ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка веб-системи індикаторів якості роботи працівників»

(тема кваліфікаційної роботи українською мовою)

«Development of the employees work quality indicators web system»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
Нікітін Нікіта Олексійович
(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Штефан Н.З.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент Перелигін Б.В.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри
(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 21 від 20 червня 2024 р.

Оцінка відмінно / A / 90 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

СКБД – система керування базами даних

CSS – Cascading Style Sheets – каскадні таблиці стилів.

HTML – HyperText Markup Language – мова гіпертекстової розмітки.

MySQL – вільна система управління базами даних (СУБД).

PHP – Hypertext Preprocessor – препроцесор гіпертекста.

ODBC – OpenDatabaseConnectivityStandard – стандарт підключення до відкритих баз даних.

SADT – Structured Analysis and Design Technique – методологія структурного аналізу і проектування.

UML – Unified Modeling Language – мова візуального моделювання.

URL – Uniform Resource Identifier – уніфікований ідентифікатор ресурсу.

ВСТУП

У сучасному світі стрімких технологічних змін та інтенсивної конкуренції на ринку праці, ефективне управління персоналом набуває все більшого значення. Від якісної роботи кожного працівника залежить успіх всієї організації, тому важливо мати надійні інструменти для оцінки та покращення продуктивності працівників. Одним з таких інструментів є індикатори якості.

Індикатори якості роботи – це конкретні метрики або параметри, за допомогою яких вимірюється ефективність та результативність працівника, команди або організації в цілому. Ці індикатори дозволяють керівництву оцінювати, наскільки добре виконується робота, визначати досягнення цілей та виявляти можливості для покращення. Індикатори якості можуть бути різними в залежності від конкретної галузі, виду діяльності або цілей організації.

Розробка та впровадження електронних систем для оцінки якості роботи персоналу дозволяє автоматизувати процеси збору, аналізу та інтерпретації даних, що значно підвищує об’єктивність та точність оцінок. Використання таких систем допомагає визначати сильні та слабкі сторони працівників, а також оптимізувати робочі процеси всередині організації.

Метою дипломної роботи є проектування та розробка електронної системи оцінки якості роботи працівників, яка б забезпечувала керівництво компаній та організацій ефективними інструментами для моніторингу та оцінки кості роботи персоналу.

Об’єктом розробки є веб-додаток як цілісна система, що включає серверну частину, бази даних, інтерфейс користувача, та засоби інтеграції з іншими системами, а також компоненти, які відповідають за різні функції, такі як збір даних, аналіз, створення звітів, управління користувачами тощо.

Предметом розробки є конкретні елементи та функції системи, що забезпечують роботу з індикаторами якості: конкретні метрики або параметри,

за допомогою яких вимірюється ефективність та результативність працівників, а також механізми та методи, що використовуються для обробки та аналізу зібраних даних.

Ця тема є актуальною в контексті сучасних тенденцій у сфері управління персоналом, де все більше компаній прагнуть до цифрової трансформації своїх бізнес-процесів. Впровадження електронної системи індикаторів якості роботи працівників сприятиме підвищенню ефективності роботи організацій, покращенню корпоративної культури та забезпеченню стійкого розвитку підприємств.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Опис предметної області

«Електронна система індикаторів якості» – це веб-додаток, який проводить підрахунок індикаторів працівника та виводить фінальне значення якості роботи. Даний проект реалізовує функціонал зберігання даних, стосовно показників роботи, їх обробку, підрахунок індикаторів, автоматичну генерацію Excel-таблиці з даними, а також функціонал для роботи із персоналом і налаштування індикаторів для різних бізнес-цілей, оновлення інформації про них та видалення [1].

Основний функціонал програми можна розділити на три основних частини:

– реєстрація підприємства, налаштування потрібних індикаторів та заповнення даних про персонал;

– введення необроблених даних о роботі працівників у таблиці створених індикаторів якості;

– автоматична генерація звіту у форматі Excel-таблиці, з обробленими за вказаними налаштуваннями даними, готовими для аналізу якості роботи.

Ідеєю для реалізацій даного проекту була діджиталізація та централізація процесу збору інформації щодо якості роботи працівників на підприємстві [1].

Даному веб-додатку властиво успішно використовуватись в медичній сфері, де він, на даний момент і використовується і для якої він і розроблявся. Це обумовлено тим, що якість роботи лікаря напряму та суворо залежить від індикаторів якості, яких необхідно притримуватись.

Але, варто зазначити, що розроблені алгоритми можуть бути підлаштовані під інші сфери діяльності в яких проводиться оцінка ефективності роботи працівників, шляхом змінення індикаторів, налаштування параметрів, їх оцінки та підсумовування. Реалізація проведення

операцій над показниками дозволять легко управляти списком персоналу та гнучко налаштовувати потрібні індикатори та методи підрахунку значень якості роботи.

Даний веб-додаток дає змогу відмовитись від паперової волокити у вигляді цілих стопок звітів про роботу працівників, які надходять адміністрації від уповноважених різних відділень. Саме «електронна система індикаторів якості» надає змогу централізовано збирати потрібну інформацію то проводити незалежну її оцінку виводячи єдиний звіт у зручному вигляді.

1.2 Порівняння аналогів

Першим аналогом є ряд рішень, які базовані на створенні підрахунку індикаторів якості у програмі Microsoft Excel (Рис. 1.1).

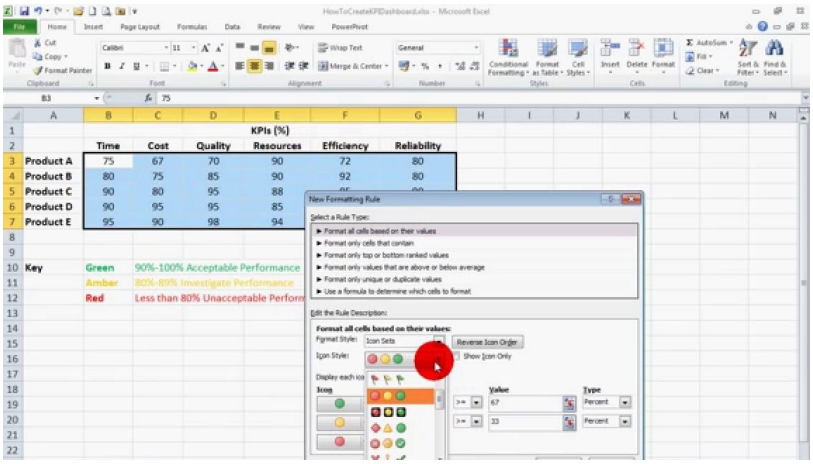

Рисунок 1.1 – Приклад розробки таблиці індикаторів якості у Microsoft Excel

Основними принципами у розробці рішень у даній програмі є написання скриптів та формул, які матимуть змогу обраховувати введені дані та виводити інформацію щодо якості роботи працівників. Але усі ці рішення мають свої недоліки, а саме:

– обмежені можливості обробки даних: Excel може бути неефективним для обробки великих обсягів даних. Це особливо актуально, якщо мова йде про

роботу з розгалуженою струкутурою закладу або з великими обсягами даних для аналізу;

– відсутність автоматизації: у Excel обмежені можливості автоматизації. Хоча можна створити скрипти для автоматизації деяких завдань, це може виявитися складним для реалізації та підтримки;

– вразливість до помилок: при великій кількості формул і даних, Excel може бути схильним до помилок, особливо якщо працює більше ніж один користувач. Навіть невеликі помилки у введенні даних чи формулах можуть призвести до недостовірних результатів аналізу якості роботи;

– обмежена спільна робота: Excel може бути не найкращим інструментом для спільної роботи над даними. Коли багато людей працюють з одним файлом Excel, можуть виникати проблеми зі збереженням цілісності даних та синхронізацією змін.

Отож, хоча Excel може бути зручним інструментом для початкового створення таблиць індикаторів даних через свою доступність та знайомство багатьох користувачів з ним, він може виявитися не найкращим вибором для великих та складних проектів або для довгострокового використання.

Також, ще одним аналогом у веденні та підрахунку індикаторів якості працівників є програмне забезпечення Smartsheet (рис. 1.2). Smartsheet – це програма для управління проектами та задачами, яка також має функціонал для створення та управління якістю роботи. В ній можна створювати таблиці для ведення індикаторів якості роботи та розподіляти індикатори [2].

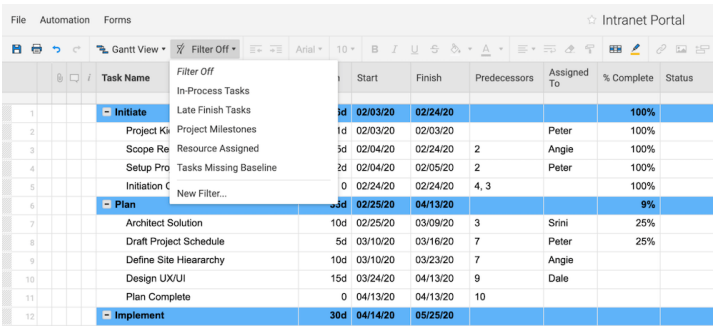

Рисунок 1.2 – Інтерфейс програми Smartsheet

Але використання Smartsheet для ведення індикаторів якості роботи працівників може мати деякі недоліки, зокрема:

– необхідність додаткових дій для відображення структури індикаторів: у складних системах індикаторів якості може бути важко відстежувати критерії оцінки та параметри у Smartsheet, що може вимагати додаткових дій для вирішення цього;

– обмежена можливість автоматизації: хоча Smartsheet має деякі можливості автоматизації за допомогою макросів та спеціальних формул, він може бути менш гнучким у цьому плані;

– вартість ліцензії та обмеження на кількість користувачів: використання Smartsheet може вимагати підписки на платну версію, особливо якщо потрібно більше функцій. Крім того, вартість може залежати від кількості користувачів, що може зробити його дорожчим для більших команд;

– можливість втрати даних через помилки вручну введених даних: якщо дані вводяться вручну, існує ризик введення помилкових даних або втрати даних через неправильне форматування, що може призвести до неточностей у веденні індикаторів якості роботи.

Ще одним аналогом є програма Quip Spreadsheets (рис. 1.3), яка є частиною платформи Quip (платформа офісних додатків). Дане програмне забезпечення надає можливість створювати та редагувати таблиці в реальному часі. Quip Spreadsheets дозволяє кільком користувачам працювати з однією таблицею одночасно та інтегрувати інші інструменти Quip [3]. Розробники даного програмного забезпечення позиціонують його як «командне» та відмічають цю особливість, як перевагу. Не дивлячись на це, Quip Spreadsheets має ряд недоліків, які є суттєвими для використання цієї програми з ціллю ведення індикаторів якості, а саме:

– обмеженість функціональності: наявність відносно малого набору формул, що може стати перепоною для проведення складних обчислень;

– швидкодія: При обробці великих обсягів даних чи складних формул Quip Spreadsheets може працювати повільно;

– обмежена можливість налаштування: Quip може мати обмежені можливості налаштування відображення та форматування даних;

– можливість втрати даних через помилки введення даних вручну: при роботі у команді існує ризик введення некоректних даних вручну, що може призвести до неточностей у веденні індикаторів якості роботи.

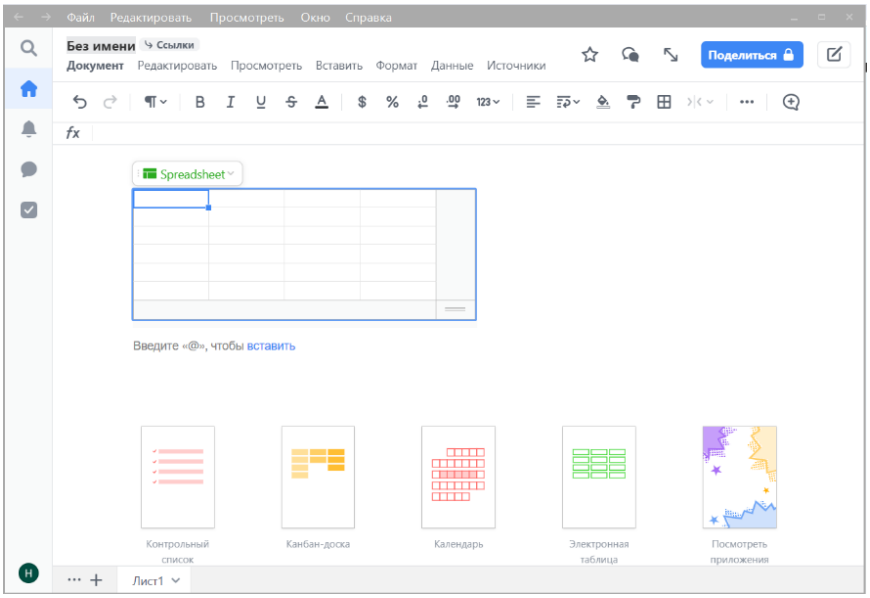

Рисунок 1.3 - Інтерфейс програми Quip Spreadsheets

Також, було проведено аналіз попередньої версії веб-додатку «Електронна система індикаторів якості» (рис. 1.4) та визначено недоліки, які потрібно доопрацювати для більш ефективного використання функціоналу. Такими недоліками є:

– обмежені можливості у розробці нових модулів: через використання застарілої архітектури, яка не дає можливість з легкістю розвивати кодову базу, стає складно розробляти нові модулі;

– обмежена гнучкість налаштування параметрів підрахунку: різні компанії матимуть різні види індикаторів у своєму пріоритеті, отож була виявлена потреба у розширені опцій по підрахунку індикаторів якості;

– недостатньо оптимізована адаптивність: брак адаптивності розроблених сторінок не дає змогу з комфортом використовувати веб-додаток на девайсах з малими екранами.

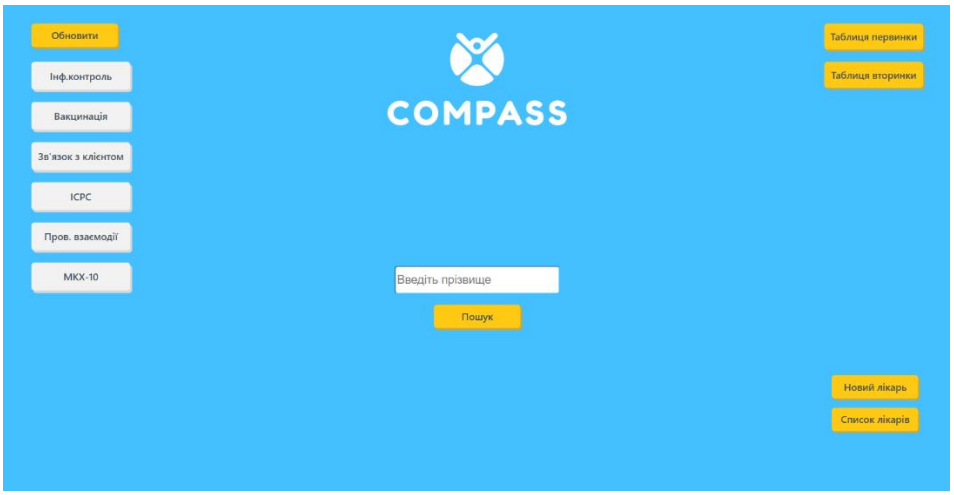

Рисунок 1.4 - Інтерфейс попередньої версії програми «Електронна система індикаторів якості»

1.3 Постановка задачі

Веб-додаток «Електронна система індикаторів якості» дозволить відкинути недоліки, які були приведені при аналізі аналогів. Основною задачею при розробці є створення програмного забезпечення прямо направленого на збереження, обчислення та роботу з індикаторами, а гнучке їх налаштування дозволить працювати з ними у інтересах бізнес-логіки, не використовуючи при цьому макроси та формули, які бувають занадто складні, або обмежені у функціоналі. Також, варто зазначити те, що обов'язковою умовою є забезпечити захист даних від помилок, завдяки збереженню згенерованих звітів та заповненої інформації на сервері без права доступу сторонніх користувачів.

Проаналізувавши попередню версію веб-додатку, будуть усунені знайдені недоліки, а саме оновлення архітектури та підходу до її

конструювання, розширення можливостей по налаштуванню підрахунку індикаторів та покращення адаптивності веб-додатку.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Аналіз предметної області та огляд існуючих рішень.
2. Специфікація функціональних і нефункціональних вимог до системи.
3. Моделювання поведінки системи.
4. Визначення сутностей і розробка бази даних.
5. Проектування і реалізація UI.
6. Програмна реалізація основних функцій системи.

Важливим є те, що дане програмне забезпечення розробляється для безоплатного користування на території України на благо розвитку підприємств та бізнесу, шляхом впровадження вітчизняних програмних рішень.

1.4 Обґрунтування вибору набору технологій проектування frontend частини веб-додатку

Frontend – це публічна частина веб-додатку або веб-сайту, з котрою користувач може взаємодіяти та контактувати напряму, тобто являє собою клієнтську частину. До неї входить відображення веб-сторінок, функціональних задач, які виконуються на стороні клієнта, інтерфейсу користувача і також опрацювання запитів, які були створені під час користування веб-ресурсом та відправлені до серверу [4].

1.4.1 HTML

Для конструювання веб-сторінок проекту використана мова гіпертекстової розмітки HTML (HyperText Markup Language).

