деяких випадках власники веб-сайтів використовують інтеграцію з CRM для того, щоб представити відвідувачам більше персоналізованого контенту. Але, на відміну від веб-додатків, подібна інтеграція з веб-сайтами - це скоріше опціональна функція, а не частина основного функціоналу.

1.2.3. Авторизація

Цей процес включає в себе введення користувальницьких даних для отримання доступу до веб-сайтів або систем. Ця функція важлива для систем, які вимагають особисту інформацію про клієнтів. На цьому етапі важливо приділити особливу увагу безпеці. Дуже важливо мінімізувати вірогідність доступу до особистих данних користувача постороннім людям.

На відміну від від сайтів, веб-додатки частіше вимагають авторизації, тому що вони пропонують користувачам більше можливостей, ніж веб-сайти. Наприклад, при використанні соціальних мереж, системи попереджують вас про слабкість пароля. Ігнорування подібних повідомлень може привести до того, що хакери можуть отримати доступ до вашого аккаунту.

Більшість інформаційних сайтів використовують авторизацію. У деяких випадках авторизація використовується для того, щоб дати більше можливостей, які недоступні неавторизованим користувачем. Якщо незареєстровані

користувачі можуть лише переглядати статті, зареєстровані користувачі можуть залишити коментарі, ділитися статтями у соціальних мережах та ін. Це також відмінне рішення для блокування спаму. Таким чином, авторизація необхідна як для веб-сайтів, так і для веб-додатків. Але в цей ж час авторизація потрібна веб-додаткам в цілях безпеки інформації.

У сучасному світі інтернету майже не залишилось сайтів, які б не мали ніяких інтерактивних елементів. І навпаки, багато веб-додатків часто включають у себе пошук інформації. Тим не менш, веб-сайти як і раніш являються інформаційними джерелами, а веб-додатки залишаються користувальницькими інструментами.

Можна сказати що всі Web-додатки є Web-сайтами, та не всі Web-сайти є Web-додатками. Те що робить їх такими унікальними, так це той факт, що неможливо розробити Web-додаток з чистого листа самостійно. У будь-якому разі доведеться використовувати не тільки звичні мови розмітки та програмування, як HTML, CSS та JavaScript, але й цілі бібліотеки технологій від сторонніх розробників. Є велика кількість компаній, які сконцентровані на створені подібних технологій для стартапів і малого бізнесу.

1.3. Переваги та недоліки Web-додатку

1.3.1. Переваги створення Web-додатків

Веб-додатки розробляються з такими мовами програмування, як HTML та CSS, які добре відомі серед IT-фахівців.

На відміну від нативних додатків, веб-додаток може використовуватись на усіх пристроях. Він запрограмований для роботи в будь-якій операційній системі. Він повинен належним чином адаптуватися до iOS, Android, Windows Phone та інших системах.

Веб-додатки не вимогливі до ресурсів і не пред’являють ніяких вимог до

апаратної платформи.

Немає проблем с підтримкою старих версій програм й з сумістністю. Коли з'являється нова версія нативного додатку, користувачам нерідко доводиться вирішувати проблеми, зв'язаних з оновленням вже встановленої на їх пристроях копії. У випадку с браузерним додатком таких проблем не буває – існує лише одна версія, в якій працюють усі користувачі і у випадку виходу нової, всі без виключень переходять на неї, іноді навіть не замічаючи цього.

Веб-додаток не потребує встановлення на пристрій.

Ці програми працюють у власному веб-браузері пристрою через просту URL-адресу.

Їх не потрібно завантажувати та встановлювати з магазинів додатків, таких як Google Play або Apple Store App. Це означає економію грошей, оскільки пряме посилання через веб-додаток є безкоштовним.

Вони також можуть відкривати веб-сайти. Це означає, що вони не потребують оновлення, як це роблять звичайні програми.

Високий рівень розвитку надійності сітьових з'єднань та web-технологій.

Розробка веб-додатків - дешевший вид розробки додатків. Він складається із створення посилання або декількох посилань між додатком та URL-адресою. Розробка нативного або інтерпретованого додатка тягне за собою більш високу вартість, але шанси на успіх додатку набагато більше.

Web-додатки дозволяють своїм користувачам бути насправді мобільними. Ви можете зберігати результат своєї роботи на сервері, і у випадку необхідності мати доступ до них із будь якої точки планети, де є вихід у мережу Інтернет.

1.3.2. Недоліки створення Web-додатків

Як було сказано вище, один веб-додаток може використовуватись на всіх пристрях. Але, звичайно ж, веб-сайт повинен бути запрограмований таким чином, щоб він відображався незалежно від операційної системи пристрою.

Якщо це не адаптивний веб-сайт, у вас можуть виникнути проблеми під час його відображення на iOS, Android або Windows Phone.

Підключення до Інтернету буде абсолютно необхідним для його запуску. В іншому випадку ви не зможете переглядати веб-сайт, і веб-додаток вам не принесе користі. На превеликий жаль, наприклад, у нашій країні, інтернет є не всюди. Що ускладнює роботу з web-додатками, наприклад, під час подорожі.

Кожне перезавантаження (або оновлення сторінки) викликає помітну затримку, викликану необхідністю встановити HTTP- з'єднання, обробити запит на сервері, передати по мережі у відповідь HTTP-повідомлення і перезавантажити сторінку браузером. Це створює переривчастий режим роботи та уповільнює його.

Крім того, існують деякі обмеження доступу щодо певних апаратних функцій пристрою, на якому він працює.

Існує багато велика кількість додатків, які не можуть бути реалізовані в Web, наприклад в барузері неможливо створювати складні трьохмірні моделі.

1.4 Архітектура та принципи роботи типового web-додатку.

Web-додаток являється клієнт-серверним, в я кому клієнтом виступає

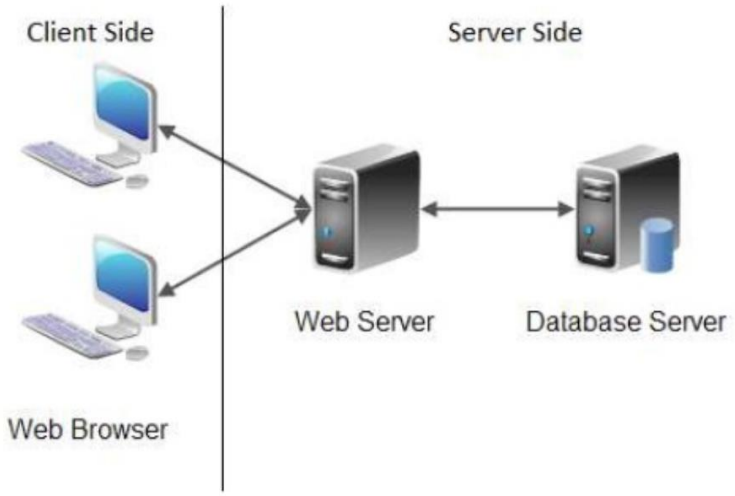

Рис. 1.1 Діаграма типового web-додатку

браузер, а сервером web-сервер. Логіка web-додатку розподілена між сервером та клієнтом. Збереження даних переважно здійснюється на сервері. Обмін інформації проходить у мережі Інтерент.

В web-додатках існує дві програми, працюючі одночасно: код, який знаходиться на сервері і відповідає на HTTP запити; код, який знаходиться в браузері і реагує на введення даних користувачем.

Серверний код не видимий для клієнта, може відповідати тільки на HTTP запити конкретного URL, а не на будь-який тип користувацького введення. Зазвичай цей код написаний на наступних язиках програмування та фреймворках: JavaScript(Node.js), Python(Django), PHP, Ruby On Rails, Java, C# та інші.

Клієнтський код розбирається браузером користувача. На відміну від серверного він може реагувати на користувацькі введення, доступний для перегляду та редагування користувачеві у повному обсязі. Не може читати файли сервера напряму, повинен звертатися з сервером через HTTP запити. Мови програмування: HTML, CSS, JavaScript.

Технологічний стек представляє собою: операційну систему (файлову систему). Частіше всього це Linux; web-сервер (Apache, NGINX, IIS та ін.); база

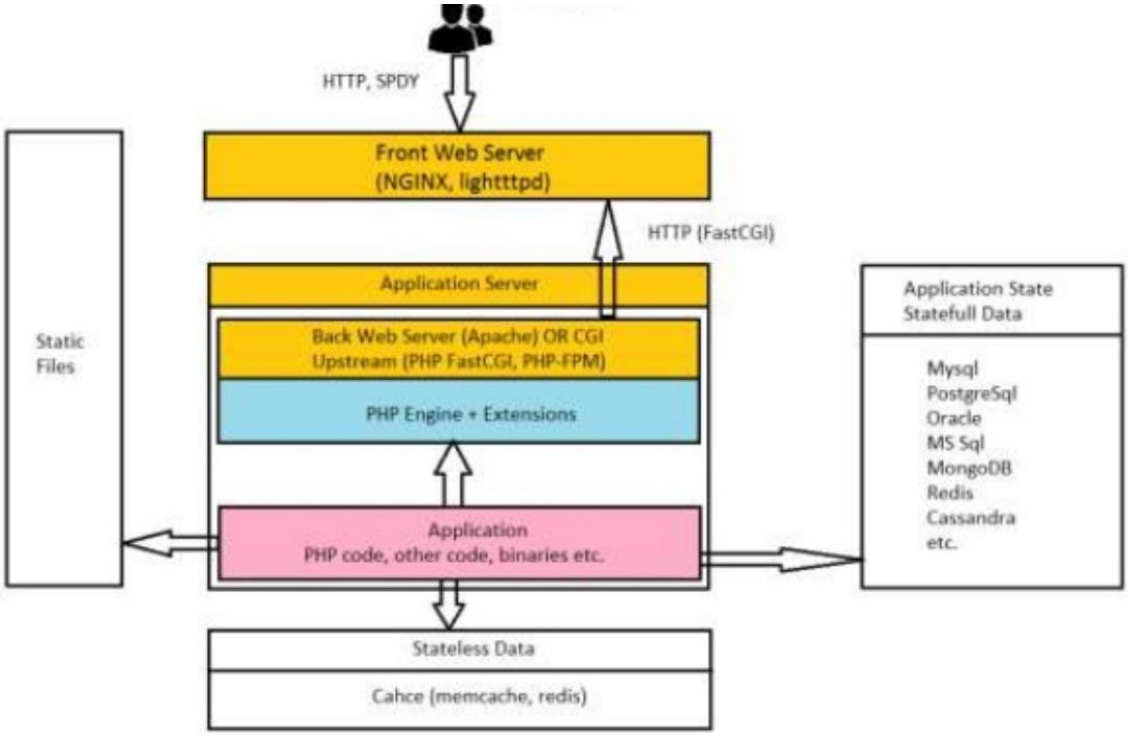

Рис. 1.2. Технологічний стек

данних (MySql, MongoDB, MS Sql, Oracle, PostgreSql, Redis та ін.)

Шлях запиту:

1. DNS.
2. HTTP, SPDY.
3. Front Server (NGINX).
4. Back Server or Application Server (Apache, PHP-FPM).
5. PHP code, opcode.
6. Response.

DNS (Domain Name System) - ієрархічна розподілена система перетворення імені хоста (комп'ютера або іншого мережевого пристрою) в IP-адресу. Всі комп'ютери, підключені до Інтернету, включаючи смартфони, настільні комп'ютери і сервери, що надають контент для величезних торгових веб-сайтів, знаходять один одного і обмінюються інформацією за допомогою цифр. Ці цифри називаються IP-адресами. Щоб відкрити веб-сайт в браузері, не потрібно запам'ятовувати довгі набори цифр. Досить ввести доменне ім'я, наприклад example.com, і браузер відкриє потрібну сторінку.

Шлях користувацького запиту починається з DNS:

1. Користувач звертається до DNS для отримки IP адреси.
2. DNS віддає IP адресу користувачеві.
3. Користувач звертається до web-додатка по IP через TCP/IP.
4. Відкривається мережеве з’єднання (HTTP, SPDY).

Стек протоколів TCP/IP – це набір мережевих протоколів, що забезпечують передачу даних, які використовуються в мережах, включаючи мережу Інтернет. Назва походить від двох найважніших протоколів: TCP (Transmission Control Protocol), IP (Internet Protocol).

IP – протокол, який лежить у основі Інтернета, його назва так і розшифровується: Internet Protocol. Наразі використовується дві версії протоколу IP:

1. IPv6 – порівняно нова (опублікована в 1998); IP-адреса має розрядність

128 біт і записується у виді восьми 16-ти бітних полів, з використанням шістнадцяткової системи числення і з можливістю зкороченням двох чи більше послідовних нульових полів до «::». Приклад: 2001:db8:1234:5::1:1.

2. IPv4 – опублікована в 1981р; IP- адреса має розрядність 32 біта і записується у виді чотирьох десятичних чисел в діапазоні 0…255 через точку; приклад: 192.0.3.43.

Кожен вузол може напряму з'єднуватись лише з вузлами своєї мережі (наприклад, підключенними к одному і тому ж сегменту Інтеренту), для визначення яких використовується адреса мережі – частина IP-адреси, обумовлена маскою мережі. Зв'язок з вузлами інших мереж здійснюється через допоміжні вузли – маршрутизатори.

TCP – це протокол, який базується на IP для доставки пакетів, але добавляє дві важливі речі: встановлення з'єднання – це дозволяє йому, на відміну від IP, гарантувати доставку пакетів; порти – дозволяють обмінюватись пакетами між застосунками, а не просто вузлами. Протокол TCP призначений для обміну даними – це надійний протокол, тому що:

1. Забезпечує надійну доставку даних, так як передбачає встановлення логічного з'єднання.
2. Нумерує пакети і підтверджує їх прийом квитанцією, а у випадку загублення пакету організує повторну передачу.
3. Ділить переданий потік байтів на частини – сегменти, і передає їх нижньому рівню, на прийомній стороні знову збирає їх у потік байтів.

З'єднання двух вузлів починається з handshake (рукопотискання):

1. Вузол А посилає вузлу В спеціальний пакет SYN – запрошення до з'єднання.
2. В відповідає пакетом SYN-ACK – згодою на установлення з'єднання.
3. А посилає пакет ACK – підтвердження, що згоду отримано.

Після цього TCP з'єднання рахується установленим, і додатки, працюючі на цих

вузлах, можуть відправляти один одному пакети з даними. З'єднання означає, що вузли пям'ятають один одного, нумерують усі пакети, які ідуть в обидві сторони, посилають підтвердження про отримання кожного пакету і знову посилають загублені по дорозі пакети. Для вузла А це з'єднання називається вихіднім, а для вузла В – вхідним.

HTTP (Hyper Text Transfer Protocol) – протокол прикладного рівня передачі даних (початково – у виді гіпертекстових документів в форматі «HTML», наразі використовується для передачі довільних даних).

SPDY (читається як «speedy», «спіді») – протокол прикладного рівня для передачі веб-контенту. Протокол розроблено корпорацією Google. По задуму розробників, даний протокол позиціонується як заміна деяких частин протоколу HTTP – таких, як управління з'єднаннями і форматами передачі даних.

Клієнт спілкується з сервером через HTTP-повідомлення. Обмін повідомленнями йде по звичайній схемі «запит-відповідь».

Кожне HTTP-повідомлення складається із чотирьох частин, які передаються у вказанному порядку:

1. Стартовий рядок (англ. Starting line) – оприділяє тип повідомлення;
2. Заголовки (англ. Headers) – характеризує тіло повідомлення, параметри передачі та інші дані.
3. Тіло повідомлення (англ. Message Body) – безпосередньо дані повідомлення. Обов'язково має відділятись від заголовків пустим рядком.

Заголовки і тіло повідомлення можуть бути відсутніми, але стартовий рядок являється обов'язковим елементом, так як вказує на тип запиту\відповіді. Виключенням є версія 0.9 протоколу, у якого повідомлення запиту містить тільки стартовий рядок, а повідомлення відповіді тільки тіло повідомлення.

Стартові рядки розрізняються для запиту і для відповіді. Рядок запиту виглядає так:

GET URI – для версії 0.9;

Метод URI HTTP/Версія – для інших версій.

Тут:

Метод – назва запиту, одне слово заглавними буквами. У версії HTTP 0.9 використовується тільки метод GET;

URI оприділяє шлях до запитаного документу;

Версія – пара розділених точкою цифр. Наприклад: 1.0.

Стартова строка відповіді має наступний формат: HTTP/Версія, код стану, пояснення, де:

Версія – пара розділених точкою цифр, як в запиті;

Код стану – три цифри. По коду стану оприділяється подальший вміст повідомлення і поведінка клієнта;

Пояснення – текстове коротке пояснення до коду відповіді для користувача. Ніяк не впливає на повідомлення і являється необов'язковим.

Методи HTTP – послідовність із будь-яких символів, окрім управляючих і роздільників, яка указує на основну операцію над ресурсом. Кожний сервер забов'язаний підтримувати як мінімум методи GET і HEAD. Якщо сервер не розпізнав вказанний клієнтом метод, то він має вернути статус 501 (Not implementer). Якщо серверу метод відомий, але він не може його примінити до конкретного ресурсу, то повертається повідомлення з кодом 405 (Method Not Allowed). Окрім методів GET і HEAD, дуже часто використовується метод POST.

ВИСНОВОК ДО РОЗДІЛУ 1

Можна зробити висновки щодо особливостей розробки Web-додатку. Саме завдяки подібним проектам можна легко позбавитись будь-яких проблем, пов’язаних із представленням інформації сучасному користувачу. Розробка Web-додатку займає не так багато часу і ресурсів, щоб віддавати перевагу звичайним сайтам. Така розробка може бути відкрита на будь-якому пристрої, тому база потенційних користувачів автоматично збільшується. Можна забути про проблеми з оновленням програмного забезпечення, релізом нових версій сервісу. Тому що Web-додаток можна легко покращувати, не докладаючи до цього багато зусиль.

Гнучкість технологій розробки дозволяє створити Web-додаток для будь-якої сфери діяльності.

РОЗДІЛ 2
АНАЛІЗ ТА ПОРІВНЯННЯ ОБРАНИХ ІНСТРУМЕНТІВ І ТЕХНОЛОГІЙ ДЛЯ WEB-ДОДАТКУ

2.1. HTML

HTML або "Мова розмітки гіпертексту" – це код, який використовується для створення веб-сторінок та веб-додатків, доступ до яких здійснюється в Інтернеті. HTML був розроблений на зорі всесвітньої павутини наприкінці 1980-х - початку 1990-х. Мова дозволяє розробникам веб-сайтів розповісти інтернет-браузеру, як відображати елементи, такі як зображення, текст, форми та інтерактивні функції.

Творці веб-сайтів регулярно використовують HTML разом із каскадними таблицями стилів (CSS) та JavaScript, що є двома іншими мовами програмування, для створення сайтів та програм, до яких користувачі можуть отримати доступ, використовуючи Інтернет-браузерні програми, такі як Chrome, Firefox, Safari та Edge. Стандарти для HTML, а також CSS історично підтримували консорціум всесвітньої веб-сторінки (W3C).

Саме в 1980 році Тім Бернерс-Лі, працюючи в Європейській організації ядерних досліджень, почав працювати над прототипом HTML. До кінця десятиліття Бернерс-Лі створив всюдисущу мову, а також Інтернет-браузер та серверне програмне забезпечення.

Протягом наступних двох десятиліть Всесвітній консорціум із веб-сторінок (W3C) продовжить розробку за міжнародним стандартом коду. У 1995 році була опублікована друга версія HTML, і протягом наступних кількох років наступні декілька додаткових оновлень.

В той час як HTML побудовано на основі правил SGML (стандартна узагальнена мова розмітки, це деяка метамова на якій можна визначати мову розмітки для документів), XHTML побудовано на основі правил XML, суворішої підмножини правил SGML. Оскільки XHTML документи мають бути коректними XML документами, їх обробку можна здійснювати стандартними інструментами обробки XML документів на відміну від HTML, який вимагає порівняно складніших, важчих і повільніших синтаксичних аналізаторів. XHTML можна розглядати як, багато в чому, перетин HTML і XML, оскільки цей стандарт є переформулюванням HTML засобами XML. XHTML 1.0 став рекомендацією консорціуму W3C 26 січня 2000 року. XHTML 1.1 став рекомендацією W3C 31 травня 2001 року.

Мова HTML інтерпретується браузером і відображається у вигляді документа, зручному для людини.

HTML є додатком SGML (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту ISO 8879.

HTML-документ є текстовим файлом розмічений за допомогою спеціальних (природно, текстових) команд. Текстовий формат представлення веб-документів був вибраний виходячи з основних вимог до веб-документу: простота, можливість безпосередньої інтерпретації в будь-якій операційній системі, мінімальний розмір файлу, зручність редагування і інтерпретації.

