

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебсайт бухгалтерської компанії для комунікації з
клієнтами»**

Виконав студент групи ІІЗ-20-3
СКОКІВСЬКИЙ Володимир Владиславович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
САВІЦЬКИЙ Роман Святославович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СКОКІВСЬКИЙ Володимир Владиславович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Вебсайт бухгалтерської компанії для комунікації з клієнтами»**,
затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**

Вихідні дані для роботи: **використання веборієнтованих технологій для
розробки, проектування та реалізація вебсайту бухгалтерської компанії для
комунікації з клієнтами**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що
їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю. М.	27.02.2024р.	27.02.2024р.
2	Єфремов Ю. М.	27.02.2024р.	27.02.2024р.
3	Єфремов Ю. М.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебсайту бухгалтерської компанії для комунікації з клієнтами і пояснювальної записки. Пояснювальна записка до випускної роботи містить 63 сторінок, 44 ілюстрацій, 1 таблицю, 3 додатки та 26 літературних джерел.

Метою випускної кваліфікаційної роботи є розробка вебдодатку, що дозволить бухгалтерській компанії ефективно взаємодіяти зі своїми клієнтами. Вебдодаток буде створено на основі фреймворку Symfony, що забезпечить високу продуктивність, масштабованість та безпеку системи. Додаток буде спрямований на покращення комунікації між бухгалтерською компанією та її клієнтами.

Інтерфейс вебдодатку буде інтуїтивно зрозумілим і зручним для користувачів різного рівня технічної підготовки. Він забезпечить швидкий доступ до основних функцій, таких як перегляд статей, можливість задати питання консультантам та відгукам компанії.

КЛЮЧОВІ СЛОВА: БУХГАЛТЕРІЯ, ВЕБДОДАТОК, SYMFONY, ПІДТРИМКА, КОНСУЛЬТАЦІЯ.

					ІПЗ.КР.Б-121-24-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Скоківський В. В.			Розробка вебсайту бухгалтерської компанії для комунікації з клієнтами	Літ.	Арк.	Аркуші
Керівник		Єфремов Ю. М.					2	
						Житомирська політехніка, група ІПЗ-20-3		
Н. контр.								
Зав. каф.								

ABSTRACT

The bachelor's thesis consists of a web application of an accounting company for communication with clients and an explanatory note. The explanatory note to the final thesis contains 63 pages, 44 illustrations, 1 table, 3 appendices and 26 literary sources.

The goal of this project is to develop a web application that will allow an accounting company to effectively interact with its clients. The web application will be built on the basis of the Symfony framework, which will ensure high performance, scalability and security of the system. The application will be aimed at improving communication between the accounting company and its clients.

The interface of the web application will be intuitive and convenient for users of different levels of technical training. It will provide quick access to the main functions, such as viewing articles, the ability to ask questions to consultants and reviews of the company.

KEY WORDS: ACCOUNTING, WEB APPLICATION, SYMFONY, SUPPORT, CONSULTING.

					ИПЗ.КР.Б-121-24-ИЗ	Арк.
						3

ЗМІСТ

РЕФЕРАТ	2
ABSTRACT	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБСАЙТУ БУХГАЛТЕРСЬКОЇ КОМПАНІЇ ДЛЯ КОМУНІКАЦІЇ З КЛІЄНТАМИ.....	7
1.1 Постановка задачі	7
1.2 Аналіз аналогів програмного продукту.....	8
1.3 Визначення архітектури вебдодатку	14
1.4 Обґрунтування вибору інструментальних засобів.....	16
Висновки до першого розділу.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ БУХГАЛТЕРСЬКОЇ КОМПАНІЇ ДЛЯ КОМУНІКАЦІЇ З КЛІЄНТАМИ.....	19
2.1 Визначення варіантів використання та структури системи	19
2.2 Розробка бази даних	24
2.3 Проектування та реалізація алгоритмів роботи додатку	26
2.4 Реалізація функціоналу додатку	28
Висновки до другого розділу	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ БУХГАЛТЕРСЬКОЇ КОМПАНІЇ ДЛЯ КОМУНІКАЦІЇ З КЛІЄНТАМИ.....	41
3.2 Структура інтерфейсу вебдодатку	42
3.2.1 Інтерфейс панелі адміністратора	45
3.2.2 Інтерфейс клієнта додатка	51
3.3 Тестування роботи вебдодатку	56
Висновки до третього розділу	59
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	62
ДОДАТКИ.....	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