Основною причиною для цього вибору послугувало те, що її використання є класичним та практичним методом. Кожен веб-сайт є створеним за допомогою, перш за все, даної стандартизованої мови розмітки.

1.4.2 Bootstrap

Задля спрощення процесу написання HTML-коду було використано такий фреймворк, як Bootstrap. Фреймворк – шаблон, існуючий для полегшення програмування та створений для визначених потреб, на основі якого можна дописати свій код [5]. Отож, Bootstrap це – вільний набір інструментів для створення веб-сайтів та веб-додатків [6], який не тільки дає змогу створити розмітку сторінки відразу адаптивною, а ще і оформлено виглядаючою. Даний фреймворк складається з [7]:

– інструментів для створення макету, а саме обгорткових контейнерів, систем сіток, адаптивних класів та багатофункціональних медіа-об'єктів;

– набору класів для стилізації базового контенту, такого як текст, зображення, таблиці тощо;

– готових компонентів користувальницького інтерфейсу, а саме кнопок, форм, горизонтальних та вертикальних навігаційних панелей, слайдерів, спливаючих підказок тощо;

– набору класів, для відображення елементів на сторінці (вирівнювання, вкриття, настройка відступів).

Саме готовий набір CSS та JavaScript файлів дає можливість використовувати Bootstrap у веб-ресурсі. Даний фреймворк дозволяє верстати сторінки в декілька разів скоріше, ніж за допомогою CSS. Відкривши каталог елементів на офіціальній сторінці Bootstrap в Інтернеті, та використовуючи вже заготовлені класи для HTML-тегів, можна додавати до сторінки елементи зі вже прописаними CSS-стилями та скриптами. Все це дало змогу прискорити створення дизайну для веб-додатку.

Завдяки створенню розмітки із використанням Bootstrap, веб-сайт стає адаптивним. Адаптивний веб-дизайн — це створення вебсайтів, які автоматично налаштовуються, щоб добре виглядати на всіх пристроях, від маленьких телефонів до великих дисплеїв [8].

1.4.3 CSS

Безпосередньо для впровадження стилів було використано CSS – каскадну таблицю стилів. Щоб створити унікальний веб-дизайн та структуру на сторінках веб-додатків, використовувалась дана формальна мова опису зовнішнього вигляду сторінки. CSS є такою ж невід'ємною складовою, як і HTML у створенні сучасного веб-сайту. Задля спрощення користування веб-додатком та неповторності досвіду користування їм, було доповнено стилі елементів із Bootstrap.

1.4.4 JavaScript

З ціллю створення динамічних візуальних ефектів та додавання інтерактиву використовувався JavaScript – скриптова мова програмування, яка здатна створювати скрипти для HTML-сторінок [9]. Скрипт – чітко прописана послідовність дій, яка описана за допомогою скриптової мови програмування [10]. Безпосередньо наразі створено велику кількість інструментів для JavaScript, які можуть відкрити нові функції у веб-додатку:

– динамічне створення HTML-коду та встановлення CSS-стилів;

– маніпуляції з відео потоком, робота з веб-камерою користувача та генерування 3D графіки;

– додавання функціоналу з інших сайтів, наприклад Twitter, Facebook тощо, надаючи користувачу можливість, знаходячись на одному веб-ресурсі працювати з іншими;

– додавання фреймворків та бібліотек для роботи з HTML, що може пришвидшити створення веб-додатків та сайтів.

Саме гнучкість визначає таку мову програмування, як JavaScript. Забезпечуючи інтерактивність на сайті, вона дає можливість створювати «події» – структури, які слідкують за тим, що відбувається у браузері і після цього надають змогу запускати код у відповідь на ті чи інші дії. Завдяки цьому веб-додаток із візуально-статичного стає візуально-динамічним та інтерактивним, що покращує досвід користування ресурсом. Також завдяки даній мові програмування відкриваються ще більші можливості у створення унікального веб-дизайну. Яскравим прикладом цього може слугувати фреймворк jQuery.

1.4.5 jQuery

jQuery – це JavaScript-фреймворк, який фокусується на взаємодії JavaScript, HTML та CSS. Має наступні можливості [11]:

– звертання до кожного елементу розмітки на моделі документу та маніпулювання їм;

– опрацювання подій;

– створення різноманітних візуальних ефектів;

– робота з AJAX – технологією, яка дозволяє тримати зв’язок із сервером та опрацьовувати його дані;

– підключення багатої кількості плагінів-доповнень, які дають змогу створювати елементи інтерфейсу користувача.

Даний фреймворк існує для полегшення роботи із JavaScript, «розвантажуючи» складну структуру коду даної мови програмування, скорочуючи його до декількох вже готових функцій, призначених спеціально для використання у створенні веб-додатків.

1.5 Обґрунтування вибору набору технологій проектування backend частини веб-додатку

Backend – програмно-апаратна частина веб-сервісу, яка відповідає за функціонування його внутрішньої (серверної) частини. Дана частина працює з логікою веб-додатку та є скритою від користувача. [12] Усі дії здійснюються за межами браузера. Як тільки з клієнтської частини надходить запит, він передається на опрацювання до серверу, де виконуються команди згідно із запитом для подальшого виводу інформації на екрані користувача, або відбувається прописана у коді реакція на нього. Безпосередньо логіка веб-сайту полягає у трьох кроках:

– відправлення інформації від користувача;
– її опрацювання на сервері;
– отримання інформації та форматування її в читаємий вигляд.

Варто відзначити і те, що засоби проектування frontend та backend частин значно відрізняються принципами роботи. В той час, коли задля створення клієнтської частини веб-ресурсу використовуються мови розмітки та таблиці стилів, для впровадження серверної частини треба працювати з мовами програмування, які додають гнучку логіку, щоб веб-додаток був динамічним та відповідав цільовим потребам користувачів.

1.5.1 PHP

Задля створення динаміки у опрацюванні даних веб-додатку, яка безпосередньо відрізняє його від веб-сайту, та роботі із сервером, було використано PHP – скриптову мову програмування, яка наразі підтримується переважною кількістю хостинг-провайдерів та є однією з лідерів у створенні динамічних веб-ресурсів [13]. Вона являє собою backend частину, отож працює із сервером та оперує інформацією. Використовуючи цю мову, більшість сторінок проекту створена, як PHP-сторінки з HTML-розміткою, але

зі впровадженою динамікою даних, які відображаються на них за допомогою саме PHP і можуть оброблятися. Ця мова програмування дозволяє створювати SQL-запити до бази даних, щоб отримати з неї інформацію, з якою буде працювати для подальшого використання на веб-сторінці. Дає можливість створювати сесії на сайті – після успішної авторизації користувач знаходиться у активній сесії у веб-додатку, що дозволяє йому використовувати його можливості. Змога інтегрувати HTML-код у PHP-файл додає динаміки у створенні розмітки веб-сторінок.

1.5.2 AJAX

Для роботи з backend також розглядалася технологія AJAX, яка дозволяє взаємодіяти із сервером не перезавантажуючи сторінку. Дана функція обумовлена тим, що при її використання оновлюється не весь веб-ресурс, а тільки його конкретна частина [14]. Використовуючи асинхронну передачу даних, AJAX дає змогу здійснювати інші, потрібні користувачу, дії на сайті, в той час коли дані на веб-сторінці можуть динамічно змінюватися. Такий підхід вважається зручним та економить час користувача, але дана технологія має свої недоліки та незручності, зокрема головними недоліками є: відсутність підтримки усіма браузерами (застарілі версії браузерів не підтримують) та можливість відключення JavaScript на веб-сайті користувачем або безпосередньо у разі помилки на сервері.

Технологія AJAX пропонує опрацювання запитів користувача, базуючись на JavaScript та потребуючи написання логіки безпосередньо на цій мові програмування для більш ефективного використання [14]. Саме це являє собою першу незручність, яка полягає у тому, що використовуючи PHP, як головний інструмент для створення backend частини веб-додатку, неможливо максимально ефективно працювати з AJAX. Прописана на PHP логіка, яка більш простіша у розумінні, створенні та налагоджені, стабільно функціонує без даної технології. Створення «середовища» для злагодженого

впровадження AJAX, яке полягає у активному використанні JavaScript у серверній частині веб-додатку, виявилось складнішим процесом. Другою незручністю послугувало те, що маючи деякі специфічні системні налаштування серверу та браузеру (наприклад використання заздалегідь налаштованого системою типу запитів на сервері, або «підчищення» url-адреси браузером) не дозволяють коректно використовувати технологію AJAX. Під час створення backend частини, використовуючи дану технологію з ціллю опрацювання запитів та передачі даних на сервер, зіткнувся з неможливістю роботи з даними процедурами, що було обумовлено специфікою функціонування AJAX. Це ставить під питання надійність роботи з веб-додатком іншими користувачами, в яких можуть бути встановлені ті чи інші налаштування, які перешкоджають користуванню веб-сервісом зі впровадженою даною технологією. Таким чином, технологія AJAX не була використана у створенні веб-додатку.

1.5.3 Бібліотека «PhpSpreadsheet»

Також була використана PHP-бібліотека «PhpSpreadsheet» для генерування Excel-таблиць для завантаження. «PHPSpreadsheet» – це бібліотека з відкритим вихідним кодом, яка включає набір класів, що дозволяють взаємодіяти і використовувати різні формати файлів електронних таблиць Microsoft Excel і LibreOffice Calc.

«PHPSpreadsheet» підтримує електронні таблиці, які містять один або кілька робочих аркушів, що містять комірки для зберігання даних різних типів, таких як числа, формули, зображення та інше [15].

1.5.4 Система об'єктно-реляційного відображення Eloquent

Задля перебудови архітектури та покращення роботи із запитами до БД було обрано Eloquent. Система об'єктно-реляційного відображення (ORM)

Eloquent — це реалізація шаблону Active Record (AR) у Laravel для роботи з базами даних [16]. Шаблон AR — це шаблон проектування, що використовується при реалізації доступу до реляційних баз даних. Інтерфейс такого об'єкта включає функції CRUD, а також поля, що відповідають полям відповідної таблиці в базі даних. Active Record реалізує популярний підхід об'єктно-орієнтованого проєкціювання. Кожен клас AR відображає таблицю (чи представлення) бази даних, екземпляр AR — запис цієї таблиці, а загальні операції CRUD реалізовані як методи AR [17]. Варто вдізначити те, що Eloquent може бути імплементований до проекту як окремий модуль, без фреймворка Laravel, чим я і вирішив скористатися. Використання шаблону AR дозволить зробити архітектуру додатку більш гнучкою із відкритими можливостями для розробки нових модулів та доопрацювань без перенавантаження кодової бази. Завдяки Eloquent буде покращена швидкодія веб-додатку через спрощення запитів до БД, а об’єктно-орієнтований підхід підвищить ефективність взаємодії із даними.

1.6 Обґрунтування вибору системи керування базою даних

Невід’ємною частиною кожного веб-додатку є база даних (БД) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних та взаємозв’язки між їх елементами [18].

Всі дані у БД контролюються і опрацьовуються завдяки системі керування базою даних (СКБД) – сукупності програмних та лінгвістичних інструментів, які забезпечують управління над створенням та користуванням баз даних [19]. Завдяки цьому з’являється можливість за допомогою коду працювати з БД.

Для збереження інформації та її опрацювання була обрана СКБД MySQL – програмне забезпечення для створення та керування БД на основі реляційної моделі [20]. Отримуючи SQL-запити від серверної частини веб-додатку,

MySQL опрацьовує їх та передає потрібні дані з БД, або доповнює та додає нові записи безпосередньо до неї.

1.7 Обґрунтування вибору локального серверу

Локальний сервер – це емулятор хостингу, який потрібен для того, щоб створювати веб-додатки та веб-сайти з подальшим перенесенням їх на хостинг [21]. Використання його зумовлено наявністю вже встановлених у ньому компонентів, таких як: база даних MySQL, мови програмування PHP та веб-сервери Apache або Nginx. Налаштована зв’язка цих компонентів і є локальним сервером. Даний сервер дозволяє створювати резервні копії про’ктів та мати можливість запускати та редагувати проекти без підключення платного хостингу.

Задля використання локального серверу було обрано віртуальну машину Homestead. Homestead – це офіційна предвстановлена віртуальна машина, яка являє собою віртуальний комп'ютер (має свою пам'ять, процесор, жорсткий диск тощо) в якому встановлена операційна система Ubuntu Server. Доступ до цього Ubuntu Server здійснюється через консоль SSH [22]. За допомогою такого інструменту, як Vagrant, можна з легкістю розгортати віртуальну машину для роботи з проектом. Vagrant – це інструмент для створення та управління віртуальними середовищами розробки. Він дозволяє розробникам легко створювати та налаштовувати віртуальні машини на своїх комп'ютерах [23].

Homestead являє собою набір налаштувань та скриптів, в яких прописано що потрібно встановити, а Vagrant в свою чергу виконує все у створеному віртуальному середовищі, яке представляє собою сервер з усім необхідним для запуску проекту. Вибір цих технологій було зроблено через легкість у налаштуванні та запуску. При коректному написанні конфігураційного файлу для Homestead, Vagrant за декілька секунд здатен запустити середовище готове до роботи.

1.8 Обґрунтування вибору середовища розробки

Середовищем розробки було обрано PhpStorm (рис. 1.5) – це спеціалізований інструмент веб-розробки, орієнтований на веб-додатки та інші види програм, які можна створити за допомогою мови PHP і за допомогою HTML, JavaScript і CSS [24].

Його особливостями є:

– спеціалізація на PHP розробці, тому дане середовище має всі необхідні інструменти та функції для комфортної і продуктивної роботи з PHP проектами;

– наявність інтегрованого середовища розробки, яке включає в себе текстовий редактор, консоль, відлагоджувач та інші необхідні інструменти;

– підтримка різних PHP-фреймворків, таких як Laravel, Symfony, Yii, Zend Framework тощо. PhpStorm надає автоматичне завершення коду, аналіз коду та інші корисні функції для роботи з цими фреймворками;

– швидкодія середовища, що дозволяє зосередитися на розробці та збільшити продуктивність;

– підтримка інших мов програмування та розмітки, таких як HTML, CSS, JavaScript, SQL та інші. Це робить PhpStorm універсальним інструментом для повноцінної веб-розробки;

– наявність великої кількісті налаштувань, що дозволяє адаптувати середовище під власні потреби та стиль роботи.

Дізнавшись про PhpStorm більше, я обрав його, як середовище розробки для свого проекту завдяки розширеним можливостям у роботі, таким як запуск локального сервера з терміналу, робота з БД у самому середовищі та можливість SSH-підключення не виходячи із середовища.

Незважаючи на те, що інтерфейс PhpStorm є багатокомпонентним, у Інтернеті є багато туторіалів по налаштуванню середовища під свої потреби.

Спеціалізація PhpStorm на обраній мною мові програмування, робить його найбільш підходящим інструментом для розробки та реалізації поставлених задач.

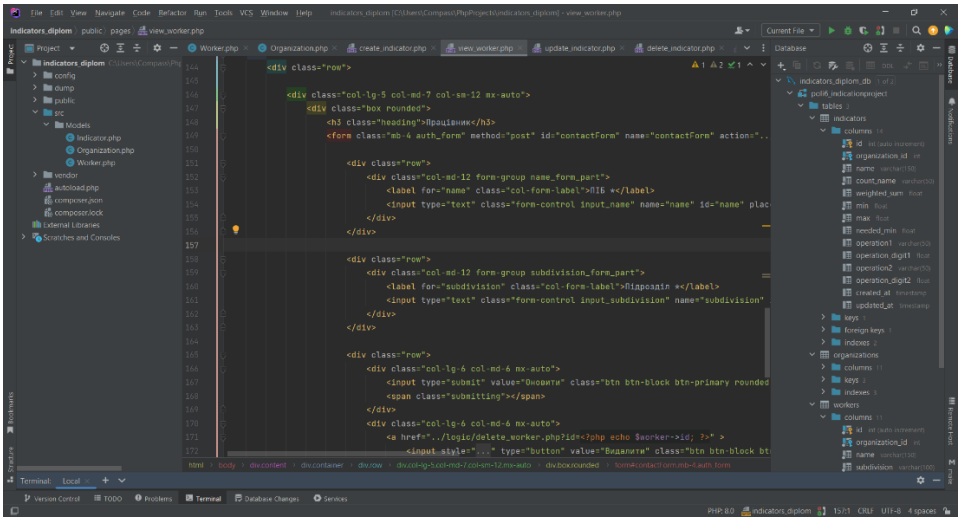

Рисунок 1.5 – Інтерфейс середовища розробки PhpStorm

2 ПРОЕКТУВАННЯ

2.1 Проектування за допомогою методології функціонального моделювання SADT

При проектуванні веб-додатку «Електронна система індикаторів якості» була обрана методологія функціонального моделювання SADT (Structured Analysis and Design Technique), стандарт IDEF0. Методологія SADT є методологією, яка використовується для аналізу та проектування систем. Вона дозволяє відобразити важливі характеристики системи, такі як: управління системою, зворотній зв'язок і ресурси. За допомогою графічної мови IDEF0, стає можливим представити веб-додаток у вигляді набору взаємопов'язаних функціональних блоків, що надає змогу вивчити загальне функціонування системи. Контекстна діаграма веб-додатку «Електронна система індикаторів якості» наведена на рис. 2.1.