Мова розмітки гіпертекстових документів HTML дозволяє визначити різні типи елементів, що забезпечують функціональність документа: текстові фрагменти із заданими параметрами форматування, списки, таблиці, зображення, гіперпосилання і т.д. Елементи HTML оголошуються за допомогою команд розмітки, званих тегами (від англійського tag – ярлик). Усі HTML-теги,

що зустрічаються в тексті документа інтерпретуються браузером при відображенні документа.

HTML - мова розмітки, це фраза, яка використовується для опису того, як програміст може використовувати код для позначення тексту. Мови розмітки, такі як HTML та XML, читаються людьми та відрізняються від машинних мов, які записуються шістнадцятковим чи двійковим кодом.

Використовуючи HTML для створення сторінки на веб-сайті, автор використовуватиме декілька ключових компонентів або атрибутів, які зможе прочитати будь-який Інтернет-браузер, що отримує доступ до веб-сайту. Багато з цих атрибутів будуть в поєднанні зі «початковим тегом» та «кінцевим тегом».

Наприклад, автор може використовувати літеру "P" з набором дужок для позначення початку абзацу та другого "P" у другому наборі дужок для позначення кінця абзацу. Перший фрагмент коду відкриває абзац, а другий фрагмент коду закриває абзац.

```
<p>Текст</p>
```

Рис. 2.1 Приклад використання

HTML 5 є найбільш актуальним на сьогоднішній день стандартом цієї мови розмітки. Саме на цьому стандарті базується найбільша кількість Web-сторінок у інтернеті. В останній версії для HTML, що представляє собою HTML 5, ми отримуємо підтримку мультимедіа (аудіо та відео). Сьогодні основним складовим елементом веб-сторінок є HTML 5. Він надав статичний вигляд веб-сторінок. А всі інші стилі та функції додаються за допомогою CSS, PHP, Javascript. Тому перш ніж вивчати ці мови, ви повинні вивчити HTML.

Можна сказати, що HTML – це основний будівельний блок Web-розробки. Всі сучасні Web-сайти, сервіси створення Web-сайтів, такі як WordPress, Wix, Weebly використовують HTML 5 для створення Web-сайту.

TML-елемент виділяється з іншого тексту в документі за допомогою "тегів", які складаються з імені елемента оточеного "<" і ">". Ім'я елемента всередині тегу не чутливе до регістру. Тобто, воно може бути написано в верхньому або нижньому регістрі, або змішано. Наприклад, тег <title> може бути записаний як <Title>, <TITLE>, або будь-яким іншим способом.

Програміст веб-сайту може використовувати атрибути та елементи HTML для визначення практично кожної грані сторінки та того, як вона буде відображатися користувачеві. Програміст може використовувати CSS та JavaScript, а також додаткові мови, такі як PHP, jQuery та XML для створення веб-сайту.

Новачки-програмісти часто вивчать HTML як свою першу мову програмування та продовжать навчання з передовими уроками JavaScript та CSS.

Вивчення основ HTML може виявитись корисним для майбутніх програмістів, а також тих, хто бажає зрозуміти, що потрібно для створення веб-сторінок, які він чи вона читає щодня на комп’ютері чи смартфоні.

2.2. CSS

CSS - це мова каскадних таблиць стилів (англ. cascading style sheets) , яка використовується для опису стилів багаторазового використання для подання документів, написаних мовою розмітки. Його концепція була створена компанією Håkon Wium Lie у 1994 році. У грудні 1996 року WSC розробила специфікацію для CSS і сьогодні дозволяє веб-розробникам змінювати макет і зовнішній вигляд своїх веб-сторінок. Наприклад, CSS може використовуватися для зміни шрифту, який використовується в певному HTML-елементі, а також його розміру та кольору. Один CSS-файл може бути пов'язаний з декількома сторінками, що дозволяє розробнику одночасно змінювати зовнішній вигляд усіх сторінок. Наведене нижче поле містить основний приклад використання коду CSS для визначення шрифтів, кольору гіперпосилань та кольору

посилання, коли курсор миші наводить курсор миші. У цьому конкретному

```
body {
 font: normal 100% "trebuchet ms", Arial, Helvetica, sans-serif;
}
a {
 color: #000000;
}
A:visited {
 color: #005177;
}
a:hover {
 color: #005177;
}
```

Рис. 2.2 Поле з CSS кодом

прикладі ми змінюємо лише теги HTML <a> та <body>, а не створюємо нові селектори класу чи id.

Переваги CSS:

1. CSS економить час - ви можете написати CSS один раз, а потім повторно використовувати той же аркуш на декількох HTML-сторінках. Ви можете визначити стиль для кожного елемента HTML і застосувати його до якомога більше веб-сторінок.
2. Сторінки завантажуються швидше - якщо ви використовуєте CSS, вам не потрібно щоразу писати атрибути тегів HTML. Просто напишіть одне правило CSS тегу та застосуйте його до всіх випадків цього тегу. Так менше коду означає швидше завантаження.
3. Простота обслуговування - Щоб зробити глобальну зміну, просто змініть стиль, і всі елементи на всіх веб-сторінках будуть оновлені автоматично.
4. Покращені стилі до HTML - CSS має набагато ширший набір атрибутів, ніж HTML, тому ви можете надати набагато кращий погляд на свою сторінку HTML порівняно з атрибутами HTML.
5. Сумісність декількох пристроїв - аркуші стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв.

Використовуючи один і той же документ HTML, різні версії веб-сайту можуть бути представлені для портативних пристроїв, таких як КПК та мобільних телефонів або для друку.

Версії CSS:

Каскадні таблиці стилів рівня 1 (CSS1) вийшли з W3C як рекомендація в грудні 1996 року. Ця версія описує мову CSS, а також просту модель візуального форматування для всіх тегів HTML.

CSS2 став рекомендацією W3C у травні 1998 року та базується на CSS1. Ця версія додає підтримку для специфічних таблиць стилів, наприклад, принтери та аудіопристрої, завантажувані шрифти, розміщення елементів та таблиці.

2.3. JavaScript

JavaScript – скриптова мова, що найчастіше використовується при створенні сценаріїв поведінки браузера, що вбудовуються у веб-сторінки.

Назва «JavaScript» є зареєстрованою торговою маркою компанії Sun Microsystems, Inc.

JavaScript був створений, щоб "оживити веб-сторінки". Програми цією мовою називаються скриптами. Їх можна записати прямо в HTML веб-сторінки та запустити автоматично під час завантаження сторінки. Сценарії надаються та виконуються як звичайний текст. Для запуску їм не потрібна спеціальна підготовка чи компіляція. У цьому аспекті JavaScript сильно відрізняється від іншої мови під назвою Java.

Коли був створений JavaScript, він спочатку мав іншу назву: "LiveScript". Але Java була дуже популярною в той час, тому було вирішено, що позиціонування нової мови як "молодшого брата" Java допоможе.

Але в міру розвитку, JavaScript став цілком незалежною мовою зі своєю специфікацією під назвою ECMAScript, і тепер він взагалі не має відношення до Java.

JavaScript найчастіше використовується в якості мови сценаріїв на стороні клієнта. Це означає, що код JavaScript записується на сторінку HTML. Коли користувач вимагає HTML-сторінки з JavaScript у ній, скрипт надсилається до браузера, і браузер повинен вирішувати щось робити.

Те, що сценарій знаходиться на сторінці HTML, означає, що ваші сценарії можуть бачити та копіювати всі, хто переглядає вашу сторінку. Тим не менш, на мій погляд, ця відкритість є великою перевагою, адже зворотний бік полягає в тому, що ви можете переглядати, вивчати та використовувати будь-який JavaScript, який ви зустрінете на WWW.

JavaScript може використовуватися в інших контекстах, ніж веб-браузер. Netscape створив JavaScript на стороні сервера як CGI-мову, яка може робити приблизно так само, як Perl або ASP. Немає жодної причини, через яку JavaScript не можна було б використовувати для написання реальних, складних програм. Однак цей сайт стосується виключно використання JavaScript у веб-браузерах.

Сьогодні JavaScript може виконуватись не тільки в браузері, але і на сервері або фактично на будь-якому пристрої, який має спеціальну програму під назвою механізм JavaScript. У браузері є вбудований «движок», який іноді називають "віртуальною машиною JavaScript". У різних двигунах є різні "кодові назви". Наприклад:

V8 - в Chrome і Opera.

SpiderMonkey - у Firefox.

Сучасний JavaScript - це "безпечна" мова програмування. Він не забезпечує низькорівневий доступ до пам'яті чи процесора, тому що він був створений для браузерів, які цього не потребують.

Можливості JavaScript значною мірою залежать від середовища, в якому він працює. Наприклад, Node.js підтримує функції, які дозволяють JavaScript читати / записувати довільні файли, виконувати мережеві запити тощо.

В браузері JavaScript може робити все, що стосується маніпуляції веб-сторінками, взаємодії з користувачем та веб-сервером. Наприклад, JavaScript

може наступне у веб-браузері:

Додати новий HTML на сторінку, змінити існуючий вміст, змінити стилі.

Реагувати на дії користувача(натискання миші, рух вказівника, натискання клавіш).

Надсилати запити по мережі на віддалених серверів, зкачувати та завантажувати файли (так звані технології AJAX та COMET).

Отримати та встановити файли cookie, задавати питання відвідувачеві, показувати повідомлення.

Запам’ятовувати дані на стороні клієнта («локальне зберігання»).

2.4 jQuery

Простіше кажучи, jQuery - це набір попередньо написаних кодів JavaScript (відомий як бібліотека JavaScript), який ви можете додати до власних проектів.

Переваги jQuery:

1. jQuery робить програмування JavaScript більш швидким та ефективним.
2. jQuery є відкритим кодом (тобто кожен може внести свій внесок у або змінити) та має велике співтовариство користувачів, тобто його постійно підтримують та додають.
3. jQuery має велику документацію (включаючи приклади коду!)
4. jQuery добре працює з будь-якими іншими бібліотеками JavaScript, якими ви можете користуватися.
5. jQuery має дуже багато плагінів, які дозволяють розширити функціональність jQuery, коли це необхідно на проекті.

Оскільки jQuery корисний для спрощення окремих функцій (наприклад, вище), його можна розширити ще більше у вигляді плагінів - колекцій коду JS з бібліотеки jQuery, які поєднують ці окремі функції та створюють надійні функції веб-сайту та інструменти (знову ж таки, не кодуючи їх з нуля).

Плагіни створюються користувачами jQuery на основі коду в бібліотеці

jQuery, а потім можуть публічно публікуватися в Інтернеті. Хоч і плагіни можна знайти в багатьох місцях, ті, які знайдені в офіційному сховищі інтерфейсу jQuery (User Interface), можна сміливо вважати якісною роботою, оскільки їх курирує професійна спільнота jQuery.

2.5 PHP

PHP («препроцесор гіпертексту») – скриптова мова програмування, яка була створена для генерації сторінок мовою HTML на стороні веб-сервера і також для роботи з базами даних. В даний час підтримується переважною більшістю провайдерів хостингу.

Препроцесор — програма, яка виконує попередню обробку даних, для того, щоб вони могли використовуватись іншою програмою, наприклад, такою як компілятор. Про дані на виході препроцесора говорять, що вони знаходяться в препроцесованій формі, якій властива обробка подальшими програмами (компілятор). Результат і вид обробки залежать від класифікації препроцесора: так, деякі з них можуть тільки виконати просту текстову підстановку, а інші здатні змагатися з іншими скриптовими мовами програмування. Найчастіший випадок використання препроцесора – обробка первинного коду перед його відправленням на наступний крок компіляції. Мови програмування C/C++ і комп'ютерної верстки використовують препроцесори, що значно розширяють їхні можливості. Назва широко поширеної скриптової мови програмування PHP є рекурсивним акронімом «PHP: Hypertext Preprocessor».

В області програмування для мережі Internet, PHP – одна з найпопулярних скриптових мов (разом з JSP, Perl, Ruby, Python і мовами, використовуваними в ASP.NET) завдяки своїй простоті, швидкості виконання, багатій функціональності і розповсюдженню початкових кодів на основі ліцензії PHP. PHP відрізняється наявністю ядра і модулів, що підключаються, «розширень»: для роботи з базами даних, сокетами, графікою, криптографією, документами

формату PDF і т.д. Будь-який охочий може розробити своє власне розширення і підключити його. Існують сотні розширень, проте в стандартне постачання входить лише декілька десятків тих, що добре зарекомендували себе. Інтерпретатор PHP підключається до веб-серверу або через модуль, створений спеціально для цього сервера (наприклад, для Apache або IIS), або як CGI-додаток.

Окрім цього, він може використовуватися для вирішення адміністративних завдань в операційних системах UNIX, GNU/Linux, Microsoft Windows, Mac OS X і AmigaOS. Проте в такій якості він не набув поширення, віддаючи пальму першості Perl, Python і VBScript.

Синтаксис PHP подібний синтаксису мови Сі. Деякі елементи, такі як асоціативні масиви і цикл foreach, запозичені з Perl.

Сьогодні PHP знайшо використання серед тисячі програмістів. Кілька мільйонів сайтів повідомляють про роботу з PHP, що складає більше ніж п'ята частина доменів Інтернету.

Команда розробників мови PHP складається з безлічі людей, що добровільно працюють над ядром і розширеннями PHP, і суміжними проектами, такими, як PEAR або документація мови.

Назва PHP – рекурсивна абревіатура, що означає «PHP: Hypertext Preprocessor» (раніше акронім розшифровувався як «Personal Home Page Tools»). Спочатку PHP створювався як надбудова над Perl для полегшення розробки веб-сторінок.

PHP зазвичай використовується як серверна мова (на відміну від мови, як JavaScript, яка, як правило, виконується на стороні клієнта). У плані програмування на стороні клієнта розуміється діяльність на веб-сайті, яка відбувається локально на комп'ютері користувача через веб-браузер користувача. Мови на стороні клієнта, такі як HTML, CSS та JavaScript, дають можливість веб-браузерам розбирати та змінювати контент на екрані комп'ютера. Зауважте, що у цьому списку є JavaScript (мова сценаріїв на зразок

PHP). Знову ж, процеси, скриптовані JavaScript, відбуваються на стороні клієнта - JS надає інструкції, які можна зрозуміти та виконати у вашому веб-браузері. Сторона клієнта - це сторона, яку ви бачите під час використання Інтернету.

З іншого боку, серверна діяльність включає веб-браузер, який надсилає запити на веб-сервер (програмне забезпечення або обладнання, яке зберігає сторінки веб-сайтів, зображення, медіа та інші об’єкти), який потім відповідає на запит із HTML-кодом, який може обробляється та надається веб-браузером і перетворюється на вміст на екрані користувача. Основна відмінність тут від діяльності на стороні клієнта полягає в тому, що цей процес включає зв'язок із сервером і не завершений повністю в браузері клієнта. Іншими словами, мова сценаріїв на стороні клієнта, як JavaScript, може автоматизувати завдання, пов’язані з вмістом, який уже доступний користувачеві в його веб-браузері, але мова скриптів на стороні сервера, як PHP, використовується для запиту вмісту з сервера веб-сайту чи бази даних та створення що вміст видимий і доступний для користувача сайту. З метою підтримки ефективності веб-сайту, швидкості завантаження та ємності пам’яті не всі активи можна постійно зберігати локально на веб-сторінках.

Наприклад, PHP-скрипт може призвести до того, що три останні публікації блогу автоматично відображатимуться на головній сторінці вашого веб-сайту. У цьому випадку самі публікації зберігаються на сервері сайту та викликаються, коли вони займають один із трьох останніх опублікованих слотів. Це дозволяє уникнути як попереднього завантаження публікацій на ваш сайт, так і необхідності адміністратора сайту завантажувати та оновлювати публікації, коли публікуються нові історії. PHP-скрипти також можуть включати умовні (якщо / else / endif) заяви, які спрямовують ваш сайт змінювати його показ та додавати вміст з вашого веб-сервера за потреби. Сюди можна віднести такі дії, як диктування того, що якщо адміністратор сайту завантажить відеопосилання на поле «x», то сайт завантажить відео з його сервера та відобразить його для користувача. Сценарій може також вказати, що якщо адміністратор не

завантажує посилання, то на цій сторінці замість цього буде відображено зображення "y" за замовчуванням. Серверні дії PHP вводять цілком новий рівень динамічних можливостей на веб-сайт (вище і за його межами статичних функцій, що пропонуються HTML і CSS, і навіть динамічний вміст на стороні клієнта, що стає можливим завдяки JavaScript).

Функціональність PHP відіграє особливо велику роль, що стосується розробки WordPress. Хоча можна створити функціональні веб-сайти WordPress, не знаючи PHP, PHP - це двигун для спеціальних тем WordPress та плагінів. Створюючи такі види користувацьких функцій WordPress, PHP - це сценарна мова, яка використовується для спілкування з сервером сайту WordPress та доставки потрібного вмісту та дій на екран користувача. Якщо ви створюєте сайт WordPress за допомогою шаблонів тем і плагінів, ви все ще будете пасивно реалізовувати PHP, але для того, щоб створити власні теми, плагіни або змінити поведінку за замовчуванням WordPress, вам потрібно буде розвивати практичні навички PHP. WordPress - система управління контентом, яка дозволяє створювати та публікувати цифровий контент в Інтернеті - є особливо привабливою платформою для веб-розробки, оскільки її крива навчання дає місце для початківців (хто може покладатися на параметри меню та шаблони за замовчуванням), але він зберігає відкриті двері для просунутих користувачів, які можуть скористатися PHP та створити більш клієнтський досвід для клієнтів. Обізнавшись лише з основами WordPress, ви вже зможете зробити платну роботу розробника WordPress, але вивчення PHP зробить вас набагато більш кваліфікованим, щоб викреслити нішу як серйозний розробник WordPress.

Тут випливає, що, хоча PHP не випереджає HTML, CSS та JavaScript у списку необхідних навичок веб-розробників, це вміння, що кожен, хто бажає закріпити набір інструментів розробника, обов'язково повинен розглянути можливість додавання до свого арсеналу. Безумовно, можна створити статичні веб-сайти лише з HTML та CSS (і створити веб-сайти з динамічним вмістом за допомогою JavaScript), але введення мови на стороні сервера в сукупність різко

збільшить типи веб-сайтів, до яких можна звертатися, та клієнтів, на яких можна вийти. Зокрема, PHP також надасть вам доступ до прибуткового світу налаштування WordPress. І так - якщо мова йде про алфавітний суп мов кодування - PHP - це три букви, які ви обов'язково повинні мати на увазі.

PHP був написаний Расмусом Лердорфом на мові програмування C у 1994 році для використання у моніторингу його резюме в Інтернеті та пов'язаної з ним особистої інформації. З цієї причини PHP спочатку розшифровувався як "Персональна домашня сторінка". Лердорф поєднав PHP зі своїм власним інтерпретатором форм, випустивши комбінацію публічно як PHP / FI (зазвичай її називають PHP 2.0) 8 червня 1995 року. Два програмісти, Зеєв Сураскі та Анді Гутманс, відновили ядро PHP, випустивши оновлений результат як PHP / FI 2 в 1997 році. Акронім офіційно було змінено на PHP: HyperText Preprocessor, на цей час. (Це приклад рекурсивної абревіатури: там, де сама акронім є власним визначенням.) У 1998 році було випущено PHP 3, що було першою широко використовуваною версією. PHP 4 був випущений у травні 2000 року, з новим ядром, відомим як Zend Engine 1.0. PHP 4 відрізнявся покращеною швидкістю та надійністю порівняно з PHP 3. Що стосується функцій, PHP 4 додає посилання, тип булів, підтримка COM в Windows, вихід буферизації, багато нових функцій масиву, розширене об'єктно-орієнтоване програмування, включення бібліотеки PCRE, і більше. Версії технічного обслуговування PHP 4 все ще доступні, в першу чергу для оновлень безпеки.14 липня 2004 року було випущено PHP 5, що працює на базі нового Zend Engine II. PHP 5 включав нові функції, такі як покращена підтримка об'єктно-орієнтованого програмування, розширення PHP Data Objects (PDO) (яке визначає легкий і стійкий інтерфейс для доступу до баз даних) та численні поліпшення продуктивності. У 2008 році PHP 5 стала єдиною стабільною версією, що розробляється. Пізнє статичне зв'язування відсутнє у PHP та додано у версії 5.3.

З 5 лютого 2008 року багато гучних проектів з відкритим кодом перестали підтримувати PHP 4 у новому коді через ініціативу GoPHP5 , надану

консорціумом розробників PHP, що сприяли переходу від PHP 4 до PHP 5.

З часом інтерпретатори PHP стали доступними для більшості існуючих 32-розрядних та 64-бітних операційних систем, або будуючи їх із вихідного коду PHP, або використовуючи попередньо вбудовані бінарні файли. Для версій PHP 5.3 та 5.4 єдиними доступними бінарними дистрибутивами Microsoft Windows були 32-бітні збірки IA-32 , що потребували 32-бітного режиму сумісності Windows під час використання Інтернет-сервісів інформації (IIS) у 64-розрядній Windows платформа. PHP версії 5.5 зробила 64-розрядні версії x86-64 доступними для Microsoft Windows.