IDE – Integrated Development Environment

JS – JavaScript

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

					ІПЗ.КР.Б-121-24-ПЗ	Арк.
						5

ВСТУП

Актуальність теми. В умовах постійних змін у розвитку бізнесу та швидких змін в економічному середовищі, бухгалтерські компанії постійно стикаються з рядом викликів, пов'язаних із необхідністю швидкого та ефективного обміну інформацією з клієнтами. Відкриття можливостей віддаленої роботи та зростання популярності онлайн-сервісів ставить під сумнів традиційні методи взаємодії.

Створення вебсайту для бухгалтерської компанії стає необхідністю для забезпечення ефективної взаємодії з клієнтами. Однією з ключових проблем, яку вирішує вебдодаток, є зручність та доступність інформації для клієнтів. Шляхом розміщення корисної інформації, такої як фінансові поради, новини з області оподаткування або зміни в законодавстві, компанія може підвищити свою цінність для клієнтів і стати для них незамінним партнером у фінансовому плануванні та обліку.

Це також відкриє нові можливості для підвищення впізнаваності бренду та просування послуг компанії, що приверне нових клієнтів та підвищить конкурентоспроможність на ринку.

Метою кваліфікаційної роботи є розробка вебдодатку для бухгалтерської компанії, який служитиме засобом комунікації з клієнтами. Додаток буде містити освітні статті з бухгалтерського обліку та оподаткування, а також форми зворотного зв'язку для клієнтів, які мають питання або потребують консультації.

Для реалізації поставленої мети потрібно виконати наступні завдання:

1. Провести аналіз вимог до функціональності додатку, враховуючи потреби користувачів.
2. Здійснити вибір засобів проектування.
3. Розробити вебдодаток для бухгалтерської компанії.

Об'єктом дослідження є розробка, проектування та реалізація вебдодатку бухгалтерської компанії для комунікації з клієнтами.

Предметом дослідження є використання веборієнтованих технологій для створення вебдодатку бухгалтерської компанії для комунікації з клієнтами.

					ІПЗ.КР.Б-121-24-ІЗ	Арк.
						6

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБСАЙТУ БУХГАЛТЕРСЬКОЇ КОМПАНІЇ ДЛЯ КОМУНІКАЦІЇ З КЛІЄНТАМИ

1.1 Постановка задачі

Нині у світі цифрових технологій наявність вебсайту для компанії є необхідністю. Сайт виступає як візитна картка компанії, інформаційний ресурс та платформа для взаємодії з клієнтами, він сприяє ефективному просуванню послуг та значно покращує комунікацію з клієнтами.

Першорядною метою створення додатку бухгалтерської компанії для комунікації з клієнтами є поліпшення обслуговування користувачів через забезпечення ефективного, швидкого та зручного процесу обміну інформацією між клієнтом та консультантами. Реалізація проекту дозволить клієнтам швидко знаходити відповіді на питання, які їх цікавлять, що допоможе їм зекономити свій час, а компанії підвищити ефективність своєї роботи.

Необхідно створити вебдодаток, де користувачі будуть мати можливість задати питання, залишити свої контакти для зв'язку з ними з метою консультації, перегляду актуальних новин у сфері бухгалтерії та залишити відгук про компанію за необхідності.

Для втілення поставленої задачі, основними фазами є:

1. Аналіз вимог та функцій вебдодатку.
2. Проектування та розробка вебдодатку:
 - Розробка архітектури;
 - Створення впорядкованого та легкого для навігації дизайну;
3. Розробка сторінок та функціоналу вебдодатку:
 - Модуль авторизації та реєстрації користувача;
 - Модуль форми для запитань;
 - Модуль форми для зв'язку;

- Модуль новин;
- Модуль коментування новин;
- Модуль відгуків;

4. Тестування вебдодатку.