Рисунок 2.1 – Контекстна діаграма веб-додатку

Методологія SADT надає розробнику методи, правила та процедури для створення функціональної моделі об'єкта в обраній предметній області. Графічна мова IDEF0, яка є частиною SADT, дозволяє описувати бізнес-

процеси цієї предметної області у вигляді ієрархічної системи взаємопов'язаних функцій.

Найбільш загальний опис інформаційної системи та її взаємодії з зовнішнім середовищем можна представити у формі контекстної діаграми. На даній діаграмі описана головна задача, а саме «Розробка веб-додатку «Електронна система індикаторів якості» для генерації Excel-звіту оцінки якості роботи». На вхід подається набір налаштувань та умов для підрахунку оцінки якості роботи. Головна задача керується загальними принципами розробки веб-додатків, документаціями MySQL, Laravel та PhpSpreadsheet. Механізмом її роботи є програмне забезпечення. Виходом є готовий згенерований Excel-звіт оцінки якості роботи працівників.

Після опису інформаційної системи, проводиться функціональна декомпозиція, а діаграми, які описують кожен фрагмент і взаємодію фрагментів, називаються діаграмами декомпозиції, тобто головна задача ділиться на декілька менших.

Синтаксис опису системи в цілому і кожного її фрагмента однаковий у всієї моделі. Після декомпозиції контекстної діаграми, можна побачити два блоки – роботи (рис. 2.2).

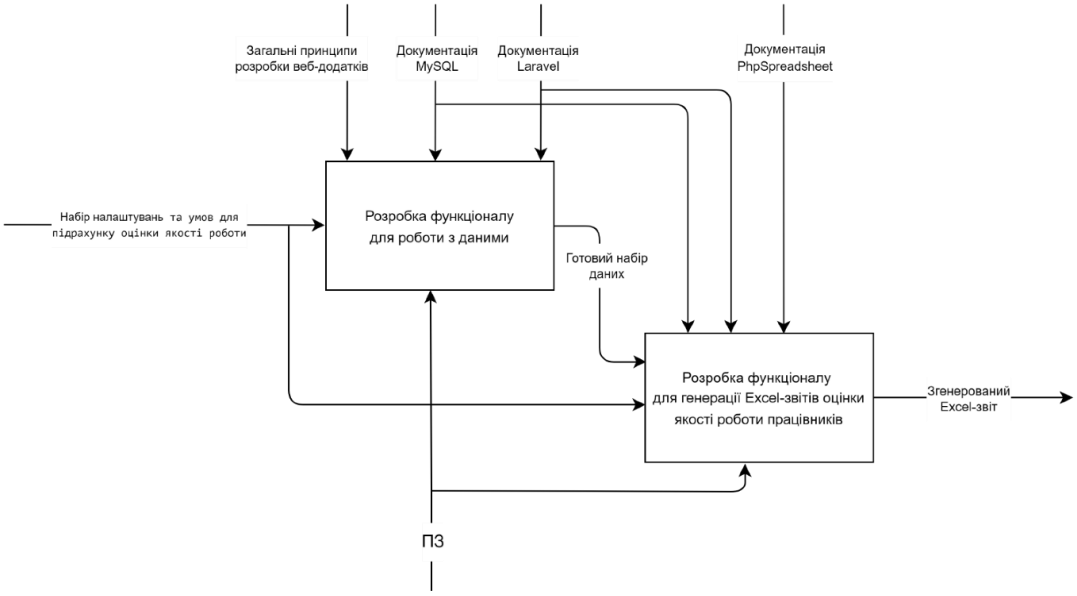

Рисунок 2.2 – Діаграма декомпозиції веб-додатку

Робота «Розробка функціоналу для роботи з даними» полягає у створенні методів та функцій, завдяки яким можна буде працювати із даними, які вводить користувач, та коректно опрацьовувати їх на серверній частині веб-додатку. Вхідними даними для неї є набір налаштувань та умов для підрахунку оцінки якості роботи. Керується вона за допомогою загальних принципів розробки веб-додатків і документаціями MySQL та Laravel. Механізмом є програмне забезпечення, яке потрібне для здійснення розробки. Результатом роботи є готовий набір даних для подальшої генерації звіту.

Наступна робота «Розробка функціоналу для генерації Excel-звітів оцінки якості роботи працівників» при виконанні дасть змогу веб-додатку генерувати Excel-звіти для аналізу якості роботи працівників. Вхідними даними для неї є набір налаштувань та умов для підрахунку оцінки якості роботи та готовий набір даних. Керується вона за допомогою документацій MySQL, Laravel і PhpSpreadsheet. Механізмом є програмне забезпечення. Результатом роботи є готовий згенерований Excel-звіт.

При здійсненні наступного етапу декомпозиції системи (рис. 2.3), а саме блоку «Розробка функціоналу для роботи з даними», було отримано три блоки декомпозиції.

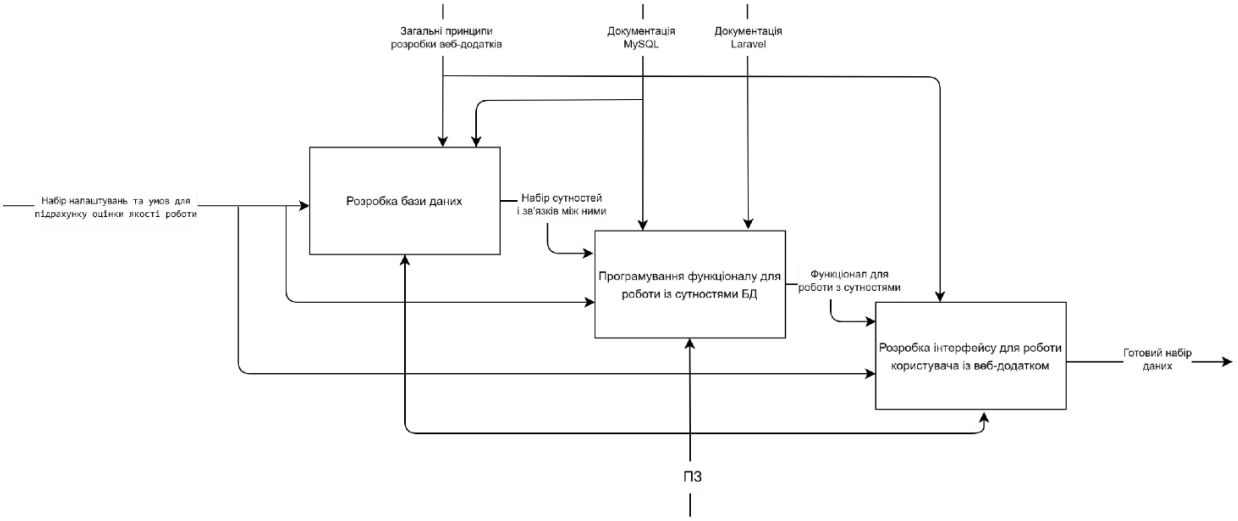

Рисунок 2.3 – Діаграма декомпозиції блоку А1

Робота «Розробка бази даних» включає у себе опрацювання сутностей веб-додатку, створення таблиць, зв’язків між ними і поля, в яких буде зберігатись інформація. Вхідними даними для неї є набір налаштувань та умов для підрахунку оцінки якості роботи та готовий набір даних. Керується вона за допомогою загальних принципів розробки веб-додатків і документації MySQL. Механізмом є програмне забезпечення. Результатом роботи є готовий набір сутностей і зв’язків між ними.

Наступна робота «Програмування функціоналу для роботи із сутностями

БД» дасть змогу мати розроблений функціонал для серверної частини, завдяки якому дані, введені у подальшому користувачем, будуть опрацьовуватись коректно. Вхідними даними для неї є набір налаштувань та умов для підрахунку оцінки якості роботи та готовий набір сутностей і зв’язків між ними, що є результатом попередньої роботи. Керується вона за допомогою документацій MySQL та Laravel. Механізмом є програмне забезпечення. Результатом роботи є готовий функціонал для роботи з сутностями.

Наступна робота «Розробка інтерфейсу для роботи користувача із веб-додатком» полягає у створені клієнтської частини веб-додатку, яка буде відображатись у браузері, завдяки якій користувач матиме змогу керувати своїм обліковим записом у веб-додатку. Вхідними даними для неї є набір налаштувань та умов для підрахунку оцінки якості роботи та готовий функціонал для роботи з сутностями. Керується вона за допомогою загальних принципів розробки веб-додатків. Механізмом є програмне забезпечення. Результатом роботи є готовий набір даних для подальшої генерації звіту.

При здійсненні наступного етапу декомпозиції системи (рис. 2.4), а саме блоку «Розробка функціоналу для генерації Excel-звітів оцінки якості роботи працівників», було отримано чотири блоки декомпозиції.

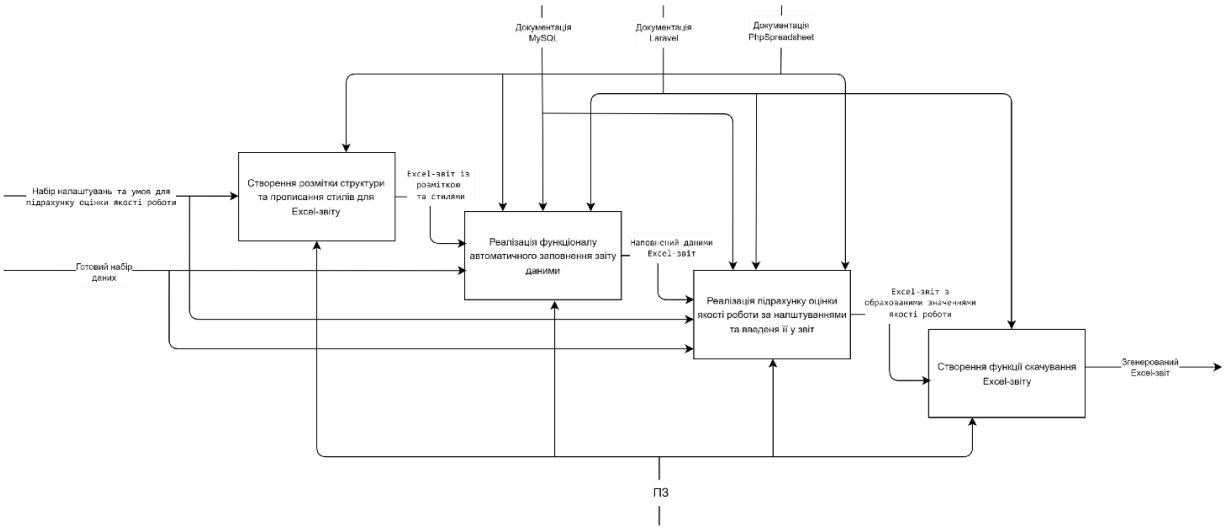

Рисунок 2.4 – Діаграма декомпозиції блоку А2

Робота «Створення розмітки структури та прописання стилів для Excel-звіту» має наступні вхідні дані: набір налаштувань та умов для підрахунку оцінки якості роботи. Керується вона за допомогою документації PhpSpreadsheet. Механізмом є програмне забезпечення. Результатом роботи є Excel-звіт із побудованою розміткою та прописаними стилями.

Наступна робота «Реалізація функціоналу автоматичного заповнення звіту даними» має наступні вхідні дані: Excel-звіт із побудованою розміткою та прописаними стилями та готовий набір даних. Керується вона за допомогою документацій MySQL, Laravel і PhpSpreadsheet. Механізмом є програмне забезпечення. Результатом роботи є наповнений даними Excel-звіт.

Робота «Реалізація підрахунку оцінки якості роботи за налаштуваннями та введення її у звіт» має наступні вхідні дані: набір налаштувань та умов для підрахунку оцінки якості роботи, готовий набір даних та наповнений даними Excel-звіт. Керується вона за допомогою документацій MySQL, Laravel і PhpSpreadsheet. Механізмом є програмне забезпечення. Результатом роботи є Excel-звіт з обрахованими значеннями якості роботи.

Наступна робота «Створення функції скачування Excel-звіту» полягає у розробці функціоналу для скачування звіту у браузері користувача. Вхідними

даними для неї є Excel-звіт з обрахованими значеннями якості роботи. Керується вона за допомогою документації Laravel. Механізмом є програмне забезпечення. Результатом роботи є готовий згенерований Excel-звіт.

2.2 Моделювання варіантів використання веб-додатку «Електронна система індикаторів якості»

Наступним етапом проектування веб-додатку «Електрона система індикаторів якості» була розробка UML діаграми Use Case.

Основна ідея моделювання випадків використання полягає в тому, щоб розробити систему з погляду кінцевого користувача. Це ефективний спосіб передати поведінку системи з точки зору користувача, вказуючи всю видиму ззовні поведінку системи.

При розробці діаграми використання веб-додатку був виявлений один актор, який представляє користувача системи.

Актор «Користувач» може виконувати дії, що зазначені нижче:

− виконати авторизацію із можливістю відновлення паролю;

− змінити налаштування облікового запису;

− створити індикатор якості із можливостями його зміни та видалення;

− створити працівника із можливостями його зміни та видалення;

− виконати пошук персоналу по ПІБ;

− переглянути список персоналу;

− отримати перелік назв підрозділів;

− заповнити дані об оцінках якості роботи працівників у індикатори;

− отримати згенерований звіт із можливостями використати прописані налаштування і скачати його.

Розроблена діаграма Use Case, представлена на рис. 2.5.

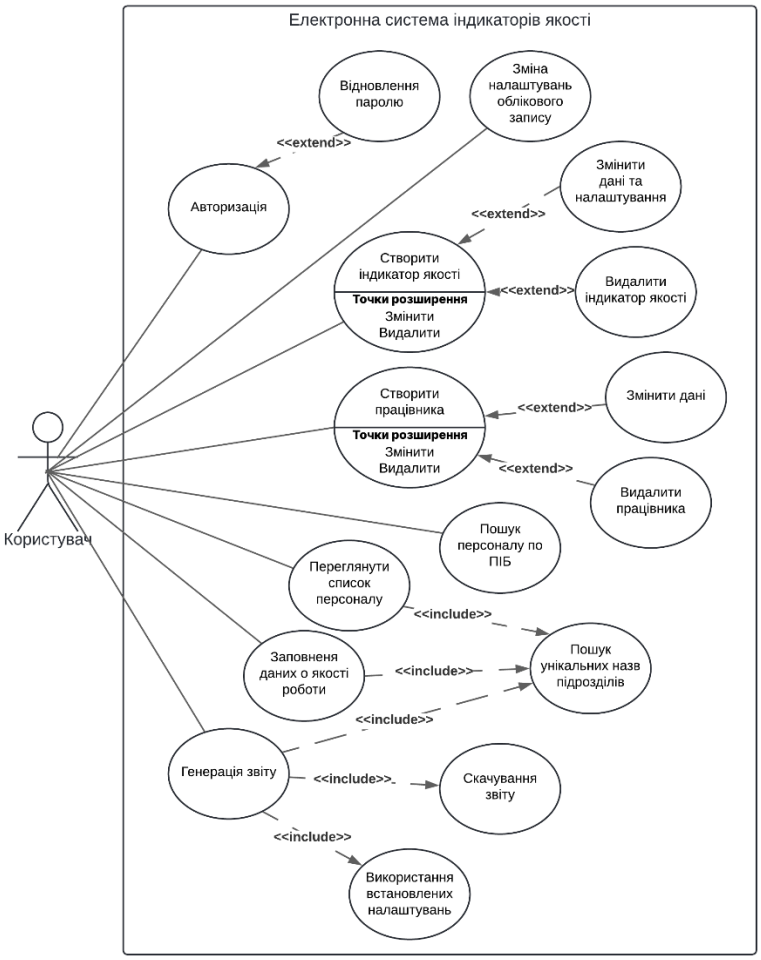

Рисунок 2.5 – Діаграма варіантів використання веб-додатку «Електронна система індикаторів якості»

2.3 Проектування бази даних

Наступним кроком у проектуванні веб-додатку є проектування бази даних, оскільки «Електронна система індикаторів якості» повинна мати сховище даних, завдяки якому буде можливість згенерувати Excel-звіт оцінки якості роботи працівників.

Першим етапом є концептуальне проектування. Тобто, потрібно створити концептуальну модель даних для системи, яка наразі проектується. Дана модель даних створюється на основі функціональних вимог користувачів та не залежить від шляхів її реалізації. Концептуальне проектування бази

даних включає у себе визначення сутностей, атрибутів, їх типів даних та визначення зв’язків між ними. При здійснені проектування були визначені основні сутності системи.

Сутність «Організації» містить наступні атрибути (табл. 1): «id» – ідентифікатор організації, тобто, облікового запису; «name» – назва організації або облікового запису; «email» – адреса електронної пошти; «password» – пароль для входу; «reset_token_hash» – хеш токену для зміни паролю; «reset_token_expires_at» – відмітка часу для визначення актуальності токену; «final_count_formula» – формула фінального підрахунку оцінки якості роботи; «needed_points» – мінімальна потрібна кількість балів; «excellent_points» – відмінна кількість балів; «created_at» і «updated_at» – для відстеження змін.