Офіційна підтримка безпеки для PHP 5.6 закінчилася 31 грудня 2018 року, але Debian 8.0 Jessie продовжить підтримку до червня 2020 року.

PHP отримав неоднозначні відгуки через відсутність підтримки рідного Unicode на рівні основної мови. У 2005 році проект, очолюваний Андрієм Змієвським, був ініційований для залучення вбудованої підтримки Unicode у всій PHP шляхом вбудовування бібліотеки Міжнародних компонентів для Unicode (ICU) та представлення текстових рядків як UTF-16 внутрішньо. Оскільки це призведе до значних змін як у внутрішній мові, так і в кодовому коді, планувалося випустити це як версію 6.0 мови, а також інші основні особливості тоді в процесі розвитку.

Однак дефіцит розробників, які зрозуміли необхідні зміни та проблеми з продуктивністю, що виникають при перетворенні на та з UTF-16, який рідко використовується у веб-контексті, призвів до затримок у проекті. Як результат, у 2009 році було створено реліз PHP 5.3, у якому багато функцій, які не переносяться Unicode, перенесені з PHP 6, зокрема простіри імен. У березні 2010 року проект в його нинішньому вигляді був офіційно відмовлений, і було підготовлено випуск PHP 5.4, що містить більшість функцій, що не належать до Unicode від PHP 6, такі як риси та повторне прив'язування закриття. Початкові сподівання полягали в тому, що буде сформований новий план інтеграції

Unicode, але станом на 2014 рік жоден не був прийнятий.

Протягом 2014 та 2015 років була розроблена нова основна версія PHP, яка була пронумерована PHP 7. Нумерація цієї версії передбачала певну дискусію. Хоча експеримент PHP 6 Unicode ніколи не був випущений, у кількох статтях та заголовках книг посилалося на ім'я PHP 6, що могло спричинити плутанину, якщо новий випуск повторно використовувати ім'я. Після голосування було обрано ім'я PHP 7.

Фундамент PHP 7 - це гілка PHP, яку спочатку називали PHP наступного покоління (phpng). Його автором були Микита Попов, Xinchen Hui та Дмитро Стогов, і він мав на меті оптимізувати продуктивність PHP шляхом рефакторингу Zend Engine, зберігаючи майже повну сумісність мови. Станом на 14 липня 2014 року показники на основі WordPress, які послужили основним набором орієнтирів для проекту phpng, показали майже 100% підвищення продуктивності. Очікується також, що зміни в phpng сприятимуть підвищенню продуктивності в майбутньому, оскільки більш компактні структури даних та інші зміни сприймаються як найбільш підходящі для успішної міграції до компілятора, що наразі працює (JIT). Через значні зміни, перероблений двигун Zend називається Zend Engine 3, який змінює Zend Engine 2, який використовується в PHP 5.

Через основні внутрішні зміни в phpng він повинен отримати нову основну кількість версій PHP, а не незначну версію PHP 5, відповідно до процесу випуску PHP. Основні версії PHP дозволяють порушити сумісність коду, тому PHP 7 представив можливість для інших вдосконалень за межами phpng, які вимагають перерв на сумісність. Зокрема, він стосувався таких змін:

Багато спадкових механізмів помилок PHP, які мають фатальний або відновлений рівень, були замінені сучасними об'єктно-орієнтованими винятками.

Синтаксис змінної перенаправлення був перероблений, щоб бути внутрішньо більш послідовним та повним, дозволяючи використовувати

оператори ->, [], (), {} і :: з довільними значущими лівими виразами.

Підтримка застарілих методів конструктора 4-х типів PHP застаріла.

Поведінка заяви foreach було змінено на більш передбачуваний.

Конструктори для кількох класів, вбудованих у PHP, які повернулися до нуля після відмови, були змінені, щоб замість них викинути виняток, для послідовності.

Кілька непідтримуваних або застарілих інтерфейсів програмування серверних додатків (SAPI) та розширень були видалені з ядра PHP, особливо це стосується застарілого розширення mysql.

Поведінку оператора list () було змінено, щоб видалити підтримку рядків.

Для застарілих роздільників стилю ASP було видалено підтримку <% і%> та <script language = "php"> ... </script>.

Виправлено контроль, який дозволяє оператору перемикання мати декілька замовчувань.

Підтримка підтримки шістнадцяткових чисел у деяких неявних перетвореннях з рядків у типи номерів була видалена.

Операторів ліворуч і праворуху змінили, щоб вони поводилися більш послідовно на всіх платформах.

Перетворення між цілими числами та числами з плаваючою комою були посилені та впроваджені більш послідовно на всіх платформах.

PHP 7 також включав нові мовні функції. Найбільш помітно, що він вводить декларації типу повернення для функцій, які доповнюють існуючі декларації типу параметрів, та підтримують скалярні типи (цілі, плавучі, рядкові та булеві) у оголошеннях параметрів та типів повернення.

2.6 Фреймворки у веб-розробці

2.6.1 Поняття фреймворку, веб-фреймворку

Фреймворки - це програмні продукти, які спрощують створення і підтримку технічно складних або навантажених проектів. Фреймворк, як правило, містить тільки базові програмні модулі, а все специфічні для проекту компоненти реалізуються розробником на їх основі. Тим самим досягається не тільки висока швидкість розробки, а й велика продуктивність і надійність рішень.

Веб-фреймворк - це платформа для створення сайтів і веб-додатків, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. За рахунок широких можливостей в реалізаціїї бізнес-логіки і високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків і веб-сервісів.

2.6.2 Важливість фреймоврків у розробці

З точки зору бізнесу розробка на фреймворку майже завжди економічно ефективніше і якісніше за результатом, ніж написання проекту на чистому мовою програмування без використання будь-яких платформ. Розробка без використання платформи може бути правильним рішенням тільки в двох випадках - або проект зовсім простий і не вимагає подальшого розвитку, або дуже навантажений і вимагає дуже низкоуровневой оптимізації (наприклад, веб-сервіси з десятками тисяч звернень в секунду). У всіх інших випадках розробка на програмній платформі швидше і якісніше.

Якщо порівнювати фреймворки з іншими класами платформ - SaaS, CMS або CMF - то фреймворки значно ефективніше використовувати в проектах зі складною бізнес-логікою і з високими вимогами до швидкості роботи, надійності і безпеки. Але в простих і типових проектах без значущих вимог швидкість і вартість розробки на фреймворку буде вище, ніж на SaaS або CMS.

Одним з головних переваг у використанні фреймворків є те, що фреймворк визначає уніфіковану структуру для побудованих на його базі додатків. Тому

додатки на фреймворками значно простіше супроводжувати і допрацьовувати, так як стандартизована структура організації компонентів зрозуміла всім розробникам на цій платформі і не потрібно довго розбиратися в архітектурі, щоб зрозуміти принцип роботи програми або знайти місце реалізації того чи іншого функціоналу. Більшість фреймворків для розробки веб-додатків використовує парадигму MVC (модель-уявлення-контролер) - тобто дуже в багатьох фреймворками ідентичний підхід до організації компонентів програми та це ще більше спрощує розуміння архітектури додатку навіть на незнайомому розробнику фреймворку.

Проектування архітектури ПЗ при розробці на фреймворку теж дуже спрощується - в методології фреймворків звичайно закладені кращі практики програмної інженерії та просто дотримуючись цих правил можна уникнути багатьох проблем і помилок в проектуванні. По суті, фреймворк - це безліч конкретних і абстрактних класів, пов'язаних між собою і впорядкованих згідно з методологією фреймворка. Конкретні класи зазвичай реалізують взаємні відносини між класами, а абстрактні класи являють собою точки розширення, в яких закладений у фреймворк базовий функціонал може бути використаний «як є» або адаптований під завдання конкретного додатка. Для забезпечення розширення можливостей в більшості фреймворків використовуються техніки об'єктно-орієнтованого програмування: наприклад, частини програми можуть успадковуватися від базових класів фреймворка або окремі модулі можуть бути підключені як домішки.

Екосистеми веб-фреймворків також багаті на готові реалізації багатьох функціональних можливостей. Розробникам при роботі над типовими завданнями не треба «винаходити велосипеди», так як вони можуть скористатися вже створеної спільнотою реалізацією. А це не тільки скорочує витрати часу і грошей, але і дозволяє домогтися більш високої стабільності рішення - компонент, який використовується і допрацьовується тисячами інших розробників зазвичай більш якісно реалізований і краще протестований на

всіляких сценаріях, ніж рішення, яке може в адекватні терміни розробити один розробник або навіть невелика команда.

2.6.3 Топ-10 фреймворків PHP

Рис. 2.3 Логотип Laravel

Представлений у 2011 році, **Laravel** став найпопулярнішим в світі безкоштовним фреймворком PHP з відкритим кодом, оскільки він може безпечно обробляти складні веб-програми, значно швидшими темпами, ніж інші фреймворки. Laravel спрощує процес розробки, полегшуючи звичайні завдання, такі як маршрутизація, сеанси, кешування та аутентифікація.

Переваги Laravel:

Laravel підходить при розробці додатків зі складними вимогами до бекенда, будь то малий чи великий проект. Встановлення Laravel було полегшене введенням Homestead, попередньо упакованого в коробці.

Цей фреймворк PHP повний функцій, які допоможуть вам налаштувати складні програми. Серед них: безперебійна міграція даних, підтримка архітектури MVC, безпека, маршрутизація, механізм перегляду шаблонів та автентифікація.

Laravel відрізняється високою експресією, а його швидкість та безпека відповідають очікуванням сучасного веб-додатку. Для розробників, які хочуть створити B2B або корпоративні веб-сайти, які розвиватимуться зі зміною веб-

Рис. 2.4 Логотип CodeIgniter

тенденцій, Laravel - це їх вибір.

Відомий своїм невеликим розміром (розміром є лише близько 2 Мб, включаючи документацію) **CodeIgniter** - це програма PHP, яка підходить для розробки динамічних веб-сайтів. Він пропонує численні попередньо вбудовані модулі, які допомагають створити надійні та багаторазові компоненти.

Переваги CodeIgniter

CodeIgniter - це легкий і простий фреймворк PHP, який легко встановлювати, на відміну від інших фреймворків. Завдяки простому процесу налаштування та високоілюстрованій документації, він ідеально підходить для новачків.

Основні особливості включають архітектуру MVC, неперевершене поводження з помилками, вбудовані засоби захисту, а також просту і відмінну документацію. Крім того, він створює масштабовані додатки.

У порівнянні з іншими фреймворками CodeIgniter значно швидше. Оскільки він також забезпечує високу продуктивність, це хороший вибір, коли ви хочете розробити легкі програми для роботи на скромних серверах. Одне застереження: випуски CodeIgniter дещо нерегулярні, тому фреймворк не є чудовим варіантом для програми, яка вимагає високого рівня безпеки.

Рис. 2.5 Логотип Symfony

Фреймворк **Symfony** бувзапущений в 2005 році, і хоча він існує набагато довше, ніж інші фреймворки цього списку, але він є надійною і зрілою платформою. Symfony - це обширний фреймворк PHP MVC і єдиний фреймворк, який, як відомо, дотримується стандартів PHP та веб-стандартів.

Переваги Symfony:

Symfony - ідеальний вибір для розробки масштабних корпоративних

проектів. Встановлення та налаштування на більшості платформ дуже просте.

Одна з його ключових особливостей це багаторазові PHP компоненти. Він також може похвалитися незалежністю від «движка» бази даних. Також він стабільний, відповідає більшості найкращих практик та моделей веб-дизайну, а також дозволяє інтегруватися з іншими бібліотеками постачальників.

Symfony також дуже гнучкий і може інтегруватися з більш великими проектами, такими як Drupal. Symfony та Laravel мають багато спільних і унікальних особливостей, через що важко сказати, який з цих двох фреймворків

Рис. 2.6 Логотип CakePHP

краще.

CakePHP допоможе розробити візуально вражаючі, завантажені функції. Крім того, CakePHP є однією з найпростіших фреймворків для вивчення, особливо через її CRUD (створювати, читати, оновлювати та видаляти). CakePHP вийшов на ринок на початку 2000-х, і відтоді він набув кращих показників роботи та багатьох нових компонентів.

Переваги CakePHP:

CakePHP легкий і простий в установці, оскільки вам потрібен лише веб-сервер і копія фреймворку.

Це робить хорошим вибором для комерційних програм завдяки функціям безпеки, які включають запобігання ін'єкцій SQL, перевірку вхідних даних, захист від підробки веб-сайтів (CSRF) та захист міжсхемових скриптів (XSS).
Деякі ключові особливості включають сучасний фреймворк, швидку побудову, належне успадкування класів, перевірку та безпеку. Крім того, CakePHP надає чудову документацію, безліч порталів підтримки та преміальну підтримку через

корпорацію Cake Development.

Рис. 2.7 Логотип Yii

Фреймворк **Yii** – насправді простий і еволюційний. Це високоефективний, створений на компонентах PHP фреймворк для розробки сучасних веб-додатків. Yii підходить для всіх видів веб-додатків. З цієї причини це універсальна структура веб-програмування.

Переваги Yii:

У Yii простий процес установки. Крім того, його надійні функції безпеки роблять його придатними для високобезпечних починань, таких як проекти електронної комерції, портали, CMS, та багато інших.

Він відрізняється високою швидкістю та продуктивністю, він дуже розширюється, і це дозволяє розробникам уникати складності написання повторюваних операторів SQL, оскільки вони можуть моделювати дані бази даних з точки зору об'єктів.

Yii має основну команду розробників та експертів, які сприяють її розвитку. Завдяки масовій спільноті, яка використовує його, ви можете розміщувати проблеми на форумах Yii та отримувати допомогу.

Yii надзвичайно розширюваний, і ви можете налаштувати майже кожен фрагмент коду ядра. Однак якщо ви це вперше використовуєте, будьте готові до

Рис. 2.8 Логотип Zend

крутої кривої навчання.

Фреймворк **Zend** - це повноцінна об'єктно-орієнтована структура, а той факт, що вона використовує такі функції, як інтерфейси та успадкування, робить її розширюваною. Він був побудований за спритною методологією, яка допомагає вам доставляти високоякісні додатки для корпоративних клієнтів. Zend дуже настроюється і дотримується кращих практик PHP - важливий момент для розробників, які хочуть додати функціональні можливості для проекту.

Переваги Zend:

Zend Framework - це чудова придатність для складних проектів на рівні підприємств. Він є кращою основою для великих відділів IT та банків.

Деякі ключові особливості включають компоненти MVC, простий

Рис. 2.9 Логотип Phalcon

хмарний API, шифрування даних та управління сеансами.

Він може інтегруватися із зовнішніми бібліотеками, а ви можете використовувати лише потрібні компоненти. Фреймворк Zend оснащений надзвичайно гарною документацією та має велику базу спільноти. Однак якщо ви розробник мобільних додатків, приготуйтеся до складностей.

Повноцінна структура PHP, що використовує схему дизайну веб-архітектури MVC, **Phalcon** спочатку був написаний на C та C ++ та випущена у 2012 році. Оскільки він постачається як розширення C, вам не доведеться турбуватися про вивчення мови програмування на C.

Переваги Phalcon:

Phalcon простий у встановленні та підходить для створення веб-додатків,

які легко налаштовуються, і які відповідають настановам щодо розвитку підприємства.

Основні характеристики включають підвищену швидкість виконання, управління активами, універсальний автозавантажувач та найвищу безпеку та кешування.

На відміну від інших фреймворків, Phalcon оптимізує продуктивність завдяки ефективному використанню пам'яті. Якщо ви хочете створити швидкий веб-сайт, спробуйте Phalcon.

З негативного боку, розробники Phalcon трохи повільно виправляють помилки, що може не збігатися з сьогоднішньою потребою у високому рівні безпеки.

Рис. 2.10 Логотип Swoft 1

Swoft - це високоефективний фреймворк програмного забезпечення мікросервісу PHP. Він публікується протягом багатьох років і став найкращим вибором для php.

Переваги Swoft:

Поставляється з ефективним пулом з'єднань Mysql / Redis / Rpc та підключенням всіх відключень. Розробники не дбають про об'єднання з'єднань, і відповідні компоненти були реалізовані.

AOP може використуватися для всіх об'єктів, якими керує контейнер Framework. Використання AOP дозволяє контролювати поведінку об'єктів екземпляра, не змінюючи внутрішню частину екземпляра.

Сервіс RPC поділений на RPC Server та RPC Client, і фреймворк забезпечує більш елегантний спосіб використання RPC-сервісів, таких як Dubbo. За допомогою службових мережевих фреймворків, таких як Istio / Envoy, і надає набір компонентів для швидкого побудови мікросервісного управління для

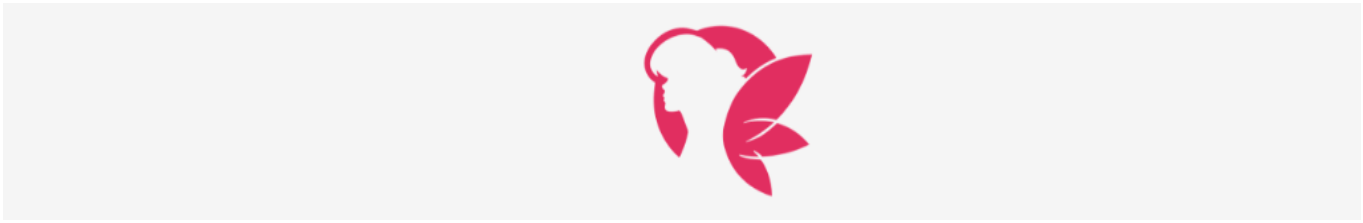

Рис. 2.11 Логотип PHPixie

малого та середнього бізнесу, включаючи реєстрацію та виявлення сервісів, долари обслуговування, зупинку обслуговування та центри налаштування.

Представлений в 2012 році і подібно до FuelPHP, **PHPixie** реалізує модель дизайну HMVC. Його метою було створити високоефективну основу для веб-сайтів, лише для читання.

Переваги PHPixie:

Початок роботи з PHPixie, який підходить для веб-сайтів у соціальних мережах, налаштованих веб-додатків та служб розробки веб-додатків, легко.

Основні характеристики включають архітектуру HMVC, стандартну ORM (об'єктно-реляційне відображення), перевірку вводу, можливості авторизації, автентифікацію та кешування.

PHPixie побудований за допомогою незалежних компонентів. З цієї причини ви можете використовувати його без самих фреймворків. Зауважте, що в PHPixie є порівняно мало модулів. Крім того, йому не вистачає підтримки для компонентів, незалежно виготовлених від залежностей. Оскільки він порівняно новий, він менш популярний і має меншу спільноту користувачів, ніж інші

Рис. 2.12 Логотип Slim

фреймворки.

Slim - ще один популярний мікро-фреймворк PHP, який допомагає розробникам швидко створювати прості, але потужні веб-програми та API.

Переваги Slim:

Так само як і PHPixie, Slim легко вивчити. Розробники PHP використовують Slim для розробки API RESTful та веб-сервісів.

Основні функції включають маршрутизацію URL-адреси, шифрування сеансів та файлів cookie, кешування HTTP на стороні клієнта тощо.

Це найкращий фреймворк для невеликого веб-додатка, який не обов'язково потребує повного пакету PHP. Крім того, активне обслуговування та дружня документація роблять Slim супер зручним для користувачів.

2.6.4 Фреймворк Yii

Зважаючи на вище наданий список, та на наявність досвіду роботи з фрейморком Yii, було вирішено вибрати його для створення веб-додатку.

Назва фреймворка - це акронім «Yes It Is!!». Серед PHP-фреймворков Yii виділяється вельми гарною продуктивністю при відносно легкому освоєнні фреймворку.

Фреймворк активно розвивається спільнотою. Yii не виглядає «монстром» в порівнянні з фреймворками Symfony та Zend Framework, кодова базою яких завжди є дуже об'ємна. Фреймворк Yii досить простий в освоєнні та використанні, що дозволяє швидко розроблювати проекти на ньому. Однак при виборі в яксоті платформи для розробки веб-проекту цей фреймворк, треба брати до уваги тей факт, що швидкість розробки на ньому типових вирішень всеодно буде нижче, ніж розробка на CMS. Як і всі фреймворки, Yii «заточений» під розробку технічно складних веб-проектів: бізнес-додатків, веб-сервіси, а також сайтів із складною бізнес-логікою та вимогливих до швидкої роботи.

Основні переваги і можливості фреймворка Yii:

1. Забезпечує високу продуктивність щодо інших php-фреймворків.
2. Заснований на парадигмі MVC (Модель-Представлення-Контролер).
3. Є інтерфейси DAO і ActiveRecord для роботи з базами даних (використовується PDO).

4. Підтримує інтернаціоналізацію.

5. Дозволяє кешувати як сторінки цілком, так і окремі фрагменти.

6. Здійснює перехоплення і обробка помилок.