Таблиця 1 – Сутність «Організації»

№	Атрибут	Тип
1	id	int(10)
2	name	varchar(100)
3	email	varchar(100)
4	password	varchar(255)
5	reset_token_hash	varchar(64)
6	reset_token_expires_at	datetime
7	final_count_formula	varchar(50)
8	needed_points	float
9	excellent_points	float
10	created_at	timestamp
11	updated_at	timestamp

Сутність «Індикатори» містить наступні атрибути (табл. 2): «id» – ідентифікатор індикатору; «organization_id» – ідентифікатор організації, до якої належить індикатор; «name» – назва індикатору; «count_name» – назва із порядковим номером; «weighted_sum» – значення «ваги» для обрахування

фінальної оцінки якості роботи методом зваженої суми; «min» – мінімальне допустиме значення; «max» – максимальне допустиме значення; «needed_min» – значення потрібного мінімуму балів для індикатору; «operation1» – перша математична операція у формулі підрахунку; «operation_digit1» – перше число для формули підрахунку; «operation2» – друга математична операція; «operation_digit2» – друге число; «created_at» і «updated_at» – для відстеження змін.

Таблиця 2 – Сутність «Індикатори»

№	Атрибут	Тип
1	id	int(10)
2	organization_id	int(10)
3	name	varchar(150)
4	count_name	varchar(50)
5	weighted_sum	float
6	min	float
7	max	float
8	needed_min	float
9	operation1	varchar(50)
10	operation_digit1	float
11	operation2	varchar(50)
12	operation_digit2	float
13	created_at	timestamp
14	updated_at	timestamp

Сутність «Працівники» містить наступні атрибути (табл. 3): «id» – ідентифікатор працівника; «organization_id» – ідентифікатор організації, до якої належить запис о працівникові; «name» – ПІБ; «subdivision» – назва підрозділу; «indicator_1», «indicator_2», «indicator_3», «indicator_4», «indicator_5» – для збереження значень оцінок по індикаторам, які мають

порядкові номери від «1» до «5»; «created_at» і «updated_at» – для відстеження змін.

Таблиця 3 – Сутність «Працівники»

№	Атрибут	Тип
1	id	int(10)
2	organization_id	int(10)
3	name	varchar(100)
4	subdivision	varchar(100)
5	indicator_1	float
6	indicator_2	float
7	indicator_3	float
8	indicator_4	float
9	indicator_5	float
10	created_at	timestamp
11	updated_at	timestamp

Наступним кроком у проектуванні бази даних є визначення зв'язків між сутностями. Було побудовано діаграму бази даних у вигляді моделі «сутність-зв'язок». Діаграма моделі «сутність-зв'язок» представлена на рис. 2.6.

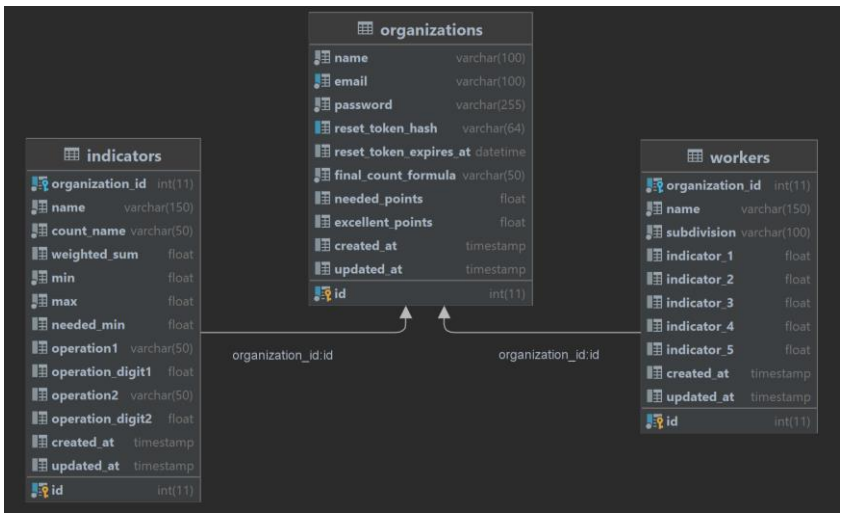

Рисунок 2.6 – Діаграма моделі «сутність-зв’язок»

На даній діаграмі можна побачити зв’язок «один-до-багатьох» між сутностями «Організації» і «Індикатори» та аналогічний зв’язок між сутностями «Організації» і «Працівники». Інакше кажучи, у однієї організації може бути декілька індикаторів та працівників.

Також, варто відзначити, що базу даних на етапі проектування було нормалізовано до третьої нормальної форми. Шляхом видалення частково залежних атрибутів було досягнуто другої нормальної форми. Після цього видаливши усі транзитивно-залежні атрибути було досягнуто третьої нормальної форми.

У результаті проведеного проектування веб-додатку «Електронна система індикаторів якості» визначена архітектура системи, проведено моделювання процесів та розроблена база даних. Основним результатом проектування стала реалізація вимог, функціональних можливостей та бізнес-логіки для початку розробки кодової бази веб-додатку.

3 РОЗРОБКА ПРОЕКТУ

3.1 Реалізація моделей сутностей

Маючи спроектовану та побудовану базу даних, першим кроком до роботи із нею за принципами об’єктно-орієнтованого програмування є створення моделей сутностей, описаних у базі даних. Наприклад, створення моделі сутності «Організації» виглядає наступним чином:

```
use Illuminate\Database\Eloquent\Model;
class Organization extends Model
{
    public function indicators(){
        return $this->hasMany(Indicator::class);
    }
    public function workers(){
        return $this->hasMany(worker::class);
    }
}
```

Дана структура реалізована за допомогою спадкування від класу «Model» із бібліотеки «Eloquent», який надає змогу користуватись усіма, потрібними для роботи із даними у БД, методами, такими як зберігання, видалення, перегляд тощо.

У створеному класі «Organization» описані методи «indicators» та «workers», за допомогою яких можна буде отримати колекцію об’єктів відповідних класів, які є записами у БД, залежними від організації зв’язком «один-до-багатьох». Для цього використовується метод «hasMany».

Створення моделі сутності «Індикатори» має наступний вигляд:

```
class Indicator extends Model
{
    public function organization(){
        return $this->belongsTo(Organization::class); } }
```

У даному фрагменті коду, у створеному класі «Indicator», який також

спадкується від класу «Model», прописується метод «organization». У даному методі, використовуючи наступний метод «belongsTo», вказується принадлежність сутності «Індикатори» до зв'язку «один-до-багатьох» із сутністю «Організації», задля подальшої зручної роботи із базою даних на серверній частині.

Створення моделі сутності «Працівники» є аналогічним із моделлю «Індикатори».

3.2 Блок реєстрації та авторизації

Задля користування функціоналом веб-додатку, користувач повинен зареєструватись, але, у разі, якщо він вже проходив реєстрацію - авторизуватись (рис. 3.1). Було створено блок реєстрації та авторизації, який складається з заповнення форми на веб-сторінці та опрацювання уведених даних до неї на сервері, який в свою чергу робить запит до бази даних, у якій або записує нового користувача, або у разі авторизації перевіряє уведені дані на предмет наявності такого користувача у БД.

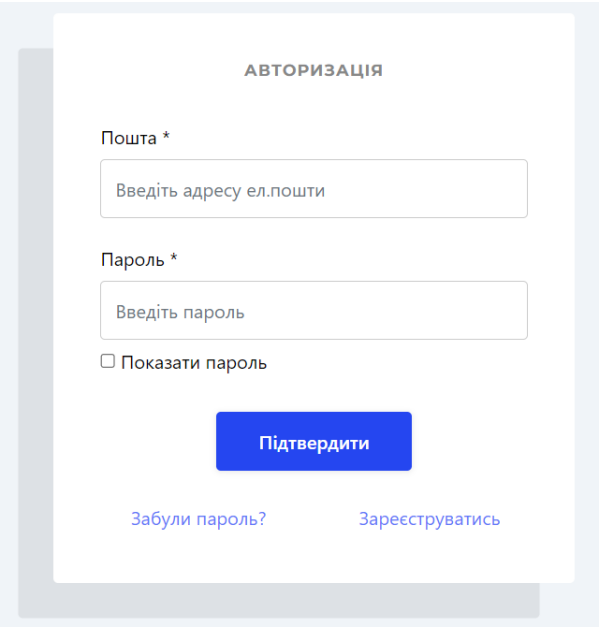

Рисунок 3.1 - Інтерфейс форми авторизації

Наявність блоку реєстрації та авторизації обумовлена тим, що даний веб-додаток розробляється для використання багатьма організаціями, що створює потребу у реєстрації облікових записів із персональними налаштуваннями для кожного з них.

Фрагмент коду форми виглядає наступним чином:

```
<form class="mb-4" method="post" id="contactForm" name="contactForm"
action="logic/authorization.php">
    <div class="row">
        <div class="col-md-12 form-group ">
            <label for="email" class="col-form-label">Пошта *</label>
<input type="text" class="form-control" name="email" id="email"
placeholder="Введіть адресу ел.пошти" minlength="5" maxlength="100"
required>
        </div>
    </div>
```

Безпосередньо, сама розмітка форми є розгалуженою через використання фреймворку Bootstrap та містить велику кількість контейнерів-обгорток, які наділяють веб-елементи адаптивністю, тому я визнав доцільним відобразити лише фрагмент коду, в якому висвітлено атрибути форми та поле для введення електронної пошти користувача.

У тегу «form», завдяки атрибуту «action» вказано PHP файл, який опрацьовує, створений методом POST, запит, як вказано у атрибуті «method».

Тег «input» відповідає за відображення поля для вводу текстових даних, що вказано в атрибуті «type», а також вказані мінімальна та максимальна кількість допустимих символів при заповненні поля, за що відповідають атрибути «minlength» і «maxlength». В свою чергу, поле для вводу пароля було налаштовано відповідним чином, із вказанням в атрибуті «type» значення «password». Також, завдяки JavaScript було реалізовано функцію відображення пароля для його перевірки, щоб користувачеві було зручно передивлятись уведені дані, відмітивши задля цього чек-бокс «Показати пароль». Код реалізації даної функції виглядає наступним чином:

```
$('.toggle_pass').on('click', function(){
    var x = document.getElementById("password");
    if (x.type === "password") {
         x.type = "text";
    } else {
        x.type = "password"; } });
```

В даному фрагменті коду зображено, що при фіксації події відмічання чек-боксу «Показати пароль», обробляється елемент зі значенням «id», яке дорівнює «password», використовуючи функцію «document.getElementById("password")». Успішно знайшовши потрібний елемент, який є полем з введеним паролем, перевіряється тип поля. У разі, якщо при відмічанні чек-боксу, атрибут «type» цього поля дорівнює «password», він змінюється на «text», що дає можливість переглянути введений пароль. І навпаки, при знятті прапорця в чек-боксі, тип поля знов стає «password» та данні скриваються.

При натисканні на кнопку підтвердження, при коректному заповнені усіх полів форми, серверною частиною отримуються значення, уведені у поля В подальшому вони обробляються для відправки їх у формі SQL-запиту до БД. Процес отримання інформації із бази даних виглядає наступним чином:

```
use Indicators\Models\Organization;
$organizations = Organization::where('email', $_POST['email'])->get();
```

Змінна «organizations» приймає значення колекції об’єктів, яка створюється при виконанні запиту до бази даних. В запиті, завдяки функції «where» відбувається пошук записів у таблиці «organizations», де поле «email» дорівнює значенню, яке передається з форми з клієнтської частини веб-додатку.

У разі успішного знаходження запису в таблиці, відбувається звіряння введеного паролю із вказаним при реєстрації, використовуючи функцію «password_verify». При підтвердженні коректності, створюється cookie-файл, який є невеликим фрагментом визначених даних, який зберігає комп’ютер

користувача для «запам’ятовування» відкритої сесії на веб-ресурсі на визначений час, завдяки функції «setcookie»:

```
if(password_verify($_POST['pass'], $organization->password)){
    setcookie('organization', $organization->id, time()+3600*24*30,"/");
    header("Location: ../pages/administration.php"); }
```

Результатом успішної авторизації є відкриття користувачеві сторінки адміністрування веб-додатком.

У тому разі, якщо користувач ще не має облікового запису, він може відкрити форму реєстрації натиснувши на «Зареєструватись» на формі авторизації. Після цього відкривається відповідна форма (рис. 3.2), в якій користувачу потрібно заповнити усі поля.

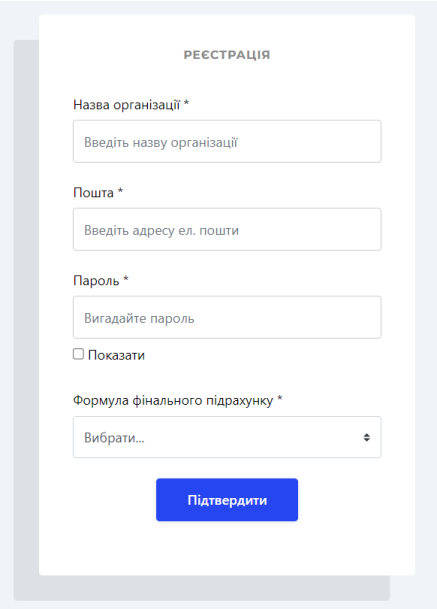

Рисунок 3.2 – Інтерфейс форми реєстрації

Варто відзначити поле «Формула фінального підрахунку», в якому потрібно обрати один з двох методів опрацювання значень індикаторів якості роботи, а саме «Метод зважених сум» або «Середнє арифметичне». В подальшому, після успішної реєстрації, користувач зможе змінити формулу підрахунку у налаштуваннях свого облікового запису.

Форма реєстрації має безпосередньо схожу структуру, але інший принцип дії, який полягає у створенні запису у БД, використовуючи дані, які були введені у формі:

```
$organization = new Organization();
$organization->name = $_POST['name'];
$organization->email = $_POST['email'];
$organization->password = password_hash($_POST['pass'],
PASSWORD_BCRYPT);
$organization->final_count_formula = $_POST['formula'];
```

Перш за все, створюється екземпляр класу «Organization», назва якого співвідноситься з таблицею «organizations» у БД, запис у якій буде створюватись, використовуючи методи Eloquent. Після цього, атрибутам класу, які відповідають назвам стовпців у таблиці, використовуючи глобальну змінну $_POST, присвоюються значення введені у формі. Пароль шифрується методом «password_hash» і тільки після цього, у формі хешу, надходить до БД задля безпеки зберігання даних користувачів.

Опрацювавши введені дані, блок «try-catch» спробує зберегти запис у базу даних, використовуючи метод «save» на екземплярі класу, та створити cookie-файл для подальшої роботи із веб-додатком:

```
try {
    $organization->save();
    setcookie('organization', $organization->id, time()+3600*24*30,"/");
    header("Location: /");
} catch (Exception $e) {
echo "Виникла помилка при опрацюванні даних або організація з такою
адресою пошти вже існує"; }
```

Якщо не вдастся зберегти запис у БД, виведеться помилка, та користувачу потрібно буде ввести інший набір даних у форму.

Після успішної реєстрації, користувача буде перенаправлено на сторінку авторизації, з якої відбудеться автоматичний перехід на сторінку

адміністрування завдяки активному cookie-файлу, який було створено при реєстрації.

Також була реалізована функція відновлення паролю, на випадок якщо користувач забув свої дані для авторизації. Задля переходу на сторінку відновлення (рис. 3.3) треба натиснути на «Забули пароль?» у формі авторизації.

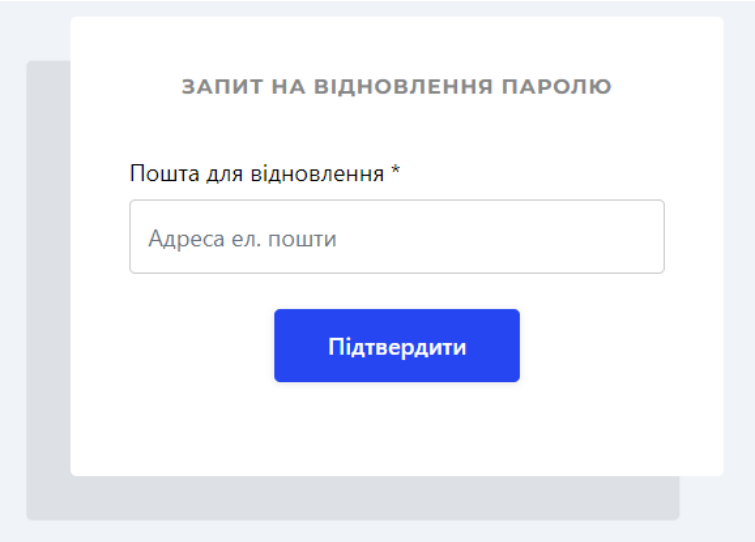

Рисунок 3.3 – Інтерфейс форми створення запиту на відновлення паролю

Форма включає в себе одне поле, в яке потрібне ввести адресу електронної пошти, на яку в подальшому прийде лист із посиланням на відновлення паролю для входу.

Процес відновлення паролю реалізовано шляхом відправки листа на вказану пошту та створення токену з обмеженим часом існування з ціллю убезпечити процедуру та унеможливити втручання шахраїв у неї.