7. Має функціонал роботи з формами, забезпечує їх побудова та валідацію.

8. Реалізовано аутентифікація і авторизація.

9. Зручний для реалізації AJAX-інтерфейсів, інтегрується з jQuery.

10. У фреймворк вбудовані генератори базового PHP-коду для CRUD-операцій (скаффолдинг).

11. Підтримує теми оформлення.

12. Має можливість підключення сторонніх бібліотек.

13. Працює з міграціями баз даних (генерація, застосування і відкат).

14. Дозволяє здійснювати автоматичне тестування і вести розробку в стилі TDD.

15. Підтримує стиль REST.

2.7 IDE та текстові редактори

Для того, щоб залишатися конкурентоспроможними та продуктивними, написання хорошого коду за мінімальний час є найважливішим навиком, яким повинен володіти кожен розробник програмного забезпечення. Написання коду стало основною вимогою для багатьох нових сегментів, включаючи IoT та AI.

Як засвідчує багато досвідчених кодерів, правильний IDE та / або редактор коду є життєво важливим для створення та підтримки коду високої якості. Оскільки кількість та стиль написання коду збільшуються, а нові мови програмування часто з’являються, важливо, щоб розробники програмного забезпечення вибирали правильний IDE для досягнення поставлених цілей.

Може здатися, що в цьому віці, коли написання коду стало достатньо поширеною майстерністю, все ще існує плутанина щодо точного визначення IDE та редактора коду. Частину проблеми можна було простежити в тому, що

лінія між ними розмивається через перехрещування функцій.

По суті, інтегроване середовище розробки або IDE - це автономний пакет, який дозволяє писати, компілювати, виконувати та налагоджувати код в одному місці. З іншого боку, редактор коду - це текстовий редактор з декількома функціями, які полегшують процес написання коду, або за допомогою власних можливостей, або за допомогою додаткових плагінів.

Як правило, IDE зосереджена на одній мові і містить компілятор / інтерпретатор та відладчик, характерні для цієї мови. На відміну від цього, редактори коду мають більш загальне призначення за своїми можливостями, тому що вони здатні працювати з низкою мов програмування. Редактори коду обмежуються написанням коду і не виходять за рамки цього етапу.

І редактори IDE, і редактори коду мають спільні функції, такі як заповнення коду, підказки, виділення розділів коду та спеціальне складання розділів коду. Вибір між IDE або редактором коду є значною мірою особистими уподобаннями, конкретною мовою програмування та робочими потоками.

Список кращих IDE для PHP:

1. PHPStorm
2. Netbeans
3. Aptana Studio
4. Eclipse
5. Visual Studio
6. ZendStudio

Список кращих текстових редакторів для PHP:

1. Sublime Text
2. Visual Studio Code
3. Atom
4. Notepad++
5. Coda
6. Brackets

Для створення web-додатку я обрав PHPStorm, оскільки в нього дуже багато переваг над іншими IDE, а найголовніше компанія Jetbrains безкоштовно надає ліцензію PHPStorm для студентів.

PhpStorm ідеально підходить для роботи з Yii, Symfony, Laravel, Drupal, WordPress, Zend Framework, Magento, Joomla, CakePHP та іншими фреймворками.

Редактор фактично 'отримує' ваш код і глибоко розуміє його структуру, підтримуючи всі функції мови PHP для сучасних і застарілих проектів. Він забезпечує найкраще завершення коду, рефакторинг, попередження помилок під час руху та інше.

Також в PHPStorm максимально легко користуватись передовими технологіями, такими як HTML 5, CSS, Sass, Less, Stylus, CoffeeScript, TypeScript, Emmet та JavaScript, з можливістю рефакторингу, налагодження та тестування одиниць.

Завдяки цієї IDE можна виконувати багато рутинних завдань, для цього у ній є: інтеграції систем управління версіями, підтримці віддаленого розгортання, базам даних / SQL, інструментам командного рядка, Docker, Composer, клієнту REST та багатьом іншим інструментам.

Сотні перевірок дбають про перевірку коду під час введення тексту, аналізуючи весь проект. Підтримка PHPDoc, аранжувальник коду та форматування, швидке виправлення та інші функції допомагають написати акуратний код, який легко підтримувати.

Можна надійно відрефакторити код за допомогою безпечного перейменування, переміщення, видалення, вилучення методу, вбудованої змінної, підштовхування членів вгору / витягування членів вниз, зміна підпису та багатьох інших рефактори. Рефакторинг, що стосується мови, допоможе здійснити зміни в межах всього проекту за декілька клацань, які все можна безпечно скасувати.

PhpStorm відомий своїм візуальним відладчиком з нульовою

конфігурацією, що забезпечує надзвичайну інформацію про те, що відбувається у вашій програмі на кожному кроці. Він працює з Xdebug і Zend Debugger і може використовуватися як локально, так і віддалено. Тестування блоків з PHPUnit, BDD з Behat та інтеграцією профілерів також доступні.

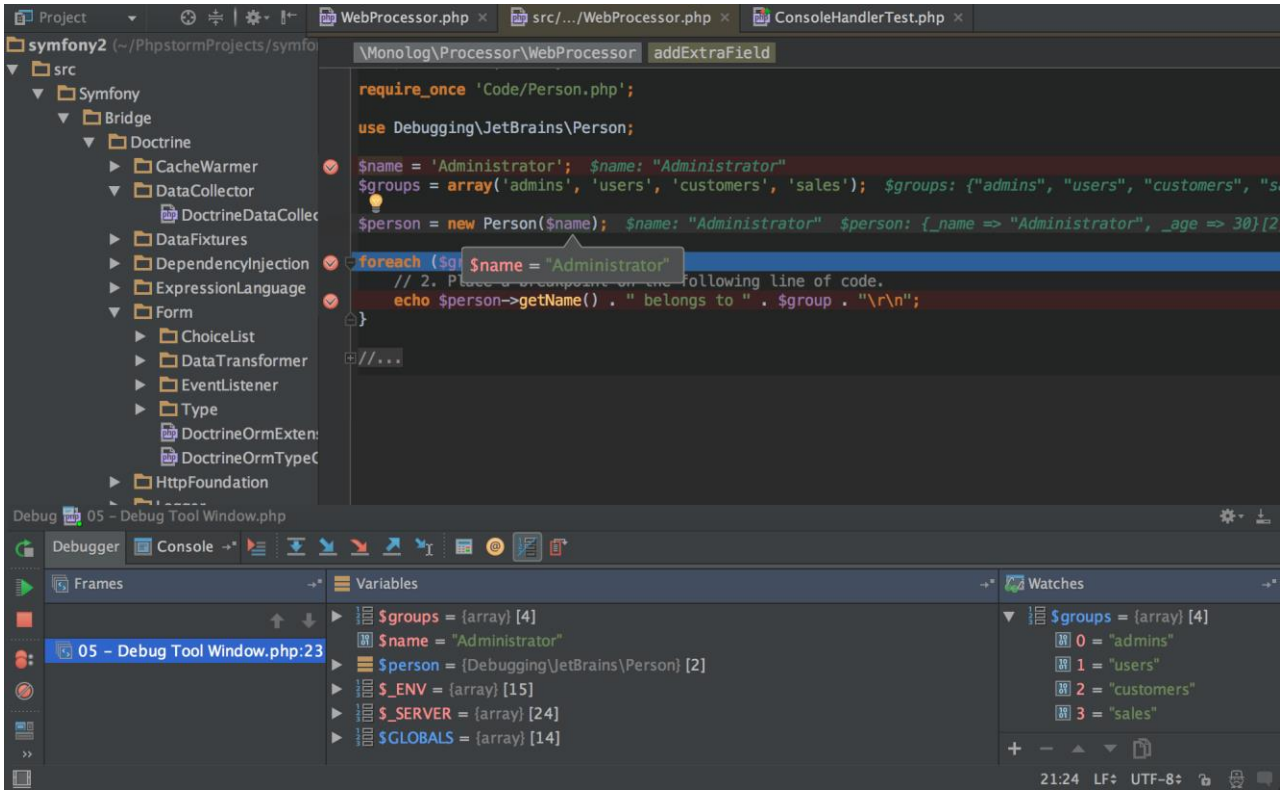

Рис. 2.13 Інструмент налагодження в PHPStorm

2.8 Open Server

Для створення web-додатку було обрано серверну платформу Open Server. Open Server - це портативний локальний WAMP / WNMP сервер, який має багатофункціональну керуючу програму і великий вибір підключаємих компонентів. Open Server - це перший проект, який наповнений професійними інструментами та створений спеціально для веб-розробників з урахуванням їх рекомендацій і побажань.

Для налагодження скриптів у різному оточенні Open Server пропонує на

вибір відразу два види HTTP серверів, різні версії PHP і СУБД модулів, а так само можливість швидкого перемикання між ними.

Основні компоненти:

- OSPanel **5.3.5**
- Apache **2.2.31** / **2.4.38** / **2.4.41**
- Bind **9.14.5**
- ConEmu **19.07.14**
- FTP FileZilla **0.9.60**
- Ghostscript **9.27**
- Git **2.23.0**
- HeidiSQL **10.2.0.5599**
- Nginx **1.17.3**
- NNCron Lite **1.17**
- Sendmail **32**
- Wget **1.20.3**

Системы управління базами даних:

- MariaDB **5.5.63** / **10.0.38** / **10.1.38** / **10.2.22** / **10.3.13**
- Memcached **1.2.6** / **1.4.5**
- MongoDB **2.4.14** / **2.6.12** / **3.0.15** / **3.2.22** / **3.4.19** / **3.6.11** / **4.0.6** / **4.2.0**
- MySQL **5.1.73** / **5.5.62** / **5.6.43** / **5.7.25** / **8.0.15**
- PostgreSQL **9.2.24** / **9.3.25** / **9.4.21** / **9.5.16** / **9.6.12** / **10.7** / **11.2**
- Redis **2.8.2402** / **3.0.504** / **3.2.100**

PHP модулі:

- PHP **5.2.17**
- PHP **5.3.29**
- PHP **5.4.45**
- PHP **5.5.38**
- PHP **5.6.40**
- PHP **7.0.33**

- PHP **7.1.32**
- PHP **7.2.22**
- PHP **7.3.9**

PHP додатки:

- Adminer **4.7.3**
- PHPMemcachedAdmin **1.3**
- PHPMyAdmin **4.9.0.1**
- PHPPgAdmin **7**
- PHPRedisAdmin **1.11.4**

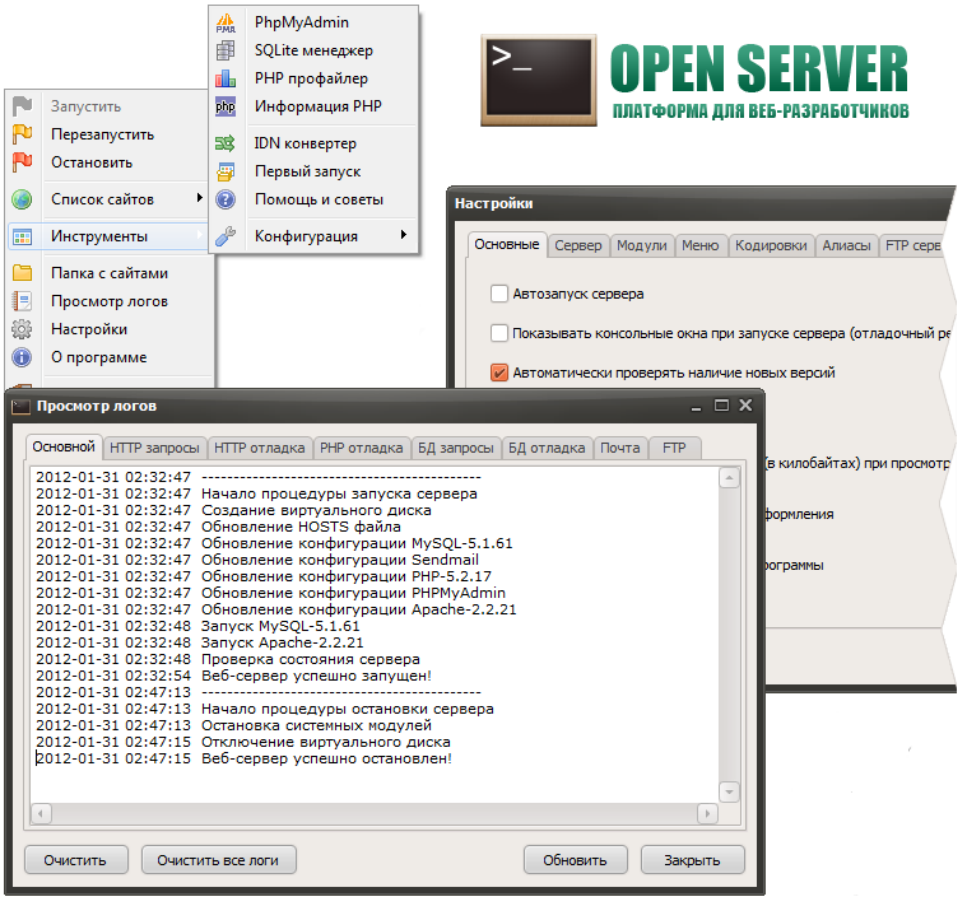

Рис. 2.14 Інструменти Open Server

Open Server - це єдиний проект у цьому сегменті, в якому є Nginx! Також тут реалізовано зручне підключення правил рерайтингу через файли .nxaccess в корені домену, а PHP працює в режимі True FastCGI .

Всі компоненти цього проекту взяті з офіційних репозиторіїв і завжди оновлюються до актуальних версій з кожним оновленням пакета.

Перш за все треба відзначити, що Open Server - це цілком і повністю портативний сервер. Ніяких системних сервісів, куп сміття в реєстрі і system32. Ви можете завжди брати його з собою на флешці та запускати на робочій / домашній машині без побоювання що у вас щось не запрацює.

У разі відсутності на комп'ютері потрібних системних компонентів Open Server встановить їх сам, досить вибрати в меню [Інструменти - Перший запуск] якщо сервер запускається на комп'ютері вперше.

Не буду описувати основні можливості програми, оскільки в більшості своїй вони стандартні для такого роду софта. Само собою, що за допомогою Open Server можна запустити / зупинити сервер або відкрити потрібний домен. Набагато цікавіше поговорити про специфічні «фічі», які роблять Open Server особливим, справді особливим:

- детальний перегляд логів всіх компонентів в реальному часі;

- вибір HTTP, СУБД і PHP модулів в будь-якому поєднанні;

- підтримка SSL і кириличних доменів з коробки;

- підтримка аліесів, або по іншому - доменних покажчиків, а також дуже зручна форма їх налаштування;

- створення локального піддомена без втрати видимості основного домену в мережі інтернет;

- доступ до доменів (в один клік) і швидкий доступ до шаблонів конфігурації модулів;

- багатомовний інтерфейс (Російська, Українська, Білоруська, Англійська);

Програма постійно удосконалюється, усі адекватні прохання з боку користувачів Open Server детально вивчаються і більшість з них реалізується.

ВИСНОВОК ДО РОЗДІЛУ 2

Цей розділ демонструє технології, які будуть використовуватись при розробці Web-додатку «Система контролю задоволеності співробітників». Було розглянуті технології HTML, CSS, JavaScript, бібліотека jQuery. Також було розноянуто мова програмування PHP та її основні фреймворки. Після опису зрівняння фрейморку було описано обраний фреймворк для створення додатку, його переваги та недоліки. Також були описані обрані інструменти для розробки, такі як IDE phpStorm та серверна платформа Open Server, які завжи залишаться в наборі моїх інструментів для розробки, адже вони дуже зручні та наповнені потрібним функціоналом для вдалої і швидкої розробки. Головне одразу правильно обрати технології, щоб у подальшому не виникало складнощів з удосконаленням проекту.

РОЗДІЛ 3
СТВОРЕННЯ WEB-ДОДАТКУ «СИСТЕМА КОНТРОЛЮ ЗАДОВОЛЕНОСТІ СПІВРОБІТНИКІВ»

3.1 Постановка задачі

Для реалізації проекту нам потрібно створити конструктор опитувань, в якому можна створювати питання різних типів: вибір із списку, шкала, вільна відповідь, группа вільних відповідей, випадаючий список, матриця, ранжування, дата, ел.пошта, номер телефону, завантаження файлу. Для кожного типу питання має бути можливість ввести текст питання, додати варіанти відповіді (якщо тип питання може мати таку функцію). Також для кожного питання має бути налаштування обов'язковості відповіді на нього. Звісно кожне питання та варіант відповіді повинен видалятися за потреби. Типи питань як «вибір із списку» та «випадаючий список» мають мати налаштування мульти-вибору (коли користувач хоче вибрати декілька варіантів відповіді).

Після збереження опитування ми маємо потрапляти на сторінку «усіх опитувань» де можна продивитися загальні відомості про статус опитування, статус розсилання цього опитування, кількість

відправлених\отриманих\продивлених листів, кількість пройдених опитувань, дата створення опитування, та дата останньої відповіді. Для кожного опитування має бути функція редагування, перегляд сконструйованого опитування (те, як користувач буде його бачити), статистика, та архівація, на випадок, якщо опитування не потрібно, але статистика має залишитись на майбутнє, або якщо при створенні нового опитування потрібно буде вибрати шаблон з готового опитування (також потрібна функція при створенні опитування).

Кафедра КІТ (47)

Для кожного опитування має бути сторінка налаштування розсилки для цього опитування. На цій сторінці користувач зможе вибрати тип розсилки (email\sms) та прикріпити таблицю excel зі списком опитуваних, з певною послідовністю назв колонок: прізвище, ім’я, по батькові, телефон, е-мейл, стать, канал розсилання (email/sms). Після додавання даних розсилки, на цій сторінці має бути можливість видалити прикріплені дані, якщо їх потрібно змінити (до тих пір поки розсилання не почалось), також перехід на сторінку, де можна продивитися завантажені дані. Звісно на цій же сторінці (налаштування розсилки) має бути перехід на сторінку налаштування змісту листа\смс-повідомлення. Після прикріплених даних розсилання та налаштування листа\смс-повідомлення має активуватись кнопка активації розсилання, після якої листи\смс-повідомлення будуть відправлені одержувачам з даних розсилки. Також має бути реалізована система оновлення статусів відправлених листів і звісно статистика, яка буде доступна після закінчення опитування (закінчування опитування має здійснюватись по натисканню кнопки, після чого опитування буде неактивне і навіть ті хто мають посилання не зможуть більше його пройти).

3.2 Підготовка до розробки додатку

Для початку розробки попередньо було встановлено PHPStorm, Open Server, інтерфейс яких було проілюстровано у розділі вище.

Open Server – це дуже зручна платформа, адже у ній є все що потрібно, а користуватись нею – одне задоволення.

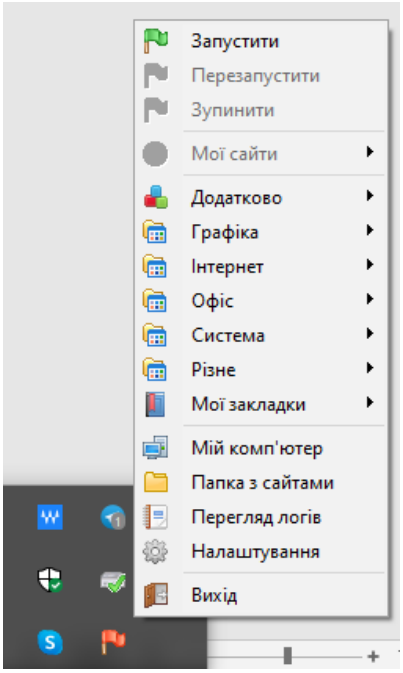

Рис. 3.1 Панель керування Open Server

Для початку нам потрібен локальний сервер, на якому ми розмістемо наш додаток. В Open Server – це питяння однієї хвилини, потрібно тільки запустити платформу і відкрити панель керування, а після натиснути «Запустити».

Далі потрібно створити директорію на сервері щоб розмістити там проект. Для цього у папці Domains, що розташована в дерикторії Open Server створюємо папку constructor, що і буде доменом додатку на локальному сервері (constructor, тому-що осн овна увага додатка приділена створенню конструктора опитувань).

Далі потрібно перезапустити сервер. На наступній ілюстрації видно що домен «constructor» додався до списку «Мої сайти».

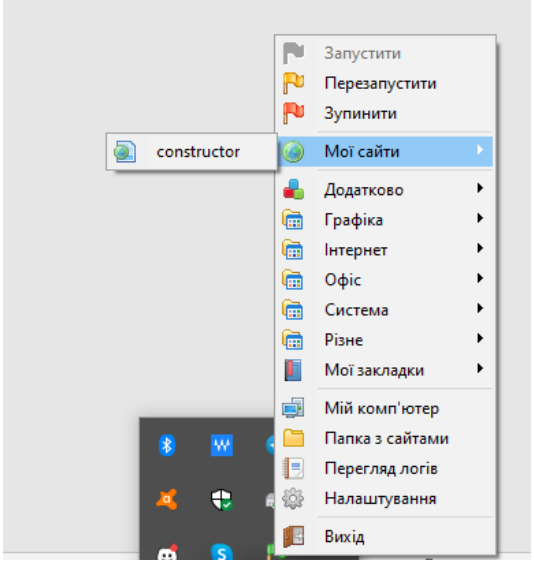

Рис. 3.2 Список доменів серверу

Щоб переірити працездатність домену потрібно натиснути в панелі керування на наш домен, або у адресному рядку браузера ввести «constructor». Для наочної перевірки створимо в папці проекту файл index.php і викличимо мовну конструкцію PHP «echo» що відображає текст у HTML розмітці. Оскільки налаштування серверу не змінювалися, то початково, при посиланні на домен, відкриється файл index.php. Для перевірки роботи виведемо на екран привітання: «Привіт!»

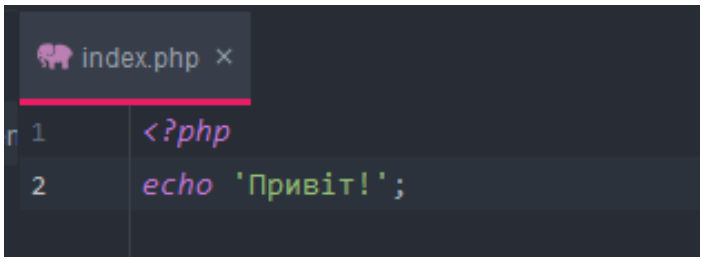

Рис. 3.3 Код привітання

Весь php код має починатись з відкриваючого тегу «<?php» та закінчуватись закриваючим тегом «?>», але якщо у файлі крім php коду немає іншого, то закриваючий тег є не обов’язковим.

Отже тепер перевіримо працездатність серверу та домену, ввівши у адресний рядок браузера «constructor».

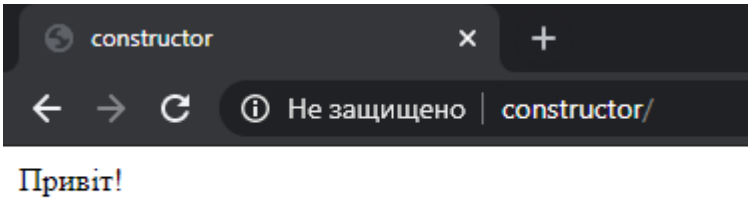

Рис. 3.4 Перевірка роботи серверу

Як видно на ілюстрації вище наш сайт працює, отже можна починати процес розробки проекту.

3.3 Розгортання та налаштування фреймворку Yii2 на проекті

Для установки фреймворку нам буде потрібен пакетний менеджер для мови програмування PHP - Composer. З офіціально сайту завантажимо пакетний менеджер та встановимо його на компьютер. Для перевірки роботи Composer

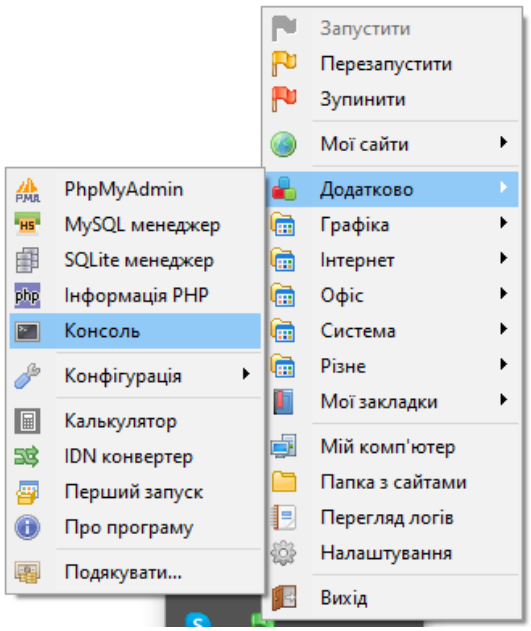

Рис. 3.5 Відкриття консолі

відкриємо консоль в інструментах Open Server:

Тепер введемо команду «composer». Як видно на рис. 3.6, ми отримали інформацию про версію та існуючі команди Composer, отже він вдало встановлений, і можна перейти до встановлення фреймворку.

Фреймворк Yii2 має два набори пакетів: Basic та Advanced. Basic – має

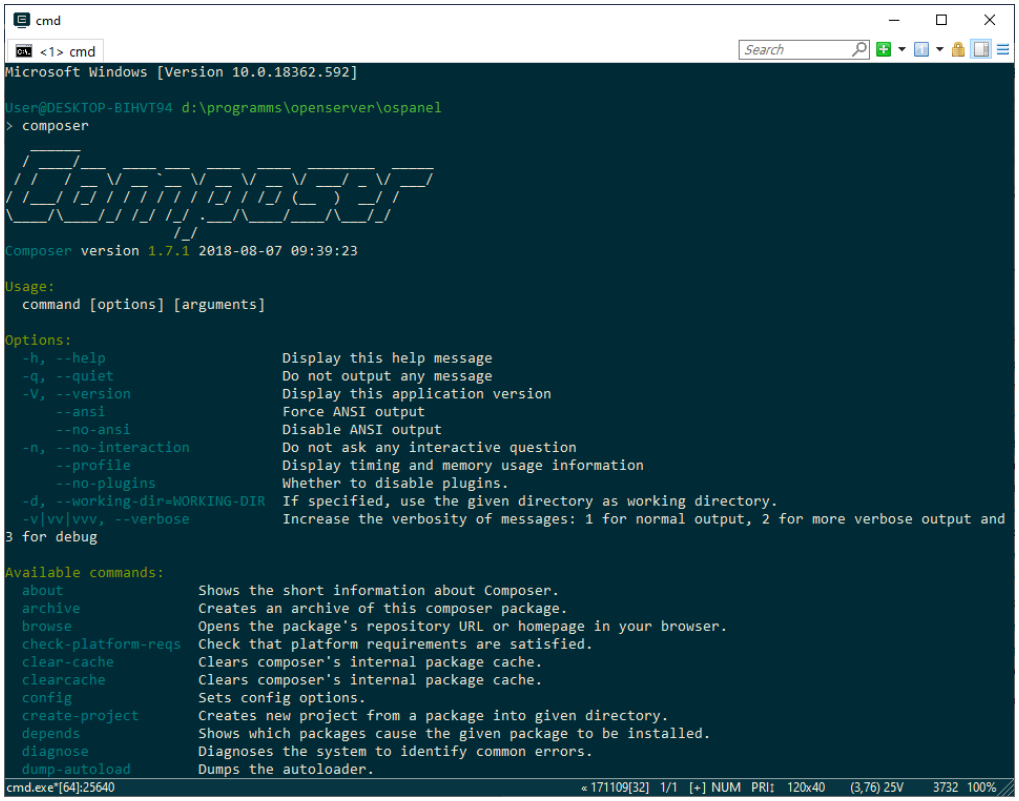

Рис. 3.6. результат команди

реалізацію звичайного додатку, Advanced – приклад більш складного додатку, який поділяється на backend, frontend, console тобто проект розділяється на три застосунки, з своїм налаштуванням. Мій проект буде складатися окремо з адмін-панелі та відображення опитувань, а також матиме консольні команди для оновлення статусу розсилання, тому мені буде потрібен Advanced.

Yii2 Advanced має наступну структуру:

common

config загальні налаштування додатку

mail/ файли представлень для e-mail листів

models/ загальні моделі, які використовуються в бекенді іфронтенді

console

config/ налаштування консольного застосунку

controllers/ консольні контролери (команди)

migrations/ міграція БД

models/ специфічні моделі для консольного застосування

runtime/ файли згенеровані під час роботи консольного застосування

backend

assets/ ресурси. наприклад, JavaScript, CSS

config/ конфігурація backend

controllers/ контролери

models/ специфічні моделі для backend

runtime/ файли згенеровані під час роботи backend-застосунку

views/ файли представлень

web/ вхідний скрипт backend та веб-ресурси

frontend

assets/ ресурси. наприклад, JavaScript, CSS

config/ конфігурація frontend

controllers/ контролери

models/ специфічні моделі для frontend

runtime/ файли сгенеровані під час роботи frontend застосунку

views/ файли представлень

web/ вхідний скрипт фронтэнда та веб-ресурси

widgets/ віджети frontend

vendor/ пакети залежностей композер

environments/ специфічні речі оточення

tests різні тести додатку

codeception/ Codeception тести

.gitignore - файл із списком ігнорованих файлів для git

composer.json - файл конігурацій композера

init - скрипт ініціалізації оточення web-додатку

init.bat – теж саме, але для Windows

requirements.php - скрипт перевірки залежностей Yii framework 2

yii - файл запуску консольного застосунку

yii.bat - теж саме, але для Windows

Для завантаження Yii2 Advanced відкриваємо консоль, переходимо в папку нашого проекту та викликаємо команду «composer create-project --prefer-dist yiisoft/yii2-app-advanced yii2-advanced» і чекаємо закінчення завантаження.

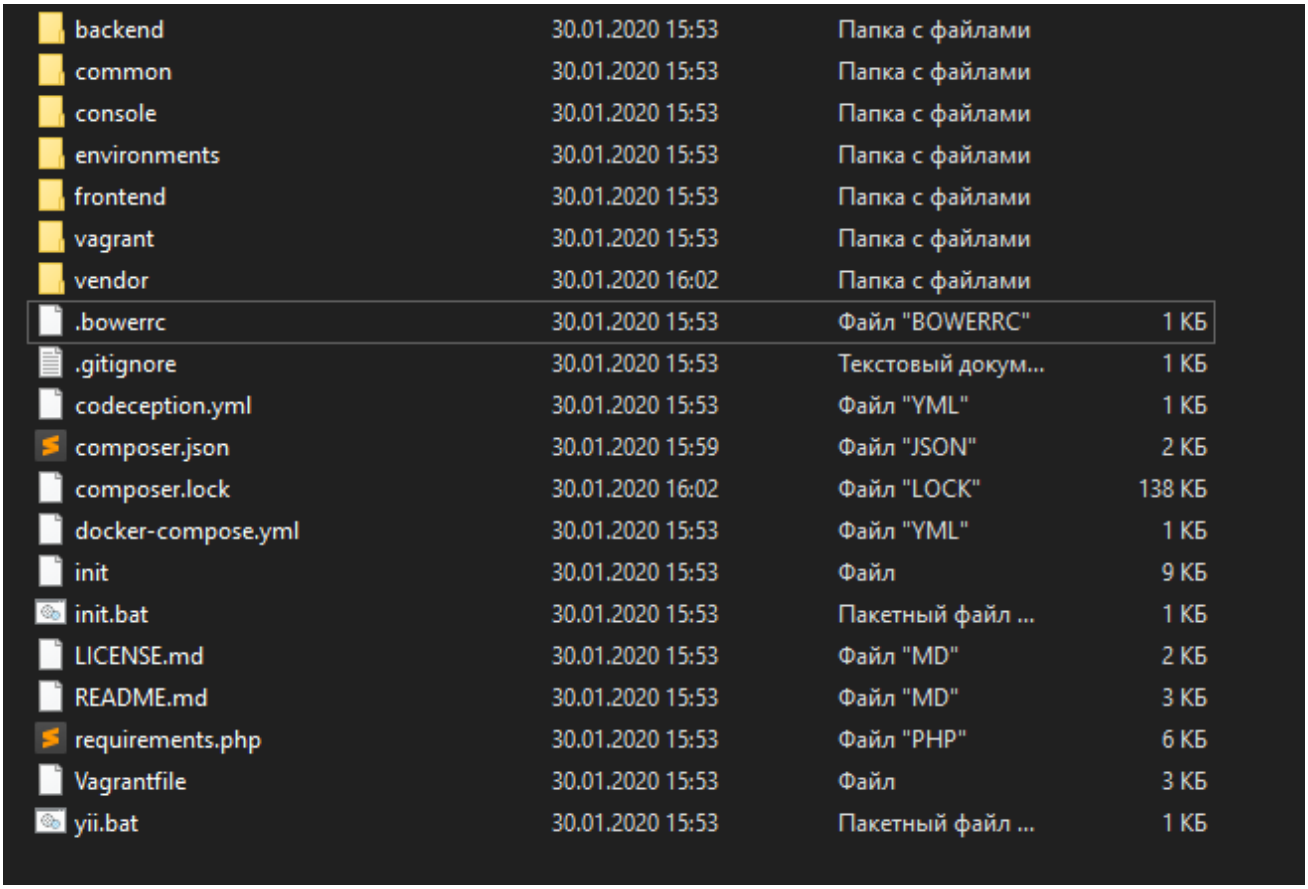

Рис. 3.7 Результат завантаження

На ілюстарції вище можна спостерігати те що завантаження відбулося успішно і ми маємо раніше описану структуру.

Далі потрібно ініціювати проект. Для цього в консолі переходимо у папку з проектом та викликаємо команду «init» та обираємо [0] Development і вводио «yes» (якщо обрати Development то це означає що будуть відображатись усі помилки коду, Production – відключення усіх виведень помилок). (рис. 3.8)

Наступним кроком буде створення бази даних для нашого проекту. Для цього в Open Server є веб інтерфейс для адміністрування базами даних –

```
User@DESKTOP-BIHVT94 d:\Programms\OpenServer\OSPanel\domains
> cd constructor

User@DESKTOP-BIHVT94 d:\Programms\OpenServer\OSPanel\domains\constructor
> init
Yii Application Initialization Tool v1.0

Which environment do you want the application to be initialized in?

  [0] Development
  [1] Production

  Your choice [0-1, or "q" to quit] 0

  Initialize the application under 'Development' environment? [yes|no] yes
```

Рис. 3.8 Ініціалізація проекту

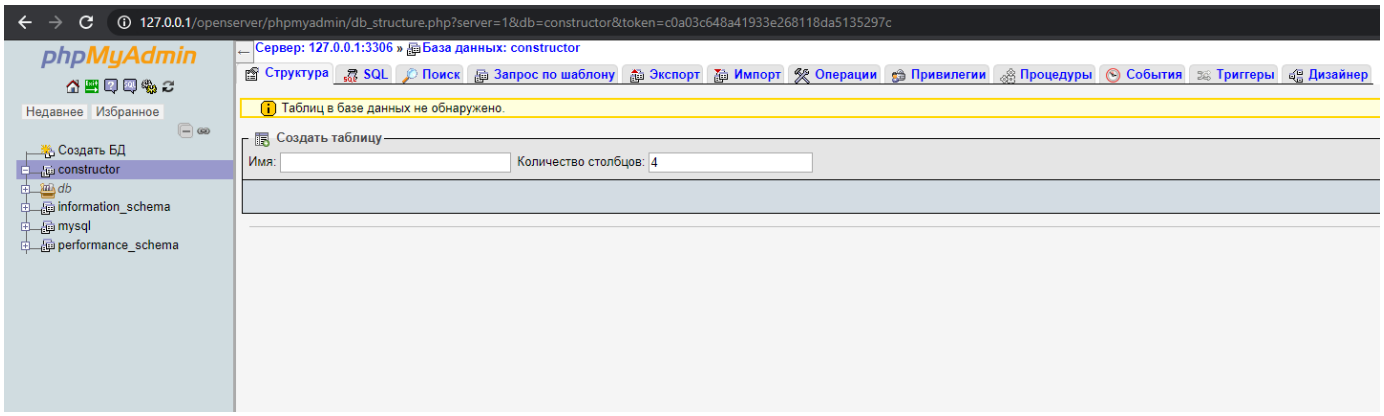

Рис. 3.9 Інтерфейс phpMyAdmin

phpMyAdmin. Через панель керування відкриваємо цей інтерфейс та створюємо базу даних (далі БД) і називаємо так як і наш проект «contstructor». При створенні вибираємо кодування utf8mb4_general_ci, що рекомендується при створенні БД з вмістом кирилиці.

Тепер потрібно прописати параметри для підключення до БД. Для цього в

```
'components' => [
    'db' => [
        'class' => 'yii\db\Connection',
        'dsn' => 'mysql:host=localhost;dbname=constructor',
        'username' => 'root',
        'password' => '',
        'charset' => 'utf8',
    ],
```

параметри доступу до БД 1

файлі *common/config/main-local.php* прописуємо параметри, зображені на наступній ілюстрації.

Після цього в консолі прописуємо команду «yii migrate», що створить у БД усі таблиці, які є в папці міграцій. Стандартно - це таблиця міграцій (для відроблених міграцій) та таблиця користувачів, з колонками: емейл, логін, зашифрований пароль та ін.

Отже тепер фреймворк встановлено, і ми можемо отримати доступ до застосувань frontend і backend за посиланнями http://constructor/frontend/web та http://constructor/backend/web відповідно, що відповідає шляху до файлів index.php у обох застосунках. Для зручності подальшої розробки у корні проекту та в директоріях web кожного застосунку створимо файл конфігурації серверу .htaccess з відповідними параметрами (ілюстрація вмісту файлів наведена нижче). Це робиться для того щоб посилання були зручнішими та зрозумілими для користувачів. Відтепер доступ до http://constructor/frontend/web буде

здійснюватись за посиланням http://constructor, відповідно http://constructor/admin до «бекенду». (див. рис. 3.10 та 3.11)

Тепер за допомогою компоненту Yii2 «urlManager» налаштуємо перехід на контроллери застосунків, щоб наші посилання були ще зручнішими. Наприклад доступ до контроллеру SiteController і його екшену (action) actionIndex в директорії backend буде здійснюватися за посиланням http://constructor/admin/site/index. Для цього в config/main.php обох директив

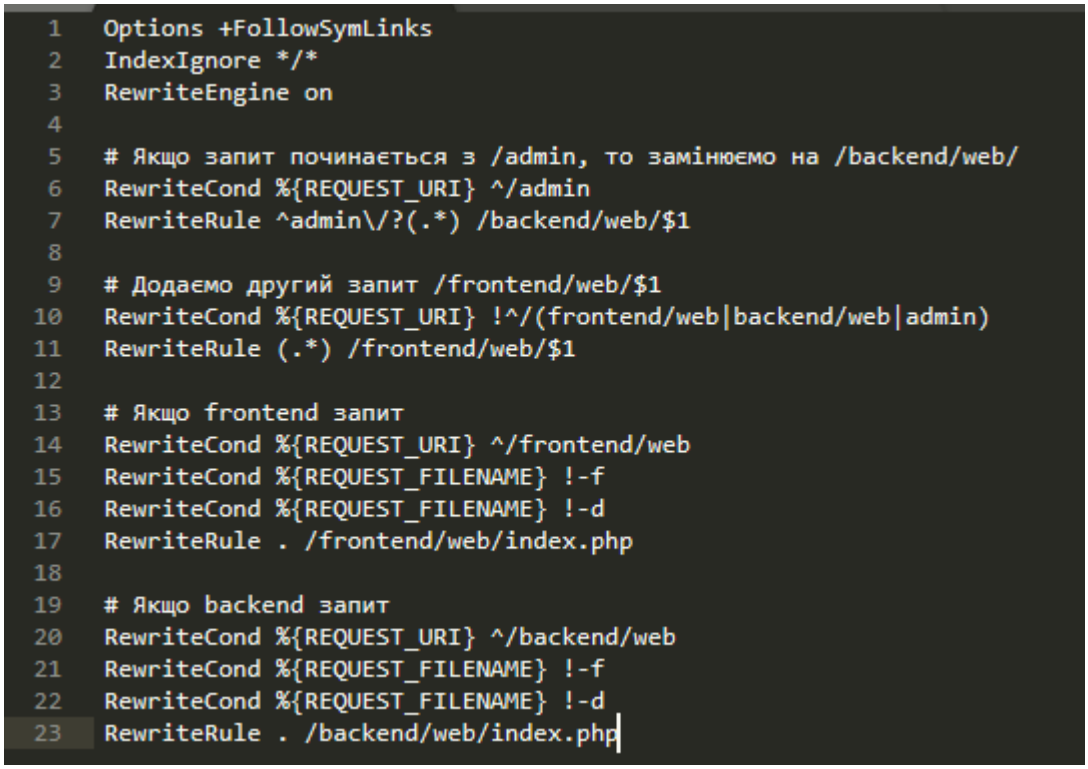

Рис. 3.10 Файл .htacces в корені проекту 1

Рис. 3.11 Файл .htaccess у директоріях web

прописуємо наступні параметри:

Отже тепер наш проект повністю налаштований і готовий для подальшої розробки.