Створення токену, завдяки якому відкривається можливість відновити пароль, виглядає наступним чином:

```
$token = bin2hex(random_bytes(16));
$token_hash = hash("sha256", $token);
$expiry = date("Y-m-d H:i:s", time() + 60 * 30);
```

У першому рядку коду створюється унікальний набір шістнадцяткових даних за допомогою функцій «bin2hex» і «random_bytes». Після цього, створена строка хешується з використанням алгоритму «sha256», як це вказано на другому рядку. Далі створюється змінна, яка зберігатиме відмітку часу, по закінченню якої токен буде неактуальним. Всі ці дані будуть передані то БД для подальшого опрацювання на етапі встановлення нового паролю для входу в обліковий запис, що відображено в наступному коді:

```
$email = $_POST["email"];
$organization = Organization::where('email', $email)->get();
$single_organization->reset_token_hash = $token_hash;
$single_organization->reset_token_expires_at = $expiry;
```

Після знаходження запису зі вказаною у формі адресою електронної пошти, до стовпців «reset_token_hash» і «reset_token_expires_at», де перший стовпець відповідає за зберігання хешу токену, а другий за час до якого токен є дійсним, вносяться оброблені раніше значення. Далі відбувається спроба зберігання даних у базі даних та відправка листа на електронну пошту:

```
try {
    $single_organization->save();
    $mail = require_once __DIR__ . '/../../config/mailer.php';
    $mail->setFrom("test_mail@indication-system.compass-hub.com.ua");
    $mail->addAddress($email);
    $mail->Subject = "Password reset";
    $mail->Body = <<<END
    Для відновлення паролю будь ласка перейдіть за посиланням:
    http://indicators_diplom/pages/reset_password.php?token=$token
    Якщо ви не подавали запиту на зміну паролю, проігноруйте цей лист.
    END;
    $mail->send(); }
```

На даному фрагменті коду можна побачити спробу збереження даних до таблиці «organizations» та формування листа. Метод «setFrom» приймає адресу електронної пошти налаштованого SMTP-серверу, з якого буде відправлено лист. В свою чергу, метод «addAddress» приймає адресу на яку прийде лист,

тобто адресу, вказану у запиті. Методи «subject» і «body» відповідають за тему та зміст листа відповідно. Метод «send» відправляє листа.

Після успішної відправки, на пошту прийде лист (рис. 3.4) із сформованим, за допомогою токена, посиланням на створення нового паролю.

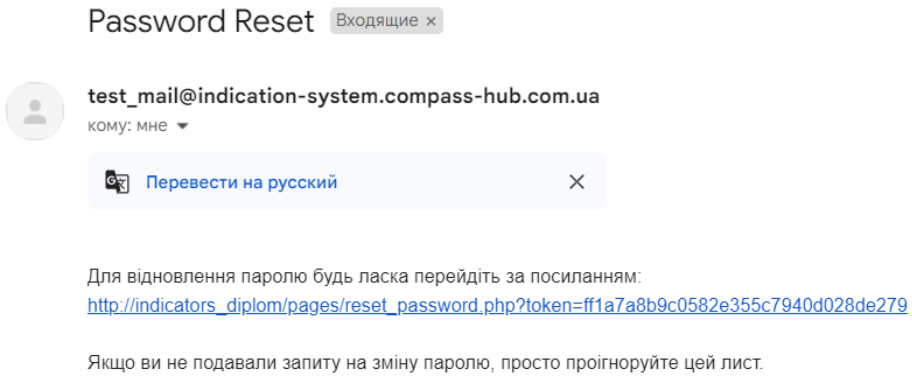

Рисунок 3.4 – Лист відновлення паролю для входу

При переході за посиланням, вказаним у листі, відкриється сторінка створення нового паролю для входу в обліковий запис користувача (рис. 3.5).

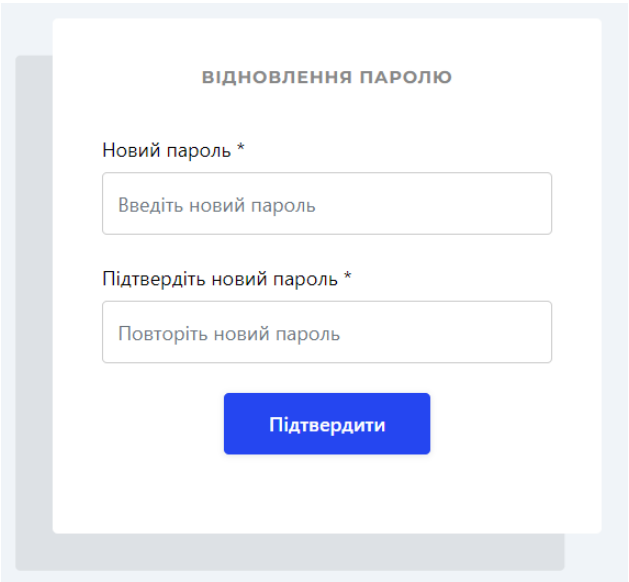

Рисунок 3.5 – Інтерфейс форми відновлення паролю

У даній формі розташовані два поля, в які користувач повинен ввести новий пароль. Але варто відзначити, що перед відображенням форми, відпрацьовується код, який перевіряє актуальність токену:

```
$token = $_GET["token"];
$token_hash = hash("sha256", $token);
$organization = Organization::where('reset_token_hash', $token_hash)-
>get();
foreach ($organization as $single_organization) {
    if(strtotime($single_organization->reset_token_expires_at)<=time())
        die("token has expired");
}
```

Отримавши із посилання значення переданого токену завдяки змінній «$_GET», генеруємо хеш токену і якщо він співпадає із хешем, який зберігається у БД, отримується колекція об’єктів, яка складається із об’єкту знайденого запису. Далі перевіряється актуальність токену, порівнявши відмітку часу у БД до якої діє токен із поточним значенням часу. У разі закінчення терміну дії, функція «die» перериває подальшу генерацію форми відновлення паролю.

При натисканні на кнопку підтвердження, у серверній частині веб-додатку ще раз перевіряється збігання токену і його термін дії, після чого перевіряється співпадіння паролю у полях форми:

```
if($_POST["password"] !== $_POST["password_confirmation"]){
    echo "passwords must match";
    die("passwords must match");
}
```

Якщо паролі не збігаються, подальшого відновлення паролю не відбувається і користувач повинен ще раз ввести новий пароль. У разі проходження перевірки, оновлюється запис у таблиці «organizations» в базі даних, а саме оновлюється пароль та анулюються стовпці, які відповідають за збереження токену відновлення та відмітки часу:

```
$single_organization->password = password_hash($_POST['password'], PASSWORD_BCRYPT);
$single_organization->reset_token_hash = NULL;
$single_organization->reset_token_expires_at = NULL;
$single_organization->save();
```

При успішному оновлені даних, користувач повертається на сторінку авторизації.

3.3 Блок адміністрування обліковим записом

Авторизувавшись у веб-застосунку, користувач переходить до сторінки адміністрування обліковим записом (рис. 3.6), на якій він матиме змогу керувати налаштуваннями, набором індикаторів та персоналом. Все це відбувається завдяки функціональним кнопкам та блокам, розташованим на веб-сторінці.

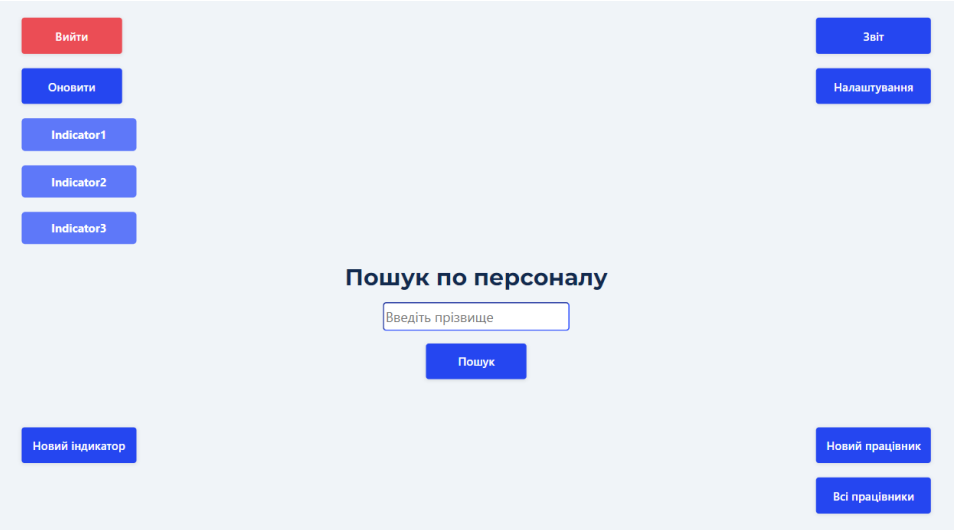

Рисунок 3.6 – Інтерфейс сторінки адміністрування обліковим записом

У верхньому лівому кутку можна побачити дві кнопки «Вийти» та «Оновити», які відповідають за завершення сеансу у веб-додатку та оновлення інформації на сторінці, відповідно. При натисканні на кнопку «Вийти» запускається код у серверній частині веб-додатку, який анулює cookie-файл,

який містить дані про активну сесію, та перенаправляє користувача на сторінку авторизації:

```
setcookie('organization', $organization->id, time()-3600 * 24 * 30, "/");
header("Location: /");
```

Нижче розташовано блок, в якому відображені індикатори якості роботи у порядку їх створення користувачем. Реалізація даного рішення виконана наступним чином:

```
if(isset($_COOKIE['organization'])){
    $organization = Organization::find($_COOKIE['organization']);
?>
<div class="indicators_block">
    <?php foreach ($organization->indicators as $indicator){ ?>
        <div class="indicator btn-ind">
<a href="indicator_fill_page.php?id=<?php echo $indicator->id; ?>"><?=
$indicator->name; ?></a>
        </div>
    <?php } ?>
</div>
```

Спочатку відбувається перевірка на наявність cookie-файлу, завдяки якому користувальницька сесія є активною. Якщо перевірка пройшла успішно, відбувається пошук запису у БД в таблиці «organizations» статичним методом «find» по ідентифікатору, збереженому до cookie-файлу, та створюється екземпляр класу із даними, отриманими із бази даних.

Далі, маючи доступ до пов’язаних з обліковим записом індикаторів якості, які зберігаються у БД, цикл «foreach», проходиться по кожному з них. На кожній ітерації циклу створюється розмітка кнопки індикатору із посиланням на відповідну сторінку з GET-запитом, який містить ідентифікатор індикатору, та назвою, яка відображається на самому елементі.

По центру сторінки адміністрування (рис. 3.6) розташовано блок пошуку по персоналу, в якому, за допомогою поля для введення прізвища робітника,

з’явиться елемент у вигляді кнопки (рис. 3.7), завдяки якому можна перейти до відповідної сторінки.

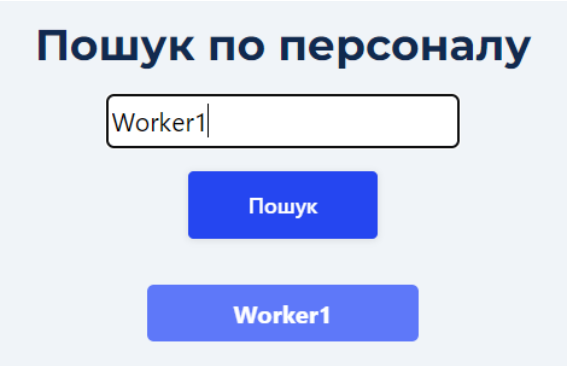

Рисунок 3.7 – Інтерфейс блоку пошуку

Даний блок є формою з одним текстовим полем. При натисканні на кнопку «Пошук» відбувається створення глобальної змінної $_POST і перезавантаження сторінки для подальшого опрацювання даних. Маючи дану змінну, в якій зберігається рядок, який було введено у поле пошуку, відбувається пошук по базі даних і подальше виведення результатів:

```
<?php if(isset($_POST['search_worker'])){
$found_workers = Worker::where('name', 'like', '%' .
$_POST['search_worker'] . '%') -> get(); ?>
    <div class="workers_container">
        <?php foreach ($found_workers as $worker){ ?>
            <div class="worker_unit">
                <a href="view_worker.php?id=<?php echo $worker->id; ?>"
class="worker_unit_text"><?= $worker['name']; ?></a>
            </div> <?php } ?>
    </div> <?php } ?>
```

Якщо змінна $_POST була створена, робиться запит до БД, в якому, в таблиці «workers», відбувається пошук записів, які мають у стовпці «name» рядок, який збігається із даний, який було передано до змінної $_POST формою. У разі знаходження одного або декількох запитів, створюється колекція об’єктів класу «Worker», яка зберігається у змінній «$found_workers».

Далі, в колекції, яку отримали, цикл «foreach», опрацьовує кожний об’єкт, внаслідок чого, створюється розмітка елементу-кнопки робітника із посиланням на відповідну сторінку з GET-запитом, який містить ідентифікатор працівника у таблиці «workers», та ПІБ, який відображається на самому елементі.

У правому верхньому кутку розташовані кнопки «Звіт» і «Налаштування». Перша відповідає за запуск скрипту генерації звіту, а друга відкриває сторінку налаштувань облікового запису користувача (рис. 3.8).

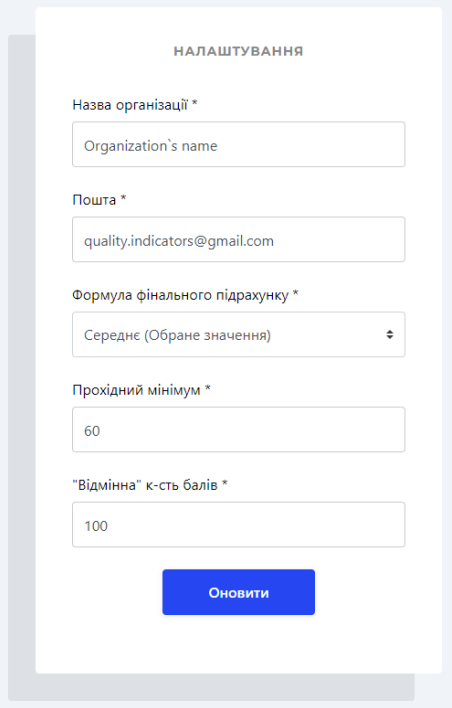

Рисунок 3.8 – Інтерфейс форми налаштувань облікового запису

Налаштування створені у вигляді форми, завдяки якій можна оновлювати дані облікового запису. Вона складається із полів, в яких, використовуючи інформацію з бази даних, підставляються збережені значення назви організації, електронна пошта, формула фінального підрахунку, число для прохідного мінімуму та «відмінна» кількість балів. Наприклад, дані про адресу електронної пошти відображаються наступним чином:

```
<input value="<?=$organization->email; ?>" >
```

У даному рядку коду зображена передача значення із БД до атрибуту «values», завдяки чому поле у формі перегляду даних є вже заповненим.

Останні два поля відповідають за числові значення, які служать межами оцінювання задля створення більш зрозумілого та інформативного вигляду фінальної Excel-таблиці, а саме завдяки даним значенням, скрипт визначає чи є загальна оцінка роботи працівника допустимою, відмінною або недостатньою.

Задля зміни даних користувач повинен ввести потрібні значення у поля та натиснути на кнопку «Оновити». Після цього на серверній частині веб-додатку запуститься скрипт оновлення даних у БД:

```
$organization = Organization::find($_COOKIE['organization']);
$organization->name = $_POST['name'];
$organization->email = $_POST['email'];
$organization->final_count_formula = $_POST['formula'];
$organization->needed_points = $_POST['needed_points'];
try {
    $organization->save();
    header("Location: ../pages/view_organization.php");
} catch (Exception $e) {
echo "Виникла помилка при зміні даних або організація з такою адресою
пошти вже існує"; }
```

На першому рядку у даного фрагменті коду, завдяки cookie-файлу, проводиться пошук запису у таблиці «organizations», після чого його дані присвоюються об’єкту класу. Далі, змінюються атрибути об’єкту та блок try-catch намагається зберегти введені дані. Варто відзначити те, що скрипт не дозволить оновити адресу електронної пошти на ту, котра вже є у іншому обліковому записі.

У нижній частині сторінки адміністрування розташовано кнопки «Новий індикатор», «Новий працівник» та «Всі працівники», які відповідають за керування набором індикаторів та персоналом у веб-додатку.

3.4 Керування персоналом

Натиснувши на кнопку «Новий працівник» на сторінці адміністрування веб-додатком, користувач попаде на сторінку створення запису про нового працівника у базу даних (рис. 3.9). Процес внесення даних реалізовано у вигляді форми.

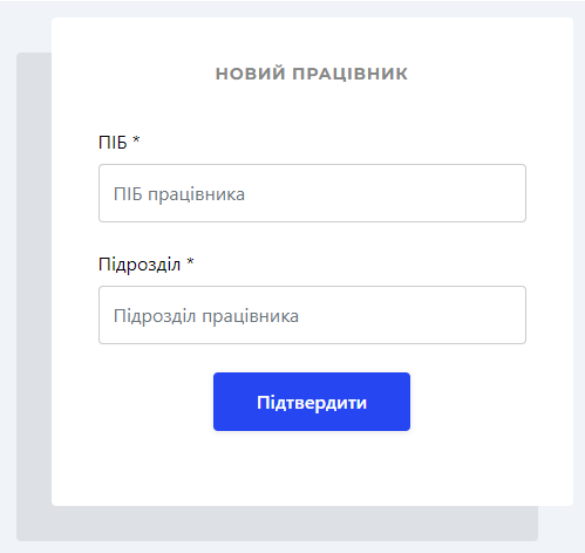

Рисунок 3.9 – Інтерфейс форми введення даних нового працівника

Структура форми є стандартною для веб-додатку. При натисканні на кнопку «Підтвердити», введені дані передаються на сервер, де вони обробляються та додаються до БД у вигляді нового запису:

```
$organization = Organization::find($_COOKIE['organization']);
$worker = new Worker();
$worker->organization_id = $organization->id;
$worker->name = $_POST['name'];
$worker->subdivision = $_POST['subdivision'];
```

У першому рядку коду, завдяки cookie-файлу, скрипт знаходить запис організації у таблиці «organizations», ідентифікаційний номер котрої буде введено у відповідний стовпчик запису у таблиці «workers», що можна

побачити на третьому рядку. Потреба у цьому виникає через наявність зв’язку «один-до-багатьох» між таблицями «organizations» та «workers». На другому рядку коду створюється екземпляр класу «Worker», після чого атрибути об’єкту заповнюються значеннями із форми.