Рис. 3.12 параметри urlManager

3.4 Проектування БД

Для реалізаціїї завдання нам потрібно спроектувати відповідну БД. Ми маємо сутність опитування, яке буде мати певну кількість питань, певних типів, та у свою чергу кожне з питань матиме свої варіанти відповідей чи певні налаштування. Збережені відповіді будуть прив’язані до питання та\чи до варіанту відповіді, для зручного маніпулювання даними в фреймворці Yii2. Також потрібна таблиця «клінтів» яка буде заповнюватись при завантаженні Excel файлу за даними для розсилання. При кожному завантаженні файл Excel буде зберігатися на сервері, що займає певне місце, і з часом таких файлів може зібратися дуже велика купа, тому при видаленні опитуванні мають видалятись усі данні, зв’язані з цим опитуванням (питання, варіанти відповідей, відповіді, файли завантажень, таблиця клієнтів і тд). Тому буде створено таблицю для завантажень даних розсилки, де буде збережено шлях на сервері до файлу, час завантаження та ідентифікатор завантаження, до якого можна буде прив’язати клієнтів з таблиці клієнтів, для зручного маніпулювання даними. Авжеж потрібна таблиця історії відповідей, яка буде пов’язувати «клієнта» та його відповіді в таблиці відповідей. І накінець нам потрібно дві таблиці: перша для збереження налаштувань змісту повідомлень, та таблиця в якій при віправці листа буде зберігатись його ідентифікатор, ідентифікатор отримувача (з таблиці

клієнтів) та ідентифікатор опитування, знову ж таки, для зручного маніпулювання.

В фреймворці Yii використовується архітектурний шаблон MVC, тобто напочатку ми звертаємося до певного контроллеру, який має набір екшені. Далі ми звертаємося до окремого екшену, який в свою чергу, за необхідністю, викликає модель (трохи далі буде детальніше описані моделі) і отримує певні дані для подальшої обробки і накінець, після закінчення усіх вичислень та операцій, екшн звертається до файлу виду (View) що представляє собою HTML код із вставками PHP. Контроллер передає необхідні данні до файлу виду і ми отримуємо унікальну сторінку, в залежності від ситуації.

Таблиці БД в фреймворці представлені як «моделі», це класи які наслідуються від класу ActiveRecord, який в свою чергу від класу Model. Active record дозволяє звертатись до таблиці БД у представленні ООП. У наступному

Quiz.php

```php
<?php

namespace common\models;
use ...

class Quiz extends ActiveRecord{

    public static function tableName()
    {
        return '{{quiz}}';
    }

    public function rules() {

        return [
            [['name'], 'required']
        ];
    }

```

3.13 Модель таблиці опитування

рисунку створено модель для таблиці сутності «опитування».

Відтепер замість довгих та незручних SQL запитів до таблиці опитувань можно звертатись у вигляді ООП. Наприклад, якщо ми хочемо отримати дані

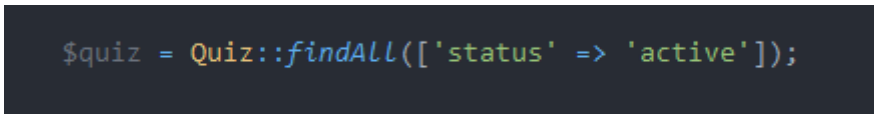

Рис.3.13 Звернення до таблиці опитувань

усіх опитувань, у яких, наприклад, активний статус (див. рис. 3.14)

Результатом, який ми помістили у змінну $quiz є масив об’єктів, тобто екземплярів класу Quiz, який, нагадаю ще раз, наслідується від класу ActiveRecord, а той від Model. Це означає, якщо отримати окремо об’єкт Quiz, до нього, звертаючись у формі ООП можно використати усі методи та функціїї з наслідуваних класів, а також легко можна звертатись до колонок запису. Наприклад ми отримуємо об’єкт класу Quiz, і хочемо отримати його дату створення (колонка в таблиці називатимесь created_at) , буде виглядати це

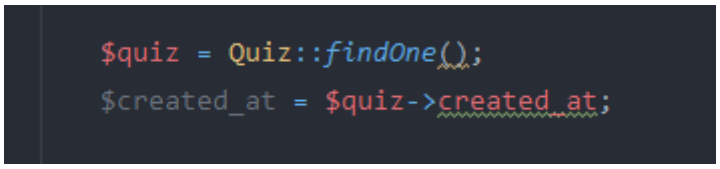

Рис. 3.14 Отримання часу створ. опитуваннння

наступним чином:

Ще однією великою перевагою моделей фреймворку – це з’вязки між таблицями. Для того щоб з’вязувати таблиці не потрібно створювати їх зв’язки у самій БД MySql. Для цього достатньо мати ідентификатор першої таблиці в колонці другої таблиці (чи навпаки), після чого можна прописати зв’язок таблиць з будь якої сторони. Наприклад, ми матимемо таблицю опитувань (назвемо її quiz – один з варіантів слова «опитування» англійскою мовою, воно коротке та зручне) і таблицю питань опитування (назвемо quiz_questions). Таблиця quiz не повинна мати в собі дані ідентифікаторів усіх своїх питань, з таблиці питань. А ось при створенні питання, воно має бути прив’язано до опитування, тому запис питання буде мати в собі ідентифікатор опитування.

Фреймворк дозволяє легко і зручно описати зв’язки. Якщо ми, маючи об’єкт опитування хочему отримати усі питання, які містять ідентифіктор цього опитування, то нам потрібно в моделі таблиці quiz створити наступну функцію

```php
public function getQuestions()
{
    return $this->hasMany(QuizQuestions::className(), ['quiz_id' => 'id'])->orderBy(['question_order'=>'SORT_ASC']);
}
```

Рис. 3.15 Зв'язок опитування з питаннями

(рис 3.15)

Таким чином, щоб отримати усі записи в таблиці питань, які відносяться до певного опитування, потрібно на об’єкт quiz викликати цю функцію (див. рис. 3.16)

Далі отримані питання можно обробити за допомогою мовної конструкції foreach, та в свою чергу для кожного питання отримати відповіді (заздалегіть прописавши зв’зки між цими таблицями) (див. рис. 3.17)

Після створення усіх таблиць та написання всіх зв’язків між ними, можна

```php
// Отримуємо об'єкт Quiz
$quiz = Quiz::findOne();
// Отриммуємо quiz_questions
$quiz_questions = $quiz->getQuestions()->all();
// Або прощений варіант, завдяки "магічному" методу
$quiz_questions = $quiz->questions;
```

Рис 3.16 Отримання питань опитувань завдяки зв’язкам моделей

```php
// Отримуємо об'єкт Quiz
$quiz = Quiz::findOne();
// Отриммуємо quiz_questions
$quiz_questions = $quiz->getQuestions()->all();
// Або прощений варіант, завдяки "магічному" методу
$quiz_questions = $quiz->questions;
foreach ($quiz_questions as $question)
{
    $answer = $question->getAnswer()->one();
    // Або
    $answer = $question->answer;
}
```

Рис. 3.17 Отримання відповідей

зобразити цю систему наступною діаграмою:

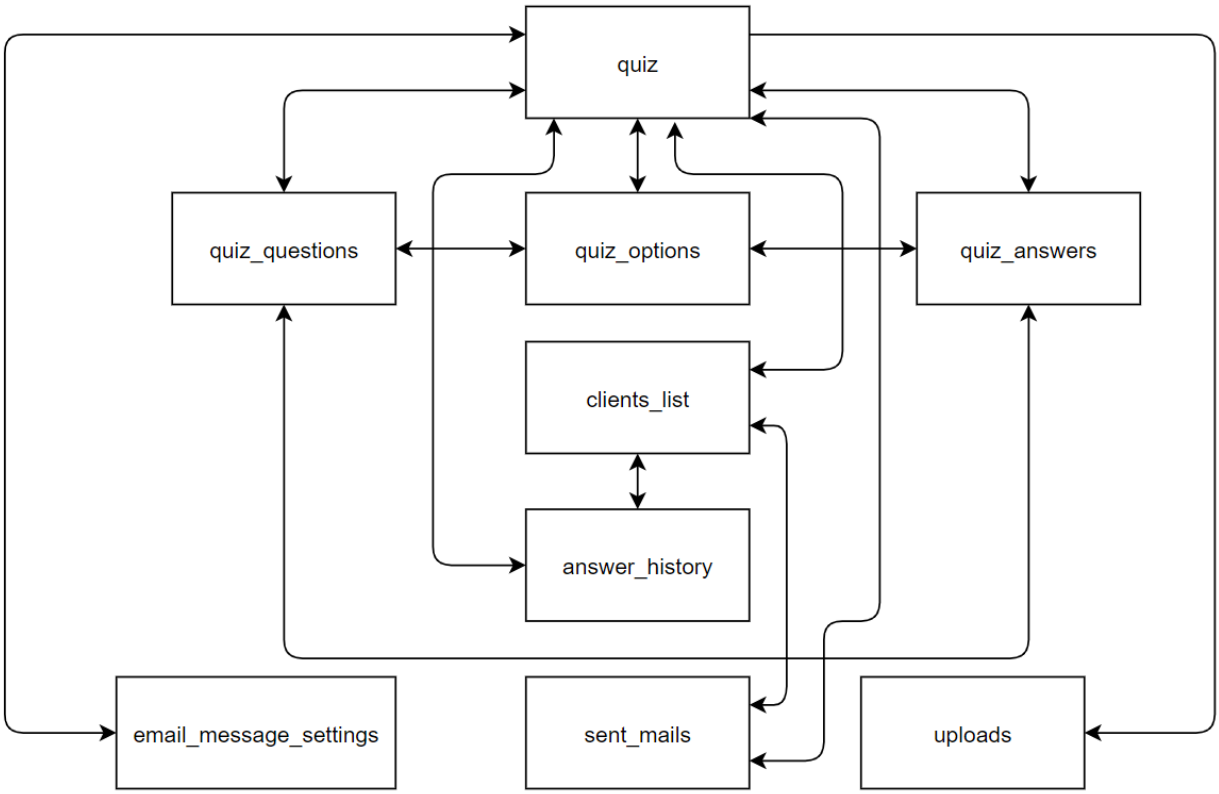

Рис 3.18 Діаграма зв’язків таблиць

3.5 Опис готового проекту

Нажаль чи на радість проект вийшов дуже об’ємним, і описувати кожну модель, кожен зв’язок, усі контроллери та екшени, усі сторінки виду і тд буде не дуже доцільно, тому у цьому пункті буде описани вже готовий функціонал проекту, а основний код проекту буде прикладено у вигляді додатків. На наступному рисунку зображена головна сторінка, на якій можна бачити список усіх опитувань, їх основні дані, та можливість перейти до налаштувань окремого опитування, а також кнопка створення нового опитування. (див рис. 3.19).

Рис. 3.19 Головна сторінка проекту

На наступному рисунку 3.21 зображені функції та сторінки для конжного

опитування. При наведенні курсором на необхідне опитування із списку, воно підсвічується і відображається кнопка, при натисканні якої відображається меню (рис. 3.20).

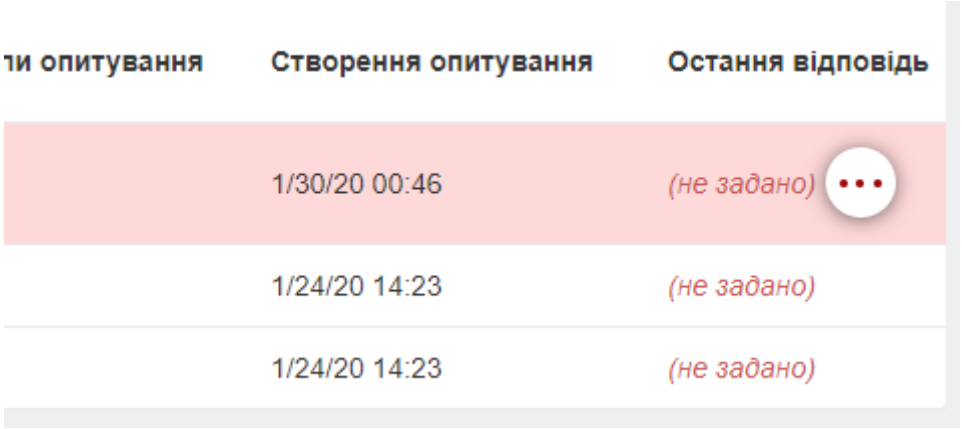

Рис. 3.20 Hover ефект

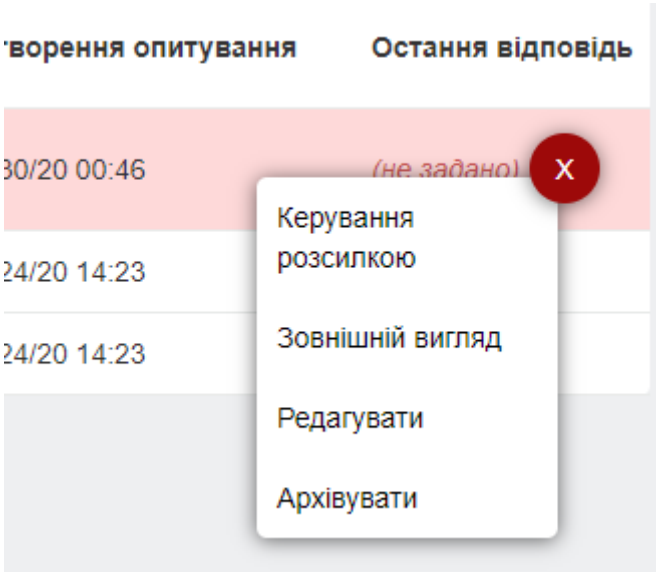

Рис. 3.21 Меню опитування

На ілюстрації вище можна бачити елементи меню опитування, такі як «керування розсилкою», «зовнішній вигляд», «редагувати», «архівувати», призначення яких розкрито у постановці завдання.

Після натискання кнопки «створити опитування» відкриється модальне вікно з формою для введеня назви опитування та (по бажанню) вибору шаблону готового опитування. (рис.3.22)

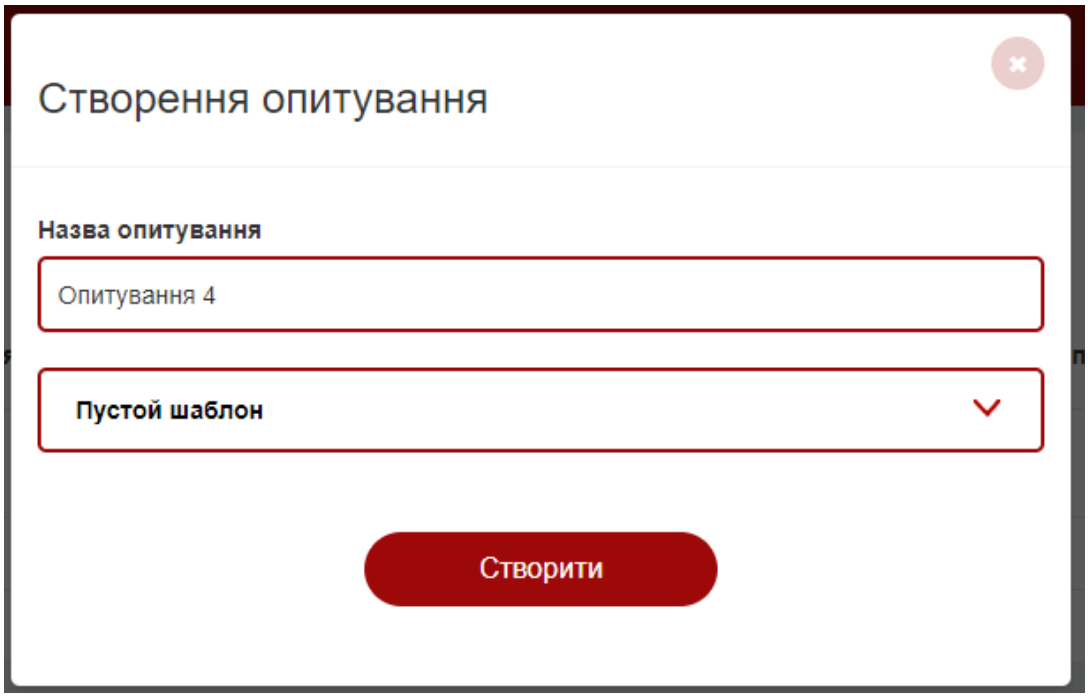

Рис. 3.22 Модальне вікно створення опитування

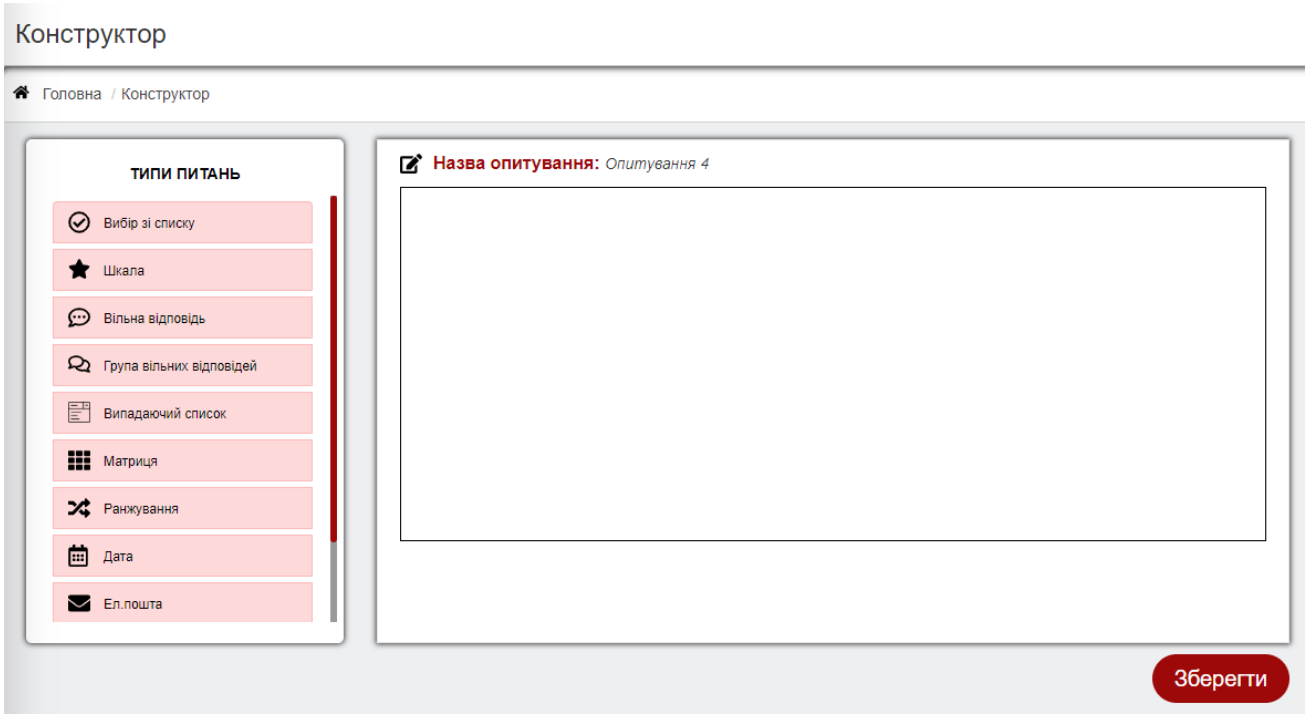

Рис. 3.23 Сторінка конструктору

Після заповнення даних і натискання «створити» ми потрапляємо на сторінку конструктору опитувань. (рис. 3.23)

Шляхом перетягування типу питання у область конструктора, в останній

створюється профіль цього питання з усіма налаштуваннями. На наступній

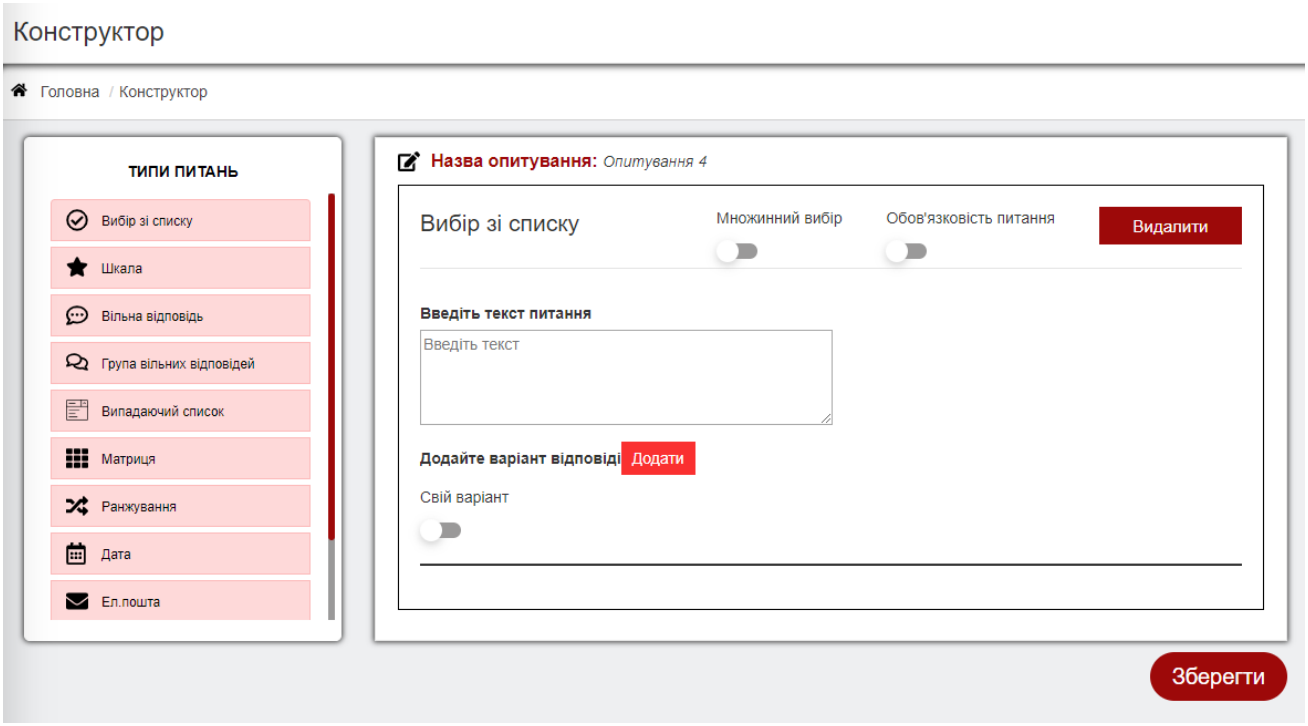

Рис. 3.24 Додавання питання до області конструювання

ілюстрації наведено приклад, у якому до області конструктора додано «Вибір зі списку».