Подальший процес збереження даних у блоці «try-catch» виглядає наступним чином:

```
$organization->workers()->save($worker);
header("Location: ../pages/view_worker.php?id=$worker->id");
```

Збереження даних про новий запис у таблицю виконується використовуючи метод «workers» прописаного у класі «Organization» для роботи із зв’язком «один-до-багатьох». Виконуючи вищеописаний метод, відбувається передача до методу «save» безпосередньо об’єкту «$worker», атрибути якого будуть збережені у таблиці в базі даних.

Далі відбувається перенаправлення користувача на сторінку перегляду інформації про новостворений запис працівника (рис. 3.10). Даний перехід виконується завдяки створенню посилання із GET-змінною «id», яке містить ідентифікаційний номер запису у БД.

Перейшовши на сторінку перегляду даних, в першу чергу перевіряється наявність GET-змінної для подальшого відображення інформації:

```
if(isset($_GET['id'])){
    $worker = Worker::find($_GET['id']);
```

Після успішної перевірки потрібних даних, створюється екземпляр класу «Worker», який має значення атрибутів відповідні до стовпців запису у таблиці із переданим ідентифікатором. Далі поля форми заповнюються отриманими значеннями:

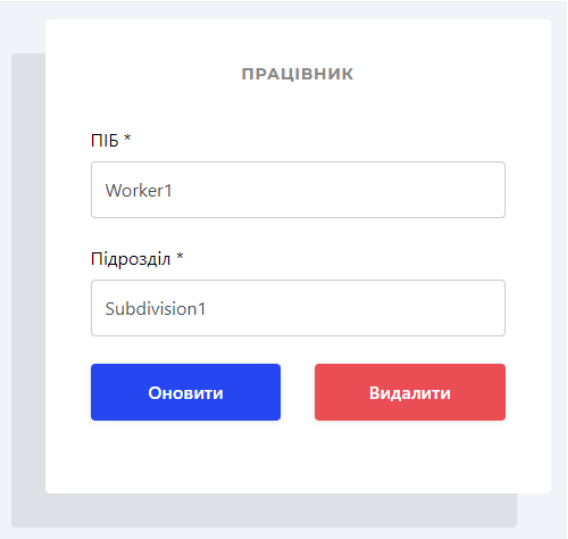

Рисунок 3.10 – Інтерфейс форми перегляду даних працівника

У даній формі, разом із кнопкою «Оновити», яка працює аналогічно із подібною кнопкою на формі перегляду налаштувань облікового запису користувача, розташована кнопка «Видалити». Завдяки даній кнопці є можливість коректно видаляти записи із бази даних. При натисканні на дану кнопку, до серверної частини веб-додатку надходить GET-змінна, яка міститься у посиланні до скрипту видалення:

```
<a href="../logic/delete_worker.php?id=<?php echo $worker->id; ?>" >
```

Видалення запису із таблиці «workers» реалізовано наступним чином:

```
$worker = Worker::find($_GET['id']);
$worker->delete();
```

Завдяки GET-змінній, яка містить ідентифікатор, створюється екземпляр класу «Worker», через пошук у БД за ідентифікатором. Даний екземпляр відповідає запису у таблиці, тому, використовуючи метод «delete» об’єкт видаляється разом із ним.

Повернувшись на сторінку адміністрування веб-додатком, користувач має можливість подивитись список усіх працівників, записи о яких було

внесено до бази даних, натиснувши на кнопку «Всі працівники». Перейшовши на сторінку керування персоналом, можна побачити розподіл по підрозділам, назви яких було введено при внесені даних о робітниках, та елементи на яких відзначені ПІБ працівників (рис. 3.11).

Рисунок 3.11 – Інтерфейс сторінки керування персоналом

Структуру даної сторінки було реалізовано завдяки роботі з даними із таблиці «workers». Розподіл персоналу по підрозділам було реалізовано наступним чином:

```
$organization = Organization::find($_COOKIE['organization']);
$subdivisions_array = array();
foreach ($organization->workers as $worker){
    if(!in_array($worker->subdivision, $subdivisions_array))
        $subdivisions_array[] = $worker->subdivision; }
```

У першому рядку даного фрагменту коду створюється екземпляр класу «Organization» із інформацією про організацію із переданим з cookie-файлу ідентифікаційним номером для подальшої роботи із працівниками, які відносяться до даної організації. Після цього, створюється масив «$subdivisions_array», який зберігатиме унікальні значення підрозділів, які

містяться у записах працівників у таблиці «workers». На третьому рядку коду можна побачити цикл foreach, який проходиться по кожному екземпляру класу працівника організації. Далі, за допомогою умови і використання функції «in_array» перевіряється чи є вже у створеному масиві така назва підрозділу, яка є у атрибуті екземпляру класу «$worker->subdivision». Якщо такої назви немає, вона переходить у масив «$subdivisions_array». По завершенню циклу, масив має у собі всі унікальні значення назв підрозділів. Принцип алгоритму пошуку унікальних назв підрозділів можна представити блок-схемою (рис. 3.12).

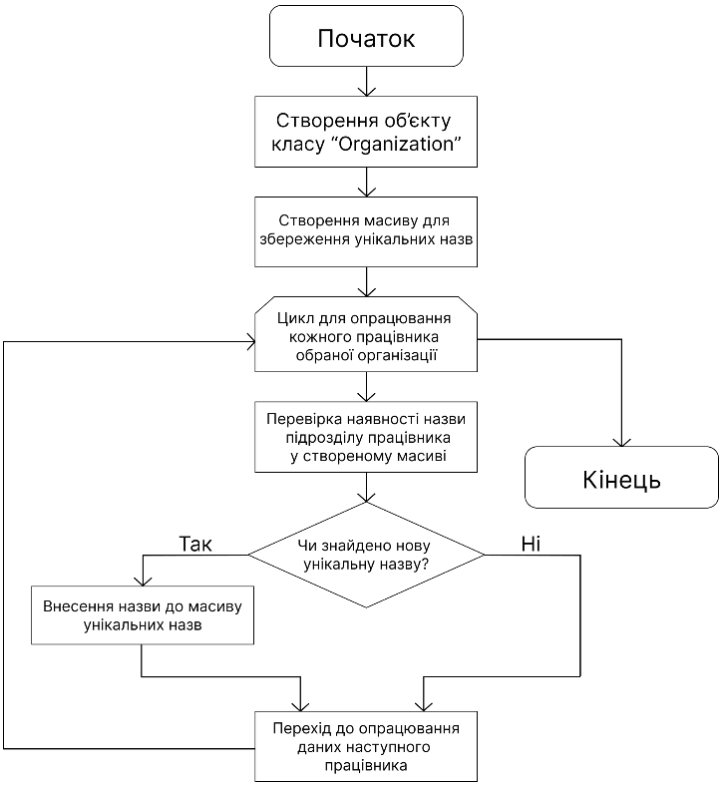

Рисунок 3.12 – Блок-схема алгоритму пошуку унікальних назв підрозділів

Далі, для побудови структури розмітки використовується цикл foreach, який опрацьовує кожний елемент у масиві назв підрозділів, створюючи div-контейнер, який буде наповнений елементами працівників даного підрозділу.

Пошук відповідних працівників та їх виведення реалізовано наступним чином:

```
<?php
$found_workers = Worker::where('subdivision', $subdivision)->get();
foreach ($found_workers as $worker){ ?>
    <div class="worker_link">
<a class="worker_text" href="view_worker.php?id=<?php echo $worker->id;
?>"><?=$worker['name']; ?></a>
    </div>
<?php } ?>
```

На другому рядку коду зображено запит до БД, в якому виконується пошук працівників організації із зазначеною назвою підрозділу, яка відповідає поточній ітерації циклу foreach масиву назв. Далі, отримавши колекцію працівників, запускається ще один цикл foreach, який опрацьовує кожний об’єкт «$worker» у колекції «$found_workers». Отримавши дані о працівникові, створюється div-блок, який містить ПІБ працівника із посиланням на перегляд даних саме цього робітника, використовуючи GET-змінну «id» із ідентифікатором з таблиці. Натиснувши на такий елемент, відбувається перехід на відповідну сторінку працівника.

3.5 Керування індикаторами та заповнення даних

Натиснувши на кнопку «Новий індикатор» на сторінці адміністрування веб-додатком, користувач попаде на сторінку створення запису про новий індикатор якості у базі даних (рис. 3.13). Процес внесення даних реалізовано у вигляді форми.

Структура форми є стандартною для веб-додатку. У ній користувач вводить усі необхідні дані для створення нового індикатору якості роботи працівників: назву, мінімальне значення та максимальне. Далі, в залежності від потреби вводиться значення прохідного мінімуму, яке відповідає за мінімальну межу виконаного плану індикатору якості, визначеного для робітників.

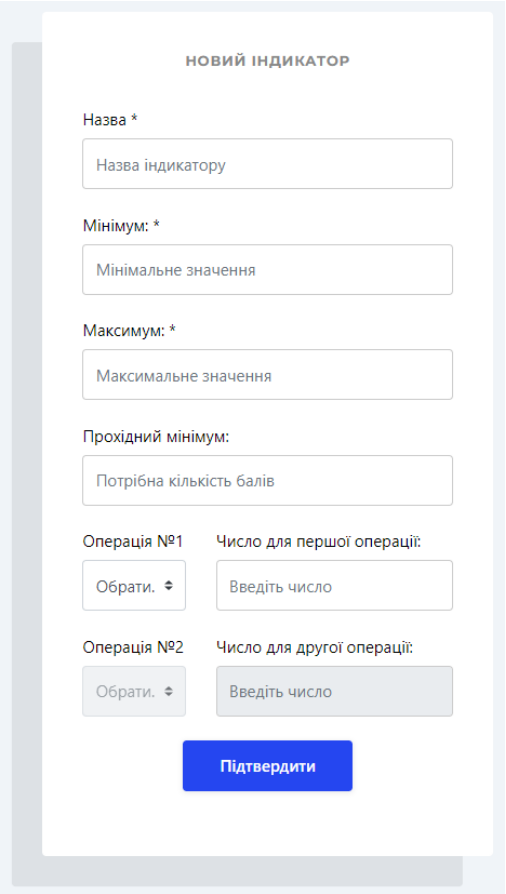

Рисунок 3.13 – Інтерфейс форми створення індикатору якості

Наступні дві групи полів відповідають за опрацювання, введеного у подальшому, значення якості роботи, а саме математичний підрахунок у вигляді вибору дій та введення чисел (рис. 3.14) для створення формул для роботи із фінальним значенням якості роботі у згенерованій Excel-таблиці.

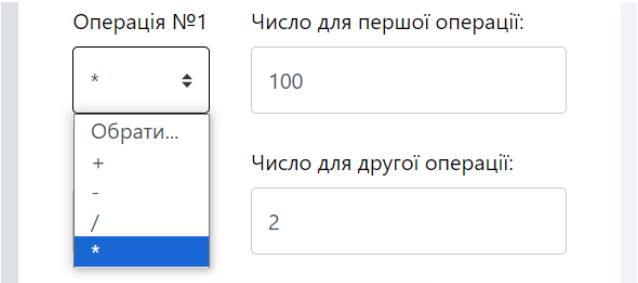

Рисунок 3.14 – Демонстрація роботи із групою полів для налаштування формули підрахунку значення індикатору якості

Перше поле у кожній з двох груп реалізовано у вигляді спадаючого списку із значенням математичних операцій. Друге поле є числовим для введення значень для подальших обчислень. За замовченням значення даного поля є «0». Як можна побачити (див. рис. 3.12) друга група полів, яка відповідає за другу операцію над значенням індикатору кості, є неактивною, але до тих пір, поки не буде заповнена перша група, яка відповідає за першу операцію. Все це було реалізовано задля коректної роботи над обчисленнями і збереженням інформації до бази даних. У вигляді коду дана реалізація виглядає наступним чином:

```
if($(".input_operation1").val() === ""){
    $(".input_operation2,     .input_operation_digit2").attr("disabled",
    "disabled")}
$(".input_operation1").on("change", function() {
    if(this.value !== ""){
        $(".input_operation2,
        .input_operation_digit2").removeAttr('disabled'); }
    else{
        $(".input_operation2,
        .input_operation_digit2").attr("disabled","disabled"); }
    });
```

Це було реалізовано за допомогою фреймворку jQuery. У першій умові описано те, що якщо значення у числовому полі першої групи є пустим, друга група полів приймає значення «disabled», що не дає змогу працювати з ними. На третьому рядку коду описано реакцію на подію зміни значення у числовому полі першої групи, а саме якщо було зафіксовано подію зміни значення у полі на будь-яке крім відсутності числа, то завдяки методу «removeAttr», значення «disabled» у другої групи полів анулюється, що надає змогу працювати з ними. У іншому випадку, коли значення у числовому полі стало або залишилось пустим, друга група полів знов блокується.

Після успішного заповнення форми даними та натискання на кнопку «Підтвердити», запит на створення запису у таблицю «indicators» переходить на серверну частину. В більшості, процес передачі даних до БД є аналогічним

із процесами при створені запису про нового працівника або реєстрації нового облікового запису, але є деякі особливості. Першою особливістю є визначення порядкового номеру індикатору серед інших у таблиці «indicators». У кожної зареєстрованої організації у базі даних є свій створений унікальний набір індикаторів і коректне ведення їх порядкових номерів є запорукою успіху коректного збереження і опрацювання даних при видаленні того чи іншого індикатору без виникнення проблем у подальшому. За це відповідає поле «count_name», тобто назва із порядковим номером у таблиці «indicators». Визначення даної назви реалізовано наступним чином:

```
$indicator = new Indicator();
$indicator_count = 1;
foreach ($organization->indicators as $indicator){
    $indicator_count++; }
$indicator->count_name = "indicator_".$indicator_count;
```

На першому рядку коду зображено створення нового об’єкту класу «Indicator». Відлік порядкового номеру індикатора починається з одиниці, як це показано на другому рядку. Після цього, використовуючи цикл «foreach», опрацьовуємо кожний вже створений індикатор та додаємо одиницю до лічильника при кожній ітерації. По завершенню роботи циклу отримуємо порядковий номер та додаємо його до строки із значенням «indicator» задля присвоєння індикатору назви із порядковим номером, яка у подальшому, буде відігравати свою роль при видалені індикаторів з облікового запису.

Другою особливістю передачі інформації до бази даних є валідація числових значень перед збереженням запису. Це створено для запобігання некоректної роботи веб-додатку при обчисленні даних о якості роботи та генерації звіту. Наприклад, валідація мінімального значення оцінки якості роботи працівника виглядає наступним чином:

```
if($_POST['min'] >= 0 && $_POST['min'] <= 1000000) {
    $indicator->min = $_POST['min'];
}
```

У даному фрагменті коду йдеться про те, що значення мінімуму, яке передається з форми, повинно бути більше або дорівнювати нулю та бути менше або дорівнювати одному мільйону. Таким чином усі негативні значення, нуль та занадто великі значення відкидаються. У разі успішного проходження валідації, дані з форми передаються до відповідного поля екземпляру класу. В іншому випадку вони не передаються, що призводить до спрацювання виключення у коді та повідомлені про це користувачу.

Після успішного створення нового запису про індикатор у базі даних, користувача буде направлено на новостворену сторінку перегляду даних (рис. 3.15).

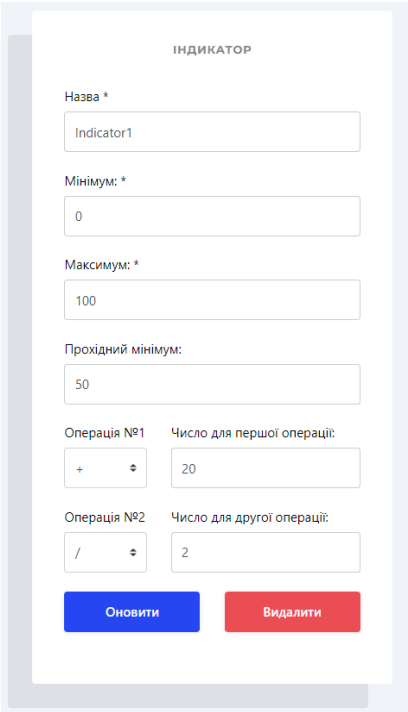

Рисунок 3.15 – Інтерфейс форми перегляду даних створеного індикатора

Кнопки «Оновити» та «Видалити» працюють аналогічним чином як у формі перегляду даних працівника. Процес оновлення даних також є типовим для веб-додатку і не відрізняється від аналогічного процесу при керуванні обліковим записом та персоналом. Але виключенням становить алгоритм видалення індикатору із таблиці «indicators». Архітектура веб-додатку

передбачає суворе ведення порядку індикаторів задля коректної роботи з ними при генерації звіту, що обумовлено наявністю п’ятьох стовпців у таблиці «workers», кожен з яких відповідає за свій індикатор оцінювання якості роботи. Тому при видалені одного з індикаторів, відповідний стовпець, порядковий номер якого співпадає з порядковим номером індикатору повинен бути очищений, а інші стовпці, за потреби, «здвинуті» для зміни свого порядкового номеру та збереження усіх своїх даних. Даний алгоритм було реалізовано наступним чином:

```
$deleted_indicator_count = $indicator->count_name;
$indicator->delete();
foreach ($organization->workers as $worker){
    $worker->$deleted_indicator_count = 0;
    $worker->save();
}
```

Як показано у першому рядку, у змінну «$deleted_indicator_count» зберігається назва із порядковим номером індикатору, після чого він сам видаляється. Потім, використовуючи цикл foreach, опрацьовуються дані робітників організації, а саме, маючи назву з порядковим номером індикатору, яка співпадає з назвою стовпця у таблиці «workers», в якому зберігаються дані про якість роботи по даному індикатору, анулюються значення пов’язані із видаленим індикатором.

```
$indicator_count = 1;
foreach ($organization->indicators as $indicator){
    $indicator_count_name_full = "indicator_".$indicator_count;
    $indicator_actual_count_name = $indicator->count_name;
    $indicator->count_name = $indicator_count_name_full;
    $indicator->save();
    foreach ($organization->workers as $worker){
        $worker->$indicator_count_name_full = $worker-
        >$indicator_actual_count_name;
        if($indicator_count_name_full != $indicator_actual_count_name){
            $worker->$indicator_actual_count_name = 0;}
        $worker->save(); }
    $indicator_count++; }
```

Продовжуючи алгоритм видалення, на першому рядку даного фрагменту коду створюється змінна-лічильник «$indicator_count» для участі у формуванні нових порядкових номерів індикаторів. На другому рядку запускається цикл foreach, який буде опрацьовувати кожний індикатор якості організації, який залишився. На третьому рядку у змінну «$indicator_count_name_full» зберігається нова назва із порядковим номером індикатору шляхом додавання рядка зі значенням «indicator» до змінної-лічильника. Це робиться на випадок того, якщо, при попередньому видаленні індикатору, порядкові номери збиваються, задля виправлення цього, порядок відновлюється без пробілів у ньому. На четвертому рядку у змінну «$indicator_actual_count_name» зберігається поточна назва із порядковим номером, яка може не збігатися з новою коректною. Після цього, коректна назва зберігається у відповідному записі у таблиці індикаторів і алгоритм переходить до виправлення даних у таблиці працівників. На сьомому рядку запускається новий цикл foreach, який опрацьовує дані персоналу організації.

У кожного працівника, на місці стовпця, який відповідав за дані вже видаленого індикатору, тобто нині маючого нульові значення, наразі будуть знаходитися дані коректного по номеру, існуючого індикатору, як це показано на восьмому рядку. На дев’ятому відбувається перевірка за наступною умовою: якщо поточний порядковий номер індикатору (підрахований за допомогою змінної-лічильника) не співпадає з номером, з яким зараз працюємо (у циклі foreach для індикаторів), то це означає, що код наткнувся на зміщення, тому стовпець у таблиці працівників, який відповідає, за, індикатор, який потрібно змістити (дані якого вже були успішно зміщені на восьмому рядку), потрібно анулювати, щоб на наступній ітерації він зміг прийняти значення наступного стовпця зі значення індикатора, які треба змістити, задля уникнення пропусків у порядку.

Таким чином відбувається перестановка значень для збереження порядку значень індикаторів якості у таблиці «workers». Після цього дані зберігаються і змінна-лічильник збільшується на одиницю задля роботи з

наступним індикатором. Саме цей алгоритм запобігає порушенню цілісності у відображені даних у звіті та у коректності їх зберігання у БД задля подальшої зручної їх інтерпретації для користувача. Після успішного відпрацювання алгоритму видалення, користувач потрапить на сторінку адміністрування веб-додатком. Принцип алгоритму видалення індикатору якості можна представити блок-схемою (рис. 3.16).

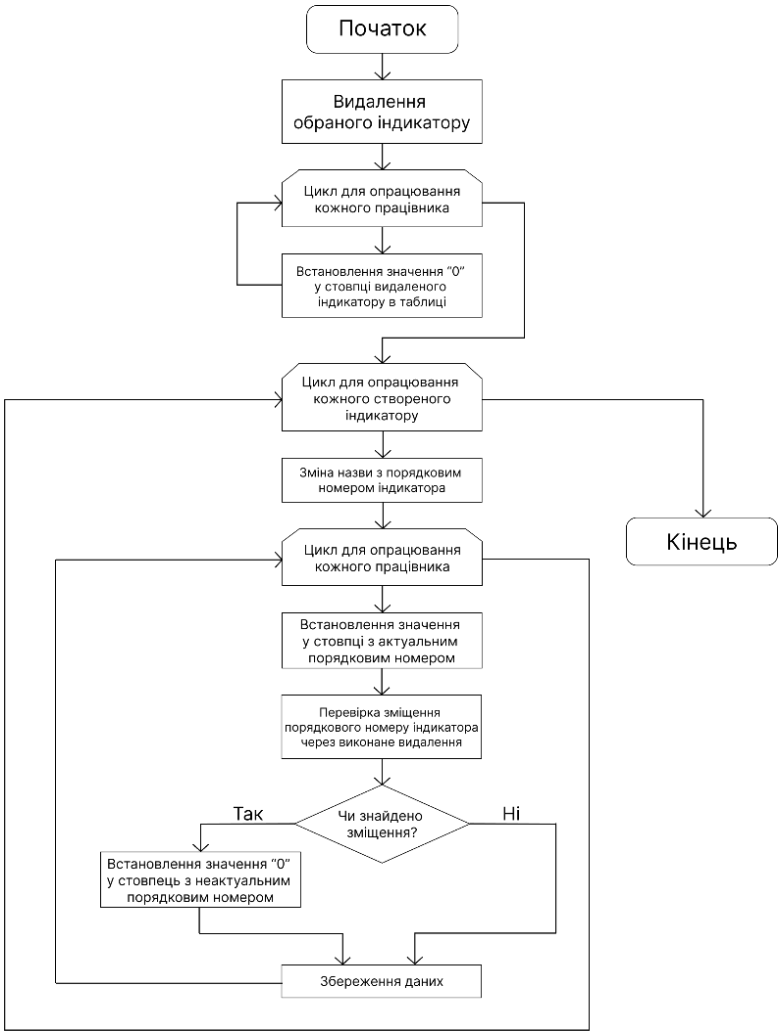

Рисунок 3.16 – Блок-схема алгоритму видалення індикатору якості

Розташовані зліва кнопки з назвами створених індикаторів дозволяють користувачеві потрапити на відповідну сторінку заповнення значень до таблиці індикатору якості роботи працівників (рис. 3.17), використовуючи GET-змінну із ідентифікатором у посиланні.

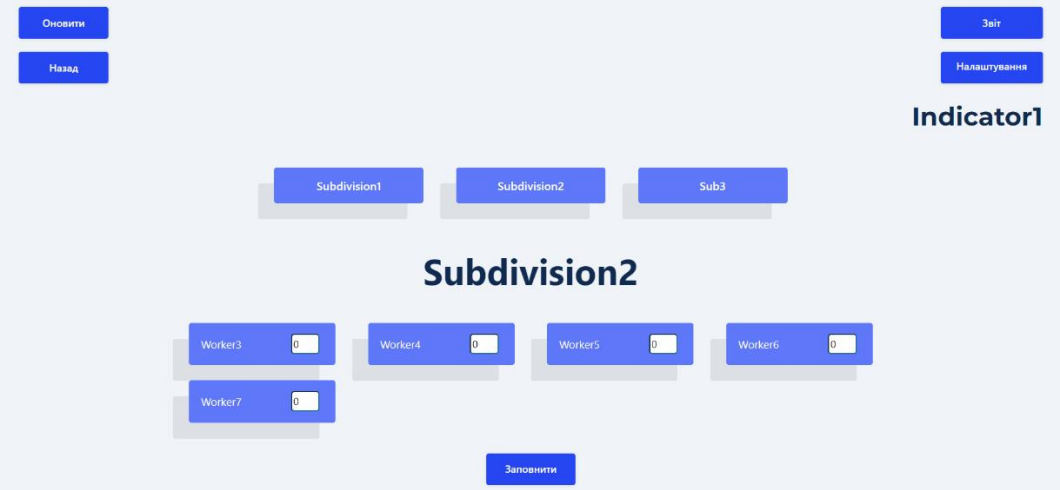

Рисунок 3.17 – Інтерфейс сторінки заповнення значень до таблиці індикатору якості роботи працівників

У лівій верхній частині знаходяться кнопки «Оновити» та «Назад», які відповідають за оновлення даних на сторінці та повернення на сторінку адміністрування відповідно.

У правій верхній частині знаходяться кнопки «Звіт» та «Налаштування». Перша дає можливість згенерувати Excel-звіт не повертаючись на головну сторінку, щоб відразу побачити зміни у ньому після внесення даних про якість роботи працівників. Друга кнопка направляє користувача на сторінку перегляду даних про обраний індикатор (див. рис. 3.15), на якій користувач матиме змогу передивитись налаштування підрахунку та внести зміни за потреби. Нижче цих двох кнопок розташовано назву обраного індикатору, яка була отримана завдяки GET-змінній, використавши яку, було отримано екземпляр класу «Indicator» з БД зі вказаним в змінній ідентифікатором.

По центру розміщено блок з переліком усіх унікальних підрозділів, які були вказані при створені записів о робітниках. Алгоритм отримання унікальних значень назв та подальше їх виведення на сторінку було детально описано при описі сторінки відображення персоналу, а саме, завдяки створенню масиву та циклу foreach, в якому відбираються унікальні значення назв підрозділів в записах працівників, було реалізовано даний функціонал.

Унікальною особливістю даного алгоритму, який реалізовано на сторінці заповнення значень є те, що кожна назва підрозділу відображена у вигляді кнопки, при натисканні на яку, перезавантажується сторінка і до змінної ідентифікатору індикатору у GET-запиті додається ще одна змінна «subdivision», яка відповідає за назву підрозділу за яким буде відбуватись заповнення значень:

```
<a href="indicator_fill_page.php?id=<?php echo $indicator->id;?>&subdivision=<?php echo $worker->subdivision;?>"><?= $worker->subdivision; ?></a>
```

У атрибуті «href» можна побачити посилання із двома змінними у GET-масиві.

Отож, при натисканні на ту чи іншу кнопку із назвою підрозділу, відкривається список працівників, пов’язаних із ним. Конструкція цього списку створена у вигляді форми, де кожний елемент працівника має поле, а при натисканні на кнопку «Заповнити» відбувається відправка даних на сервер. Структура кожного елементу із ПІБ працівника і полем зі значенням відповідним до обраного індикатора виглядає наступним чином:

```
<div class="worker ">
    <p class=" worker_text"><?=$worker->name ?></p>
<input type="text" class="hidden_id" name="<?=$hidden_input_id; ?>" value="<?=$worker->id ?>" >
    <div class="main_input">
<input type="number" class="input_points" name="<?=$hidden_input_value; ?>" min="<?=$indicator->min; ?>" max="<?=$indicator->max; ?>" value="<?=$current_points; ?>" step="0.01">
    </div>
</div>
```

У div-контейнері містяться тег «p» який відображає ПІБ працівника, приховане поле із класом «hidden_id», яке містить ідентифікатор поля у вигляді порядкового номеру у назві, який заповнюється автоматично та значення ідентифікатору поточного користувача, для зручності в обробці на

серверній частині і ще одне поле зі значенням поточної оцінки якості роботи працівника за обраним індикатором. Варто відзначити і те, що в останньому полі присутня валідація у вигляді мінімального та максимального значення, які отримуються із таблиці «indicators» у записі обраного індикатора, представлені у вигляді атрибутів «min» та «max».

Атрибути «name», тобто назви, кожного поля заповнюються автоматично. Генерація даних назв виконується при створені елементу працівника:

```
$indicator = Indicator::find($_GET['id']);
$indicator_count_name = $indicator->count_name;
$count = 1;
$found_workers = Worker::where('subdivision', $_GET['subdivision'])-
>get();
foreach ($found_workers as $worker){
    $current_points = $worker->$indicator_count_name;
    $hidden_input_id = "input".$count;
    $hidden_input_value = "input_value".$count;?>
```

Використовуючи значення ідентифікатору у GET-масиві, створюється об’єкт індикатору якості, після чого у змінну «$indicator_count_name» передається його назва із порядковим номером. На третьому рядку створюється змінна-лічильник «$count» для подальшого формування назви полів. Наступним кроком є отримання колекції об’єктів працівників, виконавши пошук у таблиці по назві обраного підрозділу. Далі цикл foreach опрацьовує кожного працівника із отриманої колекції «$found_workers». Змінна «$current_points» отримує значення оцінки якості роботи, використовуючи порядковий номер індикатора зі змінної «$indicator_count_name». На сьомому та восьмому рядках відбувається створення назв для прихованого поля ідентифікатору та для поля, яке містить поточну оцінку працівника. Завдяки об’єднанню рядка зі значенням «input» та числа у змінній «$count», створюються унікальні назви для полів, завдяки яким дані з них можна буде з легкістю опрацювати на серверній частині

додатку. Після даного блоку коду генерується вищеописана розмітка елементу працівника.

Після натискання на кнопку «Заповнити», усі введені значення опрацьовуються на сервері:

```
$indicator_count_name = $indicator->count_name;
$count = 1;
foreach ($found_workers as $worker){
    $input_id = "input".$count;
    $input_value = "input_value".$count;
    $worker = Worker::find($_POST[$input_id]);
    $worker->$indicator_count_name = $_POST[$input_value];
    $count++;
```

Спочатку змінна «$indicator_count_name» отримує значення порядкової назви обраного індикатору. Дані про індикатор та обраний підрозділ отримуються із GET-масиву зі змінними ідентифікатору і назви підрозділу. Далі було створену змінну-лічильник «$count». Створений на третьому рядку цикл foreach опрацьовує запис про кожного працівника обраного підрозділу. У четвертому і п’ятому рядках конструюються назви полів аналогічно із тим, як це було реалізовано на веб-сторінці, після чого, використовуючи статичний метод «find», створюється екземпляр класу «Worker» із значенням ідентифікатору з прихованого поля у формі. Атрибут класу, який призначений за збереження даних про оцінку якості роботи по обраному індикатору, отримує значення із відповідного поля у формі, як це зображено у сьомому рядку.

Після завершення циклу блок «try-catch» зробить спробу зберегти переданні значення.

3.6 Генерація звіту

Розташована на сторінках адміністрування веб-додатком і заповнення значень кнопка «Звіт», при натисканні на неї, генерує Excel-файл із

підрахованими значеннями індикаторів якості роботи та фінальними значеннями по кожному робітнику. Результатом роботи даного скрипту є завантажений у браузері Excel-файл (рис. 3.18).

	A	B	C	D	E
1	**Працівник**	**Indicator1**	**Indicator2**	**Indicator3**	**Фін. підрахунок**
2	Worker3	70	35	80	61,67
3	Worker4	63	50	36,25	49,75
4	Worker5	67	30	98,75	65,25
5	Worker6	70	80	92,5	80,83
6	Worker7	70	100	105	91,67
7					

Subdivision1 | **Subdivision2** | Sub3

Рисунок 3.18 – Демонстрація згенерованого Excel-звіту

Генерація даного Excel-файлу складається з двох частин: налаштування стилів таблиці та заповнення комірок значеннями.

Першим кроком є створення екземпляру класу «Spreadsheet» із бібліотеки «PhpSpreadsheet» і надання пустого Excel-файлу на сервері для створення подальшої копії та її наповнення:

```
$spreadsheet = new Spreadsheet();
$inputFileName = 'excel_file.xlsx';
$spreadsheet_copy = \PhpOffice\PhpSpreadsheet\IOFactory::load
($inputFileName);
```

Наступним кроком є створення структури Excel-файлу. Як можна побачити на демонстрації звіту (див. рис. 3.18), було згенеровано три аркуша, кожен з яких відповідає унікальному значенню назви підрозділу. Процес знаходження унікальних назв у записах у таблиці «workers» є однаковим із алгоритмами, які використовуються на сторінках перегляду персоналу і заповнення даних про оцінки якості роботи працівників. Маючи вже готовий масив унікальних назв підрозділів, цикл foreach опрацьовує кожен із них і на кожній ітерації, разом із генерацією стилів, аркуш наповнюється

опрацьованими даними. На початку роботи із аркушем підрозділу, скрипт заповнює перші комірки у кожному стовпці:

```
$count = 0;
foreach ($subdivisions_array as $subdivision){
    $spreadsheet->createSheet();
    $spreadsheet->setActiveSheetIndex($count);
    $sheet = $spreadsheet->getActiveSheet();
    $sheet->setTitle($subdivision);
    $sheet->getColumnDimension('A')->setWidth(23);
    $sheet->setCellValue('A1', 'Працівник');
```

Метод «createSheet», використаний на третьому рядку створює новий аркуш у Excel-файлі, після чого метод «setActiveSheetIndex» зі вказаним числом, яке у нашому випадку відповідає лічильнику «$count», робить аркуш активним, що дає змогу працювати з ним. Створена, на п’ятому рядку, змінна «$sheet» стає екземпляром класу «Spreadsheet», після чого, використовуючи метод «setTitle» присвоює назву аркушу, а саме назву підрозділу. Далі, метод «getColumnDimension» зі вказаним параметром «A» встановлює ширину колонці «A» у аркуші за допомогою методу «setWidth». На останньому, у даному фрагменті коду, рядку, метод «setCellValue» встановлює для комірки «A1» значення «Працівник», що робить колонку «A» колонкою для ПІБ працівників.