Як видно на рис. 3.24 в налаштуванні питання виконані поставлені задачі, а також додатково було розроблено функцію «свій варіант», якщо при створені опитування респонденту хочуть надати можливість крім наданих варіантів відповідей ввести свій варіант.

Між собою питання у області конструювання переміщюються за допомогою перетягувать.

Для наочності створимо маленьке тестове опитування, де буде три питання: «Ваша стать», «Ваш вік», «Оцініть вашу задоволеність роботою в компанії».

Перше і друге питання реалізуємо завдяки типу питань «вибір із списку», а третє за допомогою питання «шкала». Варіанти відповіді для першого питання : «чоловіча», «жіноча», для другого «18-25», «25-45», «45-60», а для третього оберемо шкалу від 0 до 10. Таким чином після збереження опитування і перегляді його зовнішнього виду можемо бачити наступне (див. рис. 3.25)

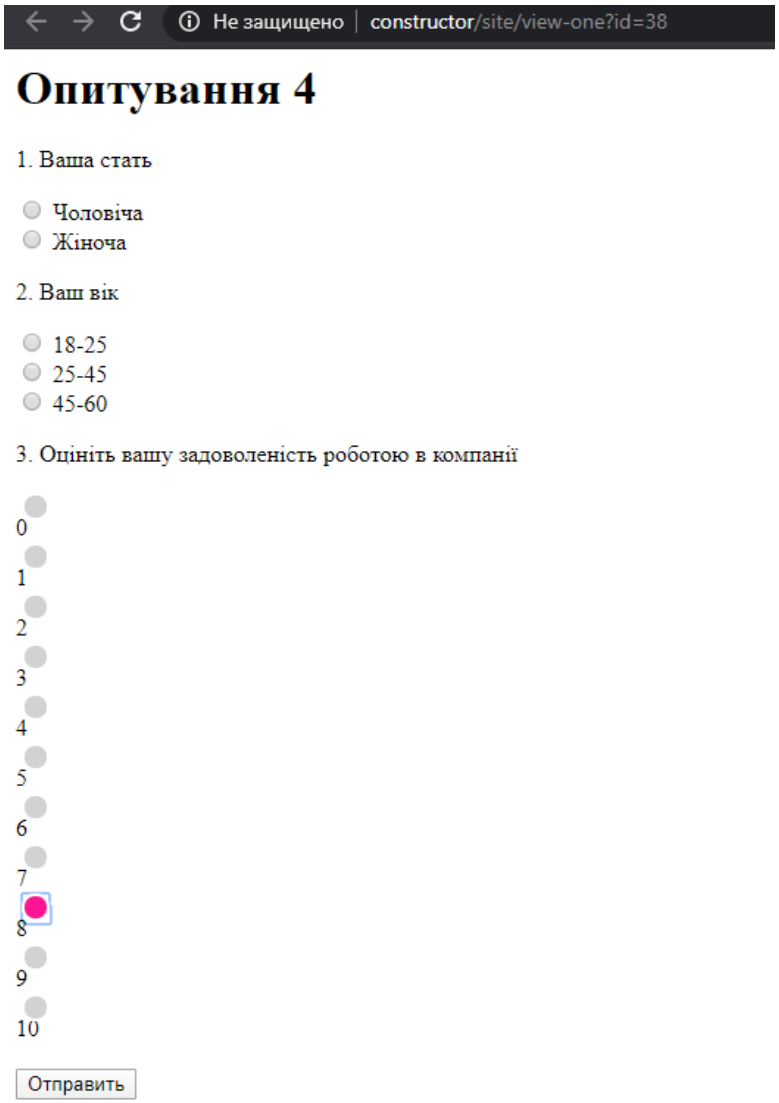

Рис. 3.25 Зовнішній вид сконструйованого опитування

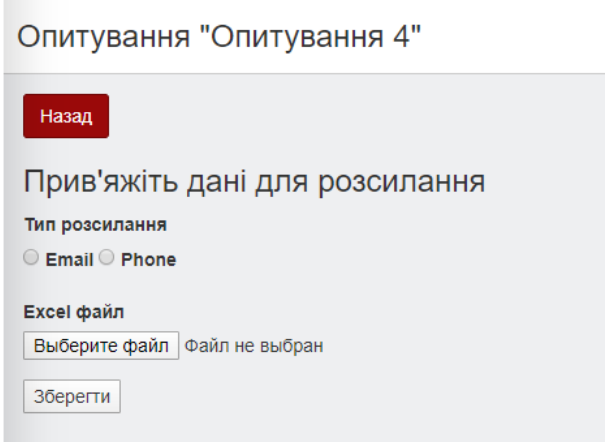

Рис. 3.26 Форма для заванатаження даних розсилки

Далі, якщо перейти у налаштування розсилкою опитування ми побачимо

наступну форму (рис. 3.26)

Оберемо тип «email» та завантажимо excel у форматі, зазначеному в постановці задачі. Після збереження сторінка керуванням розсилання оновиться і матиме наступний вигляд:

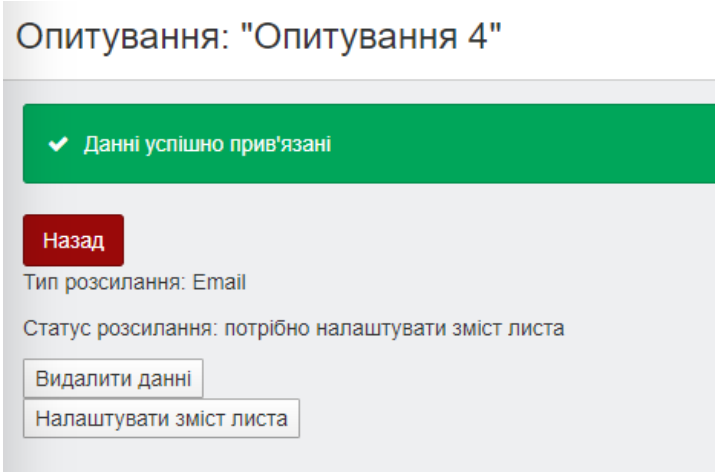

Рис. 3.27 Вигляд сторінки після завантаження даних

Як бачимо сторінка має запланований функціонал. Звісно, щоб запустити розсилання потрібно налаштувати зміст листа. Кожен лист матиме однакову структуру, а ось текст і картинку в шапці листа можна налаштовувати за потреби.

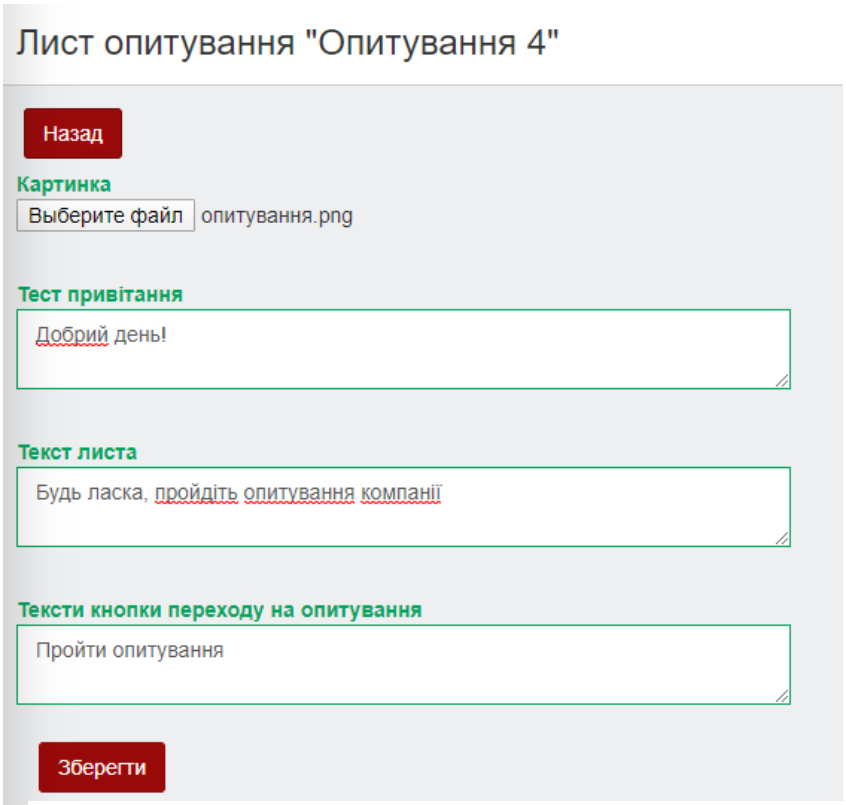

Рис. 3.28 Налаштування листа

Після цього, на сторінці налаштування активується кнопка «запустити розсилання», після натискання якої відпрацює розсилання, а також ця кнопка заміниться на «закінчити опитування».

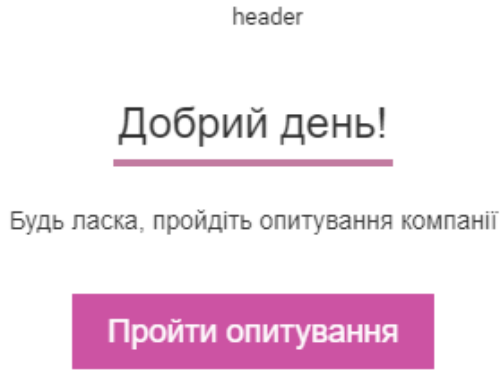

Рис. 3.29 Надійшовший лист, до кожного респондента

На рисунку 3.29 ми можемо спостерігати лист, який надійшов до кожного респондента. Нажаль картинка в шапці опитування не відображається, оскільки розробка ведеться на локальному сервері, і з мережі Інтернет до неї немає з’єднання, тому замість картинки ми можемо спостерігати надпис «header» що є описом функціональним описом цієї картинки. Перейшовши за посиланням, респондент потрапить на зовнішній вид опитування та зможе його пройти. Відправка, отримання, читання листа відстежуться. Також відстежується перехід по посиланню опитування і звісно проходження опитування. Під час тесту усі листи відправляються на мою особисту пошту, завдяки чому можна відтворити процес проходження опитування для прикладу.

Пройшовши певну кількість опитувань запустимо консольну команду, яка оновлює статуси листів. Далі закінчемо опитування, натиснувши на кнопку «закінчити опитування» на сторінці керування розсилкою. Після чого стане доступна зручна статистика в графіках, де наочно можна побачити результати опитування (див. рис. 3.30)

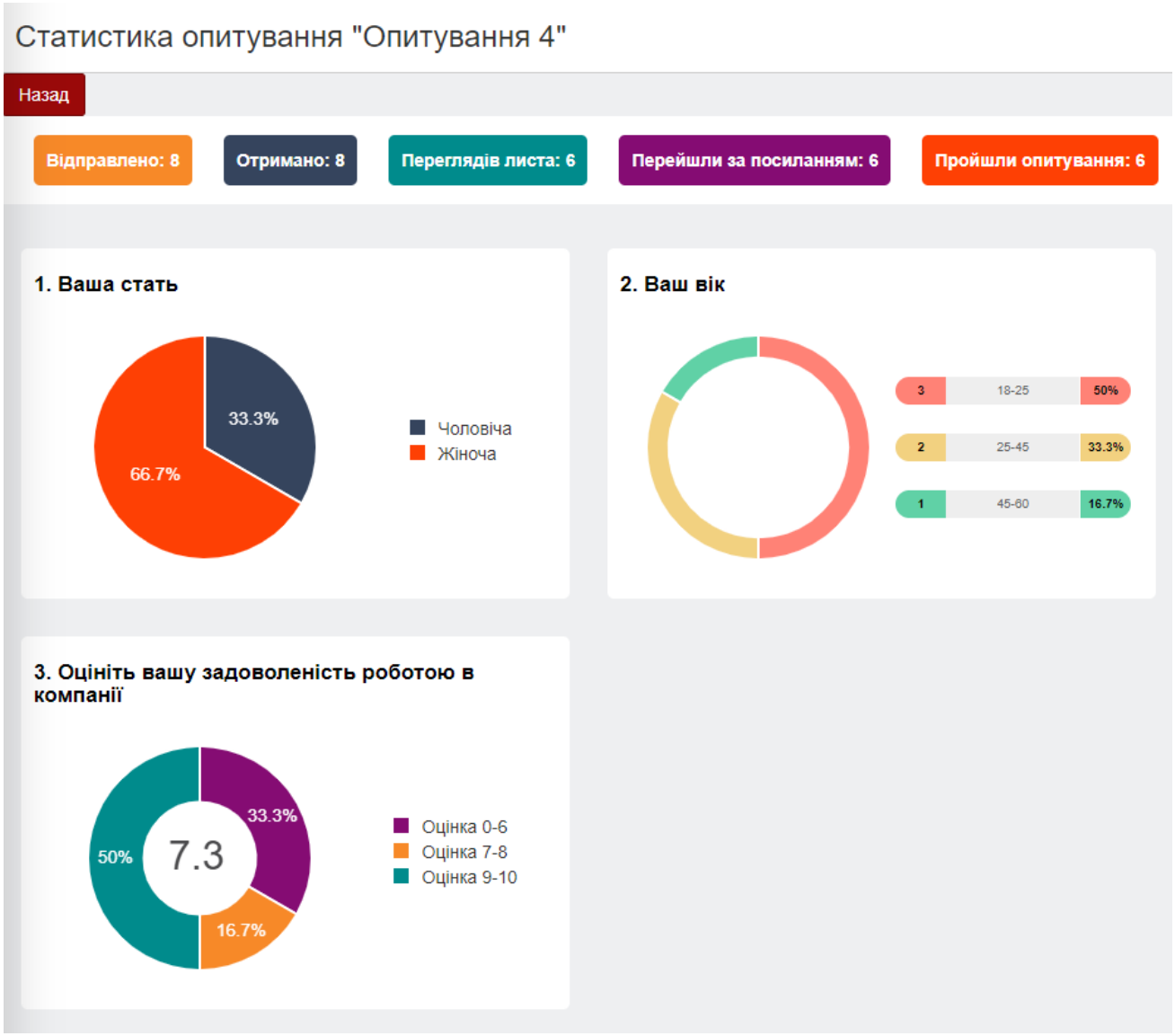

Рис. 3.30 Статистика опитування

Як видно на ілюстрації вище, статистика відображає статистику розсилки, та

статистику відповідей опитування у графіках. Кожен тип питання відображається унікальним типом графіка, так, наприклад, графік для шкали відображає середнє арифметичне число відповідей по центру.

ВИСНОВОК ДО РОЗДІЛУ 3

Цей розділ демонструє як можна реалізувати просту систему контролю задоволеності співробітниками у вигляді Web-додатку. Використовуючи якісне програмне забезпечення та правильні і зручні технології, побудова додатку не стала проблемою. Було використано велику кількість технологій. В першу чергу це мова розмітки HTML, таблиці стилей CSS та об’єтно-орієнтована мова програмування PHP, фреймворк Yii2, серверна платформа Open Server, середовище програмування phpStorm.

Web-додаток орієнтований в першу черну на керівників компаній та менеджерів, але цю систему також можна інтегрувати у різні середовища та використовувати за різних потреб, наприклад для особистого розсилання запрошення друзям. В проекті створено простий і зручний дизайн для конструктора опитувань, але нажаль за браком часу та досвіду не вдалося стилізувати зовнішній вид опитування.

ВИСНОВКИ

У ході виконання дипломного проекту були досліджені та випробувані методи створення Web-додатку системи контролю задоволеності співробітників. Проведено дослідження ефективності використання різних технологій для реалізації функціонального і готового до експлуатації Web-додатку. Для цього було проаналізовано переваги програмного забезпечення, редакторів вихідного коду, фреймворків та засобів, які допомагають полегшити розробку.

Також були розглянуті питання необхідності такої системи в наші часи, описано важливість покращення середовища для робітників та своєчасного виявлення проблем в компанії, що в свою чергу дозволить покращити виробництво підприємства.

Розглянуті та використані технології показали, що розумний підбір технологій для розробки проекту може полегшити процес розробки та навіть зробити його цікавим.

В результаті дипломної роботи було отримано працездатний web-додаток, який можна використовувати як систему контролю задоволеності співробітників так і за особистими чи корпоративними цілями. Наприклад можна робити розсилку для запрошення друзів, чи колег на захід, та запитати в них про їх вподобання і інтереси. Чи також після заходу провести опитування про його організацію, отримати побажання та рекомендації.