Далі заповнюються заголовки наступних колонок, які призначені для значень індикаторів:

```
$start_char = 'B';
foreach ($organization->indicators as $indicator){
    $sheet->getColumnDimension($start_char)->setWidth(15);
    $sheet->setCellValue($start_char.'1', $indicator->name);
    ++$start_char; }
```

Цикл foreach опрацьовує кожен створений індикатор і на кожній ітерації вказує ширину колонки, використовуючи у ролі лічильника текстову змінну «$start_char», яка наприкінці кожної ітерації змінюється на наступну літеру

абетки, таким чином значення першого індикатору будуть розташовані у колонці «B», значення другого у «C» тощо. У четвертому рядку даного фрагменту коду, кожній першій комірці у стовпчику присвоюється значення назви поточного індикатору якості.

Останньою є колонка «Фінального підрахунку»:

```
$sheet->getColumnDimension($start_char)->setWidth(15);
$sheet->setCellValue($start_char.'1', 'Фін. підрахунок');
```

Після розташування назв індикаторів, створений текстовий ітератор «$start_char» приймає значення літери наступної колонки, яке визначається після створення останньої колонки індикатору якості.

Далі скрипт встановлює стилі для різних рядків і колонок: розмір шрифту та його жирність, положення тексту в комірці, наявність меж та колір заливки. Наприклад, встановлення стилів для перших комірок у кожному стовпчику виглядає наступним чином:

```
$sheet->getStyle('A1:'.$start_char.'1')->applyFromArray([
    'font' => [
        'size'  => 12,
        'bold' => true,
    ],
    'alignment' => [
        'horizontal' => Alignment::HORIZONTAL_CENTER,
    ]
    ]);
```

Використання методу «getStyle», дозволяє вибрати потрібні комірки для додавання стилів, таким чином були обрані перші комірки з першого по останній стовпчик. Далі, метод «applyFromArray» дає змогу прописати усі потрібні стилі у вигляді масиву. Як можна побачити у даному фрагменті коду, розмір шрифту було встановлення на значення «12» та виділено жирним, а також вказано розташування тексту по центру.

Після налаштування стилів для комірок, скрипт заповнює таблицю даними про робітників, а саме ПІБ та оцінки якості роботи по кожному індикатору, обраховуючи їх значення. Заповнення першої колонки текстовими значеннями ПІБ працівників реалізовано наступним чином:

```
$row = 2;
foreach ($found_workers as $worker){
    $sheet->setCellValue('A'.$row, $worker->name);
```

На кожній ітерації циклу foreach відбувається підрахунок значень якості роботи працівника. Під час кожної ітерації запускається ще один цикл foreach, який працює із кожним індикатором якості організації. Далі скрипт перевіряє чи вказано у налаштуваннях індикатора операції над значенням. У разі наявності таких, запускається «switch-case» блок. Наприклад, якщо обрано значення множення, то виконується наступний код:

```
switch ($indicator->operation1) {
case "*":
     try {
        $value *= $indicator->operation_digit1;
        break;
    } catch (Exception $e) {
echo "Проблема з проведенням операції " . $indicator->operation1 . " " .
$indicator->operation_digit1 . " зазначенної у параметрах індикатора " .
$indicator->name;
    }
```

У кожному випадку відпрацьовує «try-catch» блок, який намагається виконати потрібну математичну дію. На четвертому рядку необроблене значення оцінки якості роботи, як вказано у прикладі множення, помножується на вказане у налаштуваннях індикатора число. Якщо це не вдається зробити, викликається помилка з докладним описом.

Якщо у налаштуваннях індикатора вказано іншу операцію, тоді викликається інший блок «switch-case».

Після обчислення значення, воно потрапляє до відповідної комірки:

```
$sheet->setCellValue($start_char_another . $row, $value);
```

В даному рядку коду змінна «start_char_another» відповідає за літеру колонки, яка ітеративно змінюється при переході на кожний новий індикатор у циклі, а змінна «$row» за число, яке ітеративно змінюється на кожному новому працівникові.

Маючи таблицю заповнену значеннями, скрипт починає зафарбовувати комірки в залежності від налаштувань того чи іншого індикатора. Таким чином, якщо не вказано потрібний мінімум, то неможливо оцінити якість роботи, але в протилежному випадку, спрацьовує ряд умов, по яким знаходиться відповідний колір комірки:

```
if($indicator->needed_min != null){
    if($value >= $indicator->needed_min){
        $color = '92D050';
    }
    elseif ($value < $indicator->needed_min && $value != 0){
        $color = 'FF7575';
    }
    elseif ($value == 0 && $indicator->operation1 == null){
        $color = 'D9D9D9';
    }
 }
```

На першому рядку даного фрагменту коду перевіряється атрибут «needed_min» на наявність значення. Якщо перевірка пройдена, тобто значення є, обраховане числове значення «$value» починає перевірятись за трьома умовами. У першому випадку, якщо значення дорівнює або є більшим за значення потрібного мінімуму, змінна «$color» приймає значення зеленого кольору у кодуванні «HEX». У другому випадку, якщо значення оцінки є меншим за значення потрібного мінімуму та не дорівнює нулю, тобто оцінка незадовільна, змінна «$color» приймає значення червоного кольору. У третьому випадку, якщо значення є нульовим та потрібний мінімум не

визначено, змінна «$color» приймає значення сірого кольору, що означає нестачу інформації для оцінювання.

Визначивши колір заливки, скрипт зафарбовує поточну комірку:

```
$sheet->getStyle($start_char_another.$row)
    ->getFill()
    ->setFillType(\PhpOffice\PhpSpreadsheet\Style\Fill::FILL_SOLID)
    ->getStartColor()
    ->setARGB($color);
```

Дана послідовність методів означає створення суцільної заливки комірки кольором вказаним у методі «setARGB».

Наприкінці ітерації циклу, який опрацьовує індикатори якості, у кожного працівника, в залежності від вказаного методу фінального підрахунку оцінки якості роботи, відбувається додавання поточної оцінки якості по індикатору для подальшого обчислення фінального значення:

```
if($organization->final_count_formula == "Зважені"){
    $final_value += $value * $indicator->weighted_sum;
}
else{
    $final_value += $value; }
```

Якщо у налаштуваннях облікового запису вказаний метод фінального підрахунку є «Метод зважених сум», то до фінального значення додається множення значень індикатора із його «вагою», вказаною в його налаштуваннях. В протилежному випадку, коли методом фінального підрахунку є «Середнє арифметичне», тоді значення індикатору, без зайвих операцій, додається до фінального значення. Після виконання даних умов, починається наступна ітерація та скрипт обробляє новий індикатор якості. Після виконання усіх ітерацій циклу, в першому випадку фінальне значення є готовим, а у другому випадку воно ділиться на кількість створених індикаторів задля отримання середнього арифметичного.

Після підрахунку, в останню колонку у рядку поточного робітника вводиться числове значення. Якщо у налаштуваннях облікового запису вказано потрібний мінімум, який потрібно набрати, перевіряється умова, за якою: якщо фінальне значення дорівнює або більше за вказаний мінімум, у змінну «$final_color» передається значення світло-зеленого кольору у форматі «HEX». В протилежному випадку, якщо значення менше, комірка зафарбовується червоним кольором. Якщо у налаштуваннях окрім потрібного мінімуму вказано кількість балів для визначення «відмінної» кількості, спрацьовує аналогічна умова в якій перевіряється чи достатня кількість балів, при проходженні якої змінна «$final_color» отримує значення насичено-зеленого кольору. По закінченню, цикл переходить на нову ітерацію і починає обробляти значення наступного працівника.

По завершеню опрацювання усіх даних, звіт буде готовий до скачування:

```
$writer = new Xlsx($spreadsheet);
$writer->save('Звіт.xlsx');
```

У даному фрагменті коду змінна «$writer» є екземпляром класу «Xlsx», після чого, використовуючи метод «save» файл зберігається на сервері.

Наступним та фінальним кроком є завантаження звіту у браузері користувача, що реалізовано наступним чином:

```
$file = 'Звіт.xlsx';
if (file_exists($file)) {
    header('Content-Description: File Transfer');
    header('Content-Type: application/octet-stream');
    header('Content-Disposition:                                    attachment;
filename="'.basename($file).'"');
    header('Expires: 0');
    header('Cache-Control: no-cache, must-revalidate');
    header('Pragma: public');
    header('Content-Length: ' . filesize($file));
    readfile($file);
    exit; }
```

Даний набір заголовків визначає параметри завантаження файлу, а саме є класичним для файлу малих розмірів. Алгоритм генерації Excel-звіту можна представити блок-схемою (рис. 3.19).

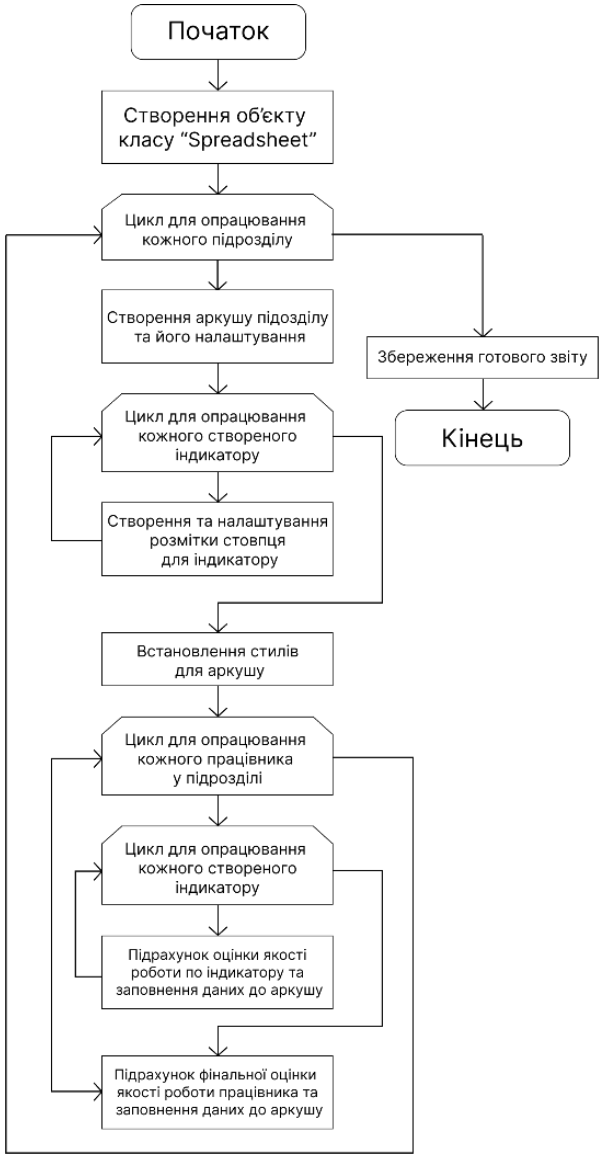

Рисунок 3.19 – Блок-схема алгоритму генерації Excel-звіту

ВИСНОВКИ

В результаті виконання дипломної роботи було здійснено проектування та програмну реалізацію веб-додатку «Електронна система індикаторів якості». Розробка даного веб-додатку надає можливість організаціям відстежувати показники якості роботи працівників, використовуючи індикатори якості. Все це дасть змогу оптимізувати робочі процеси і побачити слабкі та сильні сторони своєї компанії, що позитивно відобразиться на успішності та продуктивності.

В ході дипломного проектування було проведено аналіз існуючих аналогів, після чого були виявлені їх переваги та недоліки, що дозволило сформувати вимоги до розробки веб-додатку «Електронна система індикаторів якості». Також у ході цього була визначена архітектура системи, здійснено вибір та обґрунтування програмних засобів реалізації, проведено проектування системи та розроблена база даних веб-додатку.

Після цього система була реалізована, використовуючи обраний набір технологій, а саме: технології HTML, CSS, JS та фреймворки Bootstrap і jQuery для роботи із клієнтською частиною, а також мова програмування PHP, фреймворк Laravel, технології Vagrant, Homestead і СУБД MySQL для роботи із серверною частиною. Реалізований веб-додаток «Електронна система індикаторів якості» має зручний та адаптивний інтерфейс, зрозумілі форми з даними, що забезпечує зручність роботи з системою користувачеві, а налагоджений функціонал дозволяє гнучко налаштовувати операції для підрахунку якості роботи і отримувати Excel-звіти з точно підрахованими даними та зрозумілою розміткою.

Звіти на основі конкретних індикаторів забезпечують об’єктивну оцінку продуктивності кожного працівника. Це допомагає уникнути суб’єктивності та упередженості в оцінках, оскільки індикатори чітко визначають, які результати вважаються успішними. Звіти можуть мотивувати працівників, оскільки вони отримують чітке розуміння того, як їхня робота оцінюється і що

вони повинні робити для досягнення кращих результатів. Регулярний зворотній зв'язок допомагає утримувати високий рівень залученості та зацікавленості в роботі. На основі таких звітів керівництво може приймати обґрунтовані рішення щодо винагород, підвищень, навчання або необхідності дисциплінарних заходів. Звіти оцінки якості роботи працівників за налаштованими індикаторами є невід’ємною частиною ефективного управління персоналом і веб-додаток «Електронна система індикаторів якості» дає змогу ефективно працювати з ними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nikita Nikitin. «web application for calculating and maintaining quality indicators». 64. Konferencja Studenckich Kół Naukowych Pionu Górniczego AGH, no.64, 2023, pp. 221.
2. Сучасна платформа управління робочими процесами. URL: https://www.smartsheet.com/ (дата звернення 12.03.2024)
3. Introducing Quip Spreadsheets. URL: https://quip.com/blog/spreadsheets (дата звернення 12.03.2024)
4. Розробка з боку Front end – що це та чим відлічається від Back end. URL: https://dan-it.com.ua/blog/razrabotka-so-storony-front-end-chto-jeto-takoe-i-chem-otlichaetsja-ot-back-end/ (дата звернення 25.03.2024)
5. Що таке фреймворк: пояснюємо простими словами. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-poyasnyuyemo-prostymy-slovamy (дата звернення 11.04.2024)
6. Що таке Bootstrap?. URL: https://w3schoolsua.github.io/bootstrap/bootstrap_get_started.html#gsc.tab=0 (дата звернення 15.04.2024)
7. Що таке Bootstrap? URL: https://itmaster.biz.ua/programming/web-prohramuvannia/bootstrap.html (дата звернення 15.04.2024)
8. Як працює адаптивний веб-дизайн. URL: https://dizz.in.ua/uk/yak-praczyu%D1%94-adaptivnij-veb-dizajn/ (дата звернення 19.04.2024)
9. Основи JavaScript. URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics (дата звернення 22.04.2024)
10. Що таке скрипт та скриптова мова. URL: https://proseo.kiev.ua/stati/shcho-take-skrypt-ta-skryptova-mova/ (дата звернення 23.04.2024)
11. jQuery – Урок 1. Що таке jQuery, основні поняття і можливості. URL: https://m.site-do.ru/js/jquery1.php (дата звернення 25.04.2024)

12. Що таке backend-розробка і чим вона відрізняється від frontend. URL: https://mc.today/uk/shho-take-backend-rozrobka/ (дата звернення 26.04.2024)
13. AJAX для новачків. URL: https://habr.com/ru/post/14246/ (дата звернення 26.04.2024)
14. PhpSpreadsheet. URL: https://phpspreadsheet.readthedocs.io/en/latest/ (дата звернення 26.04.2024)
15. Eloquent: Початок роботи. URL: https://laravel.su/docs/10.x/eloquent (дата звернення 26.04.2024)
16. Active Record. URL: https://www.wikiwand.com/uk/Active_Record (дата звернення 26.04.2024)
17. Що таке база даних. URL: https://www.oracle.com/ua/database/what-is-database/ (дата звернення 26.04.2024)
18. База даних MySQL. URL: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення 26.04.2024)
19. Що таке СУБД і для чого вони потрібні. URL: https://foxminded.ua/systema-upravlinnia-bazamy-danykh/ (дата звернення 26.04.2024)
20. Що таке локальний сервер і для чого він потрібен. URL: https://programming.in.ua/web-design/allphp/68-local-serv-priority.html (дата звернення 26.04.2024)
21. Laravel детальніше про Homestead. URL: https://habr.com/ru/articles/328880 (дата звернення 26.04.2024)
22. What is Vagrant. URL: https://www.geeksforgeeks.org/what-is-vagrant/ (дата звернення 26.04.2024)
23. What is PhpStorm. URL: https://monovm.com/blog/what-is-phpstorm/ (дата звернення 26.04.2024)