Незаперечно цей проект буде і надалі розвиватись, в планах розвивати готовий функціонал, додавати новий, робити можливість інтегрування проекту у інші, покращувати зовнішній вид та зручність використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. cyberleninka.ru/ [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/rol-udovletvorennosti-trudom-personala-v-reshenii-upravlencheskih-zadach-meditsinskoy-organizatsii/viewer
2. hr-director.ru [Електронний ресурс]. – Режим доступу: https://www.hr-director.ru/article/63014-otsenka-udovletvorennosti-personala-uznaem-nastroeniya-rabotnikov
3. hrliga.com [Електронний ресурс]. – Режим доступу: https://hrliga.com/index.php?module=profession&op=view&id=1676
4. lifewire.com [Електронний ресурс]. – Режим доступу: https://www.lifewire.com/what-is-a-web-application-3486637
5. habr.com [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/450282/
6. en.yeeply.com [Електронний ресурс]. – Режим доступу: https://en.yeeply.com/blog/advantages-and-disadvantages-of-web-app-development/
7. svitla.com [Електронний ресурс]. – Режим доступу: https://svitla.com/blog/web-application-architecture
8. slideshare.net [Електронний ресурс]. – Режим доступу: https://www.slideshare.net/ssusere5f319/web-66422826
9. wikipedia.org [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Доменна_система_імен
10. wikipedia.org [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/HTML
11. tutorialspoint.com [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/css/what_is_css.htm
12. wikipedia.org [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/PHP

13. web-creator.ru [Електронний ресурс]. – Режим доступу: https://web-creator.ru/articles/about_frameworks
14. habr.com [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/464417/
15. web-creator.ru [Електронний ресурс]. – Режим доступу: https://web-creator.ru/articles/yii
16. jetbrains.com [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/phpstorm/
17.habr.com [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/137388/
18. computerhope.com [Електронний ресурс]. – Режим доступу: https://www.computerhope.com/jargon/c/css.htm
19. yii2-framework.readthedocs.io [Електронний ресурс]. – Режим доступу: https://yii2-framework.readthedocs.io/en/latest/guide-ru/tutorial-advanced-app/

Додаток А

Код основного контроллеру

```php
<?php
namespace backend\controllers;

use Codeception\Module\Cli;
use common\models\AdditionalColumns;
use common\models\AnswerHistory;
use common\models\ClientsList;
use common\models\AdditionalColumnsNames;
use common\models\EmailMessageSettings;
use common\models\QuizAnswers;
use common\models\QuizOptions;
use common\models\Quiz;
use common\models\QuizQuestions;
use common\models\SentMails;
use PhpOffice\PhpSpreadsheet\IOFactory;
use stdClass;
use Yii;
use yii\helpers\ArrayHelper;
use yii\helpers\Url;
use yii\web\Controller;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use common\models\LoginForm;
use common\models\UploadForm;
use yii\web\UploadedFile;
use yii\data\ActiveDataProvider;
/**
 * Site controller
 */
class SiteController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [
                    [
                        'actions' => ['login', 'error'],
                        'allow' => true,
                    ],
                    [
                        'actions' => [
                            'logout',
                            'index',
                            'constructor',
                            'view-all',
                            'view-one',
                            'create-quiz',
                            'edit-quiz',
                            'delete-quiz',
                            'update-quiz',
                            'change-status',
```

```
                            'upload',
                            'results',
                            'details',
                            'archive-quiz',
                            'archive',
                            'email-constructor',
                            'sms-constructor',
                           'delete-quiz-mailing-data',
                           'view-uploaded-data',
                           'stats',
                           'view-free-question-answers',
                           'end-quiz',
                       ],
                       'allow' => true,
                       'roles' => ['@'],
                   ],
               ],
           ],
           'verbs' => [
               'class' => VerbFilter::className(),
               'actions' => [
                   'logout' => ['post'],
               ],
           ],
       ];
   }

   /**
    * {@inheritdoc}
    */
   public function actions()
   {
       return [
           'error' => [
               'class' => 'yii\web\ErrorAction',
           ],
       ];
   }

   /**
    * Displays homepage.
    *
    * @return string
    */
   public function actionIndex()
   {
       return $this->redirect('view-all');
   }

   /**
    * Login action.
    *
    * @return string
    */
   public function actionLogin()
   {
       if (!Yii::$app->user->isGuest) {
           return $this->goHome();
       }
```

```
        $model = new LoginForm();
        if ($model->load(Yii::$app->request->post()) && $model->login()) {
            return $this->goBack();
        } else {
            $model->password = '';

            return $this->render('login', [
                'model' => $model,
            ]);
        }
    }

    /**
     * Logout action.
     *
     * @return string
     */
    public function actionLogout()
    {
        Yii::$app->user->logout();

        return $this->goHome();
    }

    public function actionConstructor()
    {
       $template_quiz = null;
        $questions = null;
        if (isset($_POST['template']))
        {
            if ($_POST['template'] != null)
            {
                $template_quiz = Quiz::findOne($_POST['template']);
                $questions = QuizQuestions::find()->where(['quiz_id' =>
$_POST['template']])->orderBy(['question_order' => SORT_ASC])->all();
            }
        }
        return $this->render('constructor',[
            'quiz' => $template_quiz,
            'questions' => $questions,
        ]);
    }


    public function actionCreateQuiz()
    {
//        echo '<pre>';
//        var_dump($_POST);die;
        $countOrder = 0;
        if ($post = Yii::$app->request->post()) {
            foreach ($post as $name => $value)
            {
                if (stristr($name, 'quizName')) {
                    $quiz = new Quiz;
                    $quiz->name = trim($value) ?? '';
                    $quiz->unique_id = rand();
                    $quiz->status = Quiz::STATUS_ACTIVE;
                    $quiz->company_name = ArrayHelper::getValue($post, 'company_name')
?? '';
                    $quiz->type = ArrayHelper::getValue($post, 'type') ?? '';
```

```
                    $quiz->created_at = time();
                    $quiz->save();
                } elseif (stristr($name, 'question')) {
                    $countOrder += 1;
                    $question_name_cut = str_replace('question_', '', $name);
                    $question_number_of_id = preg_replace("/[^0-9]/", "", $name);
                    $question = new QuizQuestions();
                    $question->text = trim($value);
                    $question->quiz_id = ArrayHelper::getValue(Quiz::find()-
>where(['unique_id' => $quiz->unique_id])->one(), 'id');
                    $question->type = preg_replace('/\PL/u', '', $question_name_cut);
                    $question->question_order = $countOrder;
                    $question->status = ArrayHelper::getValue($post, 'checker_' .
$question_number_of_id);
                    $question->setting = ArrayHelper::getValue($post, 'setting_' .
$question_number_of_id);
                   $question->setting_1 = ArrayHelper::getValue($post, 'setting1_' .
$question_number_of_id);
                   $question->save();
                } elseif (stristr($name, 'option')) {
                    $option = new QuizOptions();
                    $option->value = trim($value);
                    $option->question_id = ArrayHelper::getValue(QuizQuestions::find()-
>where(['quiz_id' => $question->quiz_id])
                                                                                    -
>andWhere(['question_order' => $question->question_order])->one(), 'id');
                    $option->status = ArrayHelper::getValue($_POST, 'checker_'
.str_replace('option_','',$name));
                    if (stristr($name, 'vertical')){$option->status = "column";}
                    $option->save();
                }

            }
        }
        Yii::$app->session->setFlash('success', 'Опрос создан');
        return $this->redirect('view-all');
    }


    public function actionViewAll()
    {
       $model = Quiz::find()->where(['IN','status',['inactive', 'active']])-
>orderBy(['created_at' => SORT_DESC])->all();
        return $this->render('view-all', ['model' => $model]);
    }

    public function actionViewOne($id)
    {
       $upload_form = new UploadForm();
       $quiz = Quiz::findOne($id);
       $data_uploaded = false;
       if (ClientsList::find()->where(['quiz_id' => $id])->one()) {$data_uploaded =
true;}
        return $this->render('view-one', compact('quiz', 'upload_form',
'data_uploaded'));
    }

    public function actionEditQuiz($id)
    {
       $quiz = Quiz::findOne($id);
```

```
        $questions = QuizQuestions::find()->where(['quiz_id' => $id])-
>orderBy(['question_order' => SORT_ASC])->all();
        return $this->render('constructor',[
            'quiz' => $quiz,
            'questions' => $questions,
            'edit' => 'true'
        ]);
    }

    public function actionArchive()
    {
         $model = Quiz::find()-
>select(['id','name','company_name','status','type','created_at'])->where(['status'=>
'archived'])->orderBy(['created_at' => SORT_DESC])->all();
         return $this->render('archive', ['model' => $model]);
    }

    public function actionArchiveQuiz($id)
    {
        $quiz = Quiz::findOne($id);
        if ($quiz->status != Quiz::STATUS_ARCHIVED)
        {
            $quiz->status = Quiz::STATUS_ARCHIVED;
            $quiz->update();
            Yii::$app->session->setFlash( 'success', 'Опрос успешно архивирован' );
            return $this->redirect('view-all');
        }
        else
        {
            $quiz->status = ($quiz->mailing_status != 'inactive') ? Quiz::STATUS_ACTIVE :
Quiz::STATUS_INACTIVE;
            $quiz->update();
            Yii::$app->session->setFlash( 'success', 'Опрос успешно разархивирован' );
            return $this->redirect('archive');
        }
    }

    public function actionDeleteQuiz($id)
    {

        // удаление вопросов опроса и их вариантов ответов(настроек)
         $questions = QuizQuestions::find()->where(['quiz_id' => $id])->all();
         foreach ($questions as $question)
         {
            // если есть вопросы с вставкой файла, удаляем файлы
            if($question->type == 'UploadFile')
            {
               foreach ($question->answers ?? array() as $answer)
               {
                  unlink($answer->answer);
               }
            }
             QuizOptions::deleteAll(['question_id' => $question->id]);
         }
        QuizQuestions::deleteAll(['quiz_id' => $id]);
        // удаление истории ответов и самих ответов
        $answer_histories = AnswerHistory::find()->where(['quiz_id' =>$id])->all();

         foreach ($answer_histories as $answer_history)
         {
```

```
            QuizAnswers::deleteAll(['history_id' => $answer_history->id]);
        }
       AnswerHistory::deleteAll(['quiz_id' => $id]);

       // удаление данных рассылки и данных отправленых писем, если они есть
       $quiz = Quiz::findOne($id);
       if($quiz->upload_id != null)
        {
           ClientsList::deleteAll(['upload_id'=>$quiz->upload_id]);
           SentMails::deleteAll(['quiz_id' => $quiz->id]);
           AdditionalColumnsNames::deleteAll(['upload_id'=>$quiz->upload_id]);
           AdditionalColumns::deleteAll(['upload_id'=>$quiz->upload_id]);
           if($upload = UploadForm::find()->where(['upload_id' => $quiz->upload_id])-
>one())
           {
              unlink($upload->file_path);
              $upload->delete();
           }

        }
      // удаление настроек письма
       if ($email_settings = $quiz->emailMessageSettings)
       {
          if ($email_settings->image != null )
          {
             $path = Yii::getAlias( '@webroot' ) . "/uploads/";
            unlink($path.$email_settings->image);
          }

         $email_settings->delete();
       }


       // удаление опроса
       $quiz->delete();

        Yii::$app->session->setFlash( 'success', 'Опрос успешно удалён' );
        return $this->redirect('archive');
    }

    public function actionUpdateQuiz($id)
    {
//              echo '<pre>';
//        var_dump($_POST);die;
        if (Yii::$app->request->post()) {
            $questions = QuizQuestions::find()->select(['id'])->where(['quiz_id' =>
$id])->all();
            foreach ($questions as $question)
            {
                QuizOptions::deleteAll(['question_id' => $question->id]);
            }
            QuizQuestions::deleteAll(['quiz_id' => $id]);


            $post = $_POST;
//            var_dump($post);die;
            $countOrder = 0;
            foreach ($post as $name => $value)
            {
                if (stristr($name, 'quizName'))
```

```
                {
                    Yii::$app->db->createCommand()->update('quiz', [
                        'name' => trim($value),
                        'unique_id' => rand(),
                        'company_name' => ArrayHelper::getValue($post, 'company_name'),
                        'type' => ArrayHelper::getValue($post, 'type'),
                    ], 'id = '.$id)->execute();
                }

                elseif (stristr($name, 'question'))
                {
                    $countOrder += 1;
                    $question_name_cut = str_replace('question_','', $name);
                    $question_number_of_id = preg_replace("/[^0-9]/", "", $name);
                    $question = new QuizQuestions();
                    $question->text = trim($value);
                    $question->quiz_id = $id;
                    $question->type = preg_replace('/\PL/u', '', $question_name_cut);
                    $question->question_order = $countOrder;
                    $question->status = ArrayHelper::getValue($post, 'checker_' .
$question_number_of_id);
                    $question->setting = ArrayHelper::getValue($post, 'setting_' .
$question_number_of_id);
                   $question->setting_1 = ArrayHelper::getValue($post, 'setting1_' .
$question_number_of_id);
                    $question->save();
                }
                elseif (stristr($name, 'option'))
                {
                    $option = new QuizOptions();
                    $option->value = trim($value);
                    $option->question_id = ArrayHelper::getValue(QuizQuestions::find()-
>where(['quiz_id' => $question->quiz_id])
                                                                                    -
>andWhere(['question_order' => $question->question_order])->one(), 'id');
                   $option->status = ArrayHelper::getValue($_POST, 'checker_'
.str_replace('option_','',$name));
                    if (stristr($name, 'vertical')){$option->status = "column";}
                    $option->save();
                }
            }
            Yii::$app->session->setFlash( 'success', 'Опрос успешно обновлен' );
            return $this->redirect('view-all');
        }
    }

    public function actionChangeStatus($id)
    {
        $quiz = Quiz::findOne($id);

        if ($quiz->status == Quiz::STATUS_ACTIVE)
        {
            $quiz->status = Quiz::STATUS_INACTIVE;
            $quiz->save();
            Yii::$app->session->setFlash('success', 'Опрос успешно остановлен!');
            return $this->redirect('view-all');

        }
        else
        {
```

```
//            if ($quiz->upload_id == null)
//            {
//                Yii::$app->session->setFlash('danger', '
//                    Для запуска необходимо привязать данные для рассылки!
//                    <a href="/admin/site/view-one?id='.$id.'">Перейти в профиль
опроса</a>
//                ');
//                return $this->redirect('view-all');
//            }
            $quiz->status = Quiz::STATUS_ACTIVE;
            $quiz->save();
            Yii::$app->session->setFlash('success', 'Опрос успешно запущен!');
            return $this->redirect('view-all');
        }
    }

    public function actionEndQuiz($quiz_id)
    {
       $quiz = Quiz::findOne($quiz_id);
       $quiz->status = Quiz::STATUS_INACTIVE;
       $quiz->mailing_status = Quiz::MAILING_STATUS_INACTIVE;
       $quiz->update();
       Yii::$app->session->setFlash('success', 'Опрос успешно закончен!');
       return $this->redirect(['view-one', 'id' => $quiz_id]);

    }


    public function actionUpload($quiz_id)
    {
        if (Yii::$app->request->isPost) {
            $quiz = Quiz::findOne($quiz_id);
           $upload_id = rand();
           $upload = new UploadForm;
           $upload->excelFile = UploadedFile::getInstance($upload, 'excelFile');
           $upload->created_at = time();
           $upload->upload_id = $upload_id;
           $upload->mailingType = $_POST['UploadForm']['mailingType'];
            if ($upload->upload() && $upload->save(false)) {
                $spreadsheet = IOFactory::load($upload->file_path);
                $sheetData = $spreadsheet->getActiveSheet()->toArray(null, true, true,
true);
                foreach ($sheetData as $sheet_row) {
                    $mailing_id = rand();
                    if (!isset($i))
                    {
                       if(!UploadForm::validateValue($sheet_row))
                            {
                               unlink($upload->file_path);
                               $upload->delete();
                                return $this->redirect('view-one?id='.$quiz_id);
                            }
                        foreach ($sheet_row as $item_key => $item_value)
                        {
                           if ($item_value == null){break;}
                            if (preg_match('/[M-Z]/', $item_key))
                            {
                                $additional_column_name = new AdditionalColumnsNames();
                                $additional_column_name->column_letter = $item_key;
                                $additional_column_name->name = $item_value;
```

```
                                $additional_column_name->upload_id = $upload_id;
                                $additional_column_name->save();
                            }
                        }
                        $i = 1;
                    }
                    else {
                        if ($sheet_row['A'] == null)
                        {
                            break;
                        }
                        $client_row                = new ClientsList();
                        $client_row->client_row_id = (string) $sheet_row['A'];
                        $client_row->name          = $sheet_row['B'];
                        $client_row->last_name     = $sheet_row['C'];
                        $client_row->father_name   = $sheet_row['D'];
                        $client_row->phone         = (string) $sheet_row['E'];
                        $client_row->email         = $sheet_row['F'];
                        $client_row->gender        = $sheet_row['G'];
                        $client_row->mailing_type  = $sheet_row['H'];
                        $client_row->t_office      = $sheet_row['I'];
                        $client_row->d_office      = $sheet_row['J'];
                        $client_row->macro_region  = $sheet_row['K'];
                        $client_row->worker_RB     = (string) $sheet_row['L'];
                        $client_row->mailing_id    = $mailing_id;
                        $client_row->upload_id     = $upload_id;
                        $client_row->quiz_id       = $quiz_id;
                        if ( $client_row->save() ) {
                            foreach ( range( 'M', 'Z' ) as $char ) {
                                if ( isset( $sheet_row[ $char ] ) && $sheet_row[ $char ] !=
null ) {
                                    $additional_column_row                = new
AdditionalColumns();
                                    $additional_column_row->column_name_id =
AdditionalColumnsNames::find()->select( 'id' )->where( [
                                            'column_letter' => $char,
                                            'upload_id'     => $upload_id ])->one()->id;
                                    $additional_column_row->data           = $sheet_row[
$char ];
                                    $additional_column_row->client_id      = $client_row-
>id;
                                    $additional_column_row->upload_id      = $upload_id;
                                    $additional_column_row->save();
                                } else {
                                    break;
                                }
                            }
                        }
                    }
                }
                $quiz = Quiz::findOne($quiz_id);
                $quiz->upload_id = $upload_id;
               $quiz->mailing_type = $upload->mailingType;
               $quiz->save();
                Yii::$app->session->setFlash( 'success', 'Данные успешно привязаны! <a
href=""></a>' );
                return $this->redirect('view-one?id='.$quiz_id);
            }
            else
            {
```

```
                Yii::$app->session->setFlash( 'error', 'Произошла ошибка! <a
href=""></a>' );
                return $this->redirect('view-one?id='.$quiz_id);
            }
        }
    }

    public function actionDeleteQuizMailingData($quiz_id)
    {
       $quiz = Quiz::findOne($quiz_id);
       $upload_id = $quiz->upload_id;
       $quiz->upload_id = null;
       $quiz->mailing_type = null;
       $quiz->update();
       ClientsList::deleteAll(['upload_id' => $upload_id]);
       AdditionalColumns::deleteAll(['upload_id' => $upload_id]);
       AdditionalColumnsNames::deleteAll(['upload_id' => $upload_id]);
      $upload = UploadForm::find()->where(['upload_id' => $upload_id])->one();
      unlink($upload->file_path);
       $upload->delete();
       Yii::$app->session->setFlash( 'success', 'Данные удалены! <a href=""></a>' );
       return $this->redirect('view-one?id='.$quiz_id);
    }

    public function actionViewUploadedData($upload_id)
    {
       $query = ClientsList::find()->where(['upload_id' => $upload_id])-
>orderBy(['id'=>'SORT DESC']);
       $quiz = Quiz::find()->select(['name','id'])->where(['upload_id' => $upload_id])-
>one();
       $columns = ClientsList::getColumnsForGrid($upload_id);
       $data_provider = new ActiveDataProvider([
          'query' => $query,
          'pagination' => [
             'pageSize' => 10,
          ],
       ]);
       return $this->render('view-uploaded-data',compact('data_provider', 'columns',
'quiz'));
    }

    public function actionResults($id)
    {
        $results = AnswerHistory::find()->where(['quiz_id' => $id])-
>orderBy(['created_at' => SORT_DESC])->all();
        return $this->render('results', ['results' => $results]);
    }

    public function actionDetails($id)
    {
        $model = ClientsList::find()->where(['id' => $id])->one();

       $answers_exist = AnswerHistory::find()->where(['client_id' => $model->id])->one()
? true : false;
        return $this->render('details', compact('model','answers_exist'));
    }

    public function actionEmailConstructor($quiz_id)
    {
       $trigger = false;
```

```
        $quiz = Quiz::findOne($quiz_id);
        if ($quiz->emailMessageSettings)
        {
           $model = $quiz->emailMessageSettings;
           $image = $model->image;
           $trigger = true;
        }
        else{
           $model = new EmailMessageSettings();
           $model->header_text = 'Здравствуйте, {{Фамилия}} {{Имя}} {{Отчество}}!';
           $model->body_text = 'Пройдите пожалуйста опрос!';
           $model->button_text = 'Пройти опрос';
        }
        if ($model->load(Yii::$app->request->post()))
        {
          if ($_FILES['EmailMessageSettings']['name']['image'] != null)
          {
             $image = UploadedFile::getInstance($model, 'image');
             $path = Yii::getAlias( '@webroot' ) . "/uploads/";
             if ( ! is_dir( $path ) ) {
                $old_umask = umask( 0 );
                mkdir( $path, 0777, true );
             }
             $model->image = $path .'email_head_image_'.$quiz_id . '.' . $image-
>extension;
             @unlink($model->image);
             $image->saveAs($model->image);
             $model->image = 'email_head_image_'.$quiz_id . '.' . $image->extension;
          }
          else {$model->image = $image;}
           $model->quiz_id = $quiz_id;
          if ($trigger) {$model->update();}
          else{$model->save(false);}
           Yii::$app->session->setFlash( 'success', 'Письмо сохранено!' );
           return $this->redirect('view-one?id='.$quiz_id);
        }
        return $this->render('email-constructor', compact('quiz', 'model'));
    }


    public function actionSmsConstructor($quiz_id)
    {
       $quiz = Quiz::findOne($quiz_id);
        return $this->render('sms-constructor', compact('quiz'));
    }

   public function actionStats($quiz_id)
   {
      $quiz = Quiz::findOne($quiz_id);
      $question_array = [];
      foreach ($quiz->statsQuestions as $question)
      {
         if($question->type == 'Scale')
         {
            if ($question->getOptions()->one()->value == '0-10' || $question-
>getOptions()->one()->value == '0-10(NPS)')
            {
               $question_array += array('0-10' => $question);
            }
            elseif($question->getOptions()->one()->value == '1-5')
```

```
            {
                $question_array += array('1-5' => $question);
            }
        }
        elseif (($question->type == 'Single' || $question->type == 'Drop') &&
$question->setting != 'multiple' && $question->getOptions()->count() == 2)
        {
            $question_array += array('two_answers' => $question);
        }
        elseif (($question->type == 'Single' || $question->type == 'Drop') &&
$question->setting != 'multiple' && $question->getOptions()->count() > 2
                && $question->maxAnswerLength() > 24)
        {
            $question_array += array('three_and_more_answers_long' => $question);
        }
        elseif (($question->type == 'Single' || $question->type == 'Drop') &&
$question->setting != 'multiple' && $question->getOptions()->count() > 2)
        {
            $question_array += array('three_and_more_answers' => $question);
        }
        elseif ($question->setting == 'multiple')
        {
            $question_array += array('multiple' => $question);
        }
        elseif ($question->type == 'Free')
        {
            $question_array += array('free' => $question);
        }
    }
    return $this->render('stats', compact('quiz', 'question_array'));
}

public function actionViewFreeQuestionAnswers($question_id)
{
    $question = QuizQuestions::findOne($question_id);
    return $this->render('view-free-question-answers', compact('question'));

}

}
```