В результаті виконання поставленого завдання є функціональний та естетичний вебсайт бухгалтерської компанії, який надає швидку та ефективну комунікацію з клієнтами. Вебдодаток має легкий для навігації інтерфейс, який сприяє швидкому доступу до необхідної інформації та послуг. Клієнти можуть легко звертатися за консультацією, отримувати актуальні новини та переглядати інформацію про послуги компанії.

1.2 Аналіз аналогів програмного продукту

Для розробки сайту було проведено дослідження конкуруючих вебдодатків, що дозволило оцінити конкурентне середовище та виявити їх сильні та слабкі сторони порівняно з нашим продуктом.

Для клієнтів завжди була актуальна проблема, щоб отримати консультацію з питань, які їх цікавлять у будь-який час і з будь-якого місця.

Для аналізу було обрано 3 аналоги:

- Platinum Finance;
- Paragraf;
- OblikUzh;

Першим аналогом є Platinum Finance[1]. Це консалтингова компанія, яка надає фінансові послуги для клієнтів (рис. 1.1). Їхній вебдодаток має сучасний дизайн та містить детальну інформацію про різноманітні фінансові послуги, включно з аудитом, бухгалтерським обліком та управлінням фінансами.

Основний розділ вебсайту присвячений опису та перевагам організації.

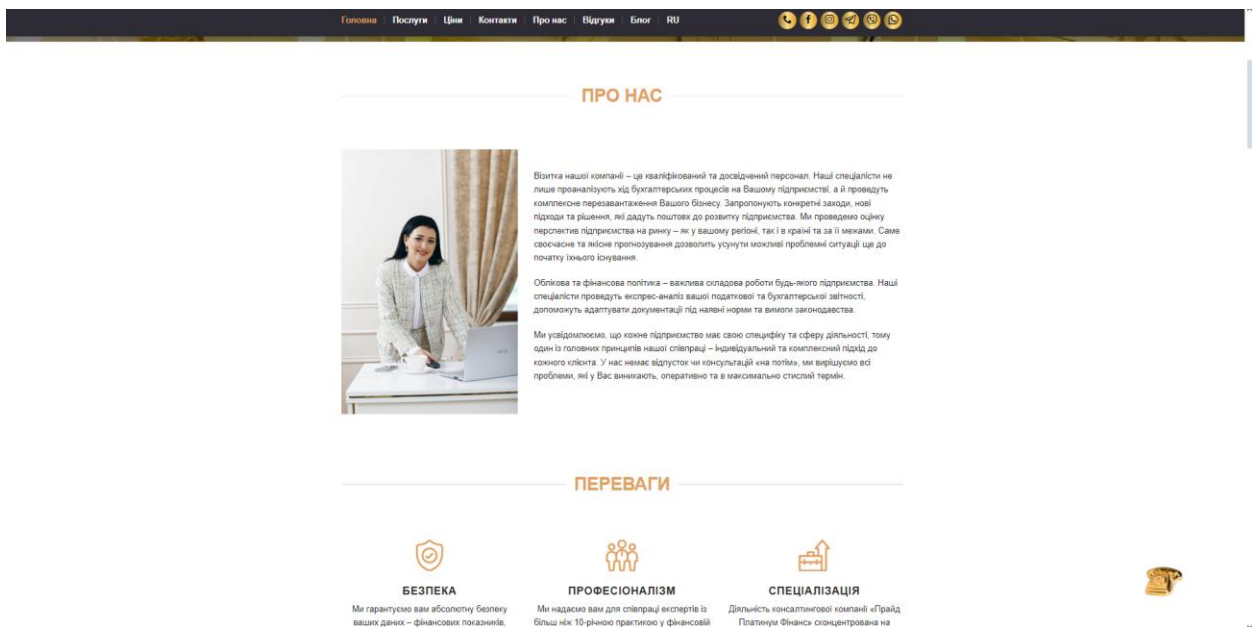


Рисунок 1.1 – Головна сторінка вебдодатку Platinum Finance

Наступний розділ зосереджений на описі послуг компанії(рис. 1.2), всі послуги детально описані, що дозволяє потенційним клієнтам з легкістю знайти послугу, яка їм потрібна та отримати консультацію через зручну форму запису на аудит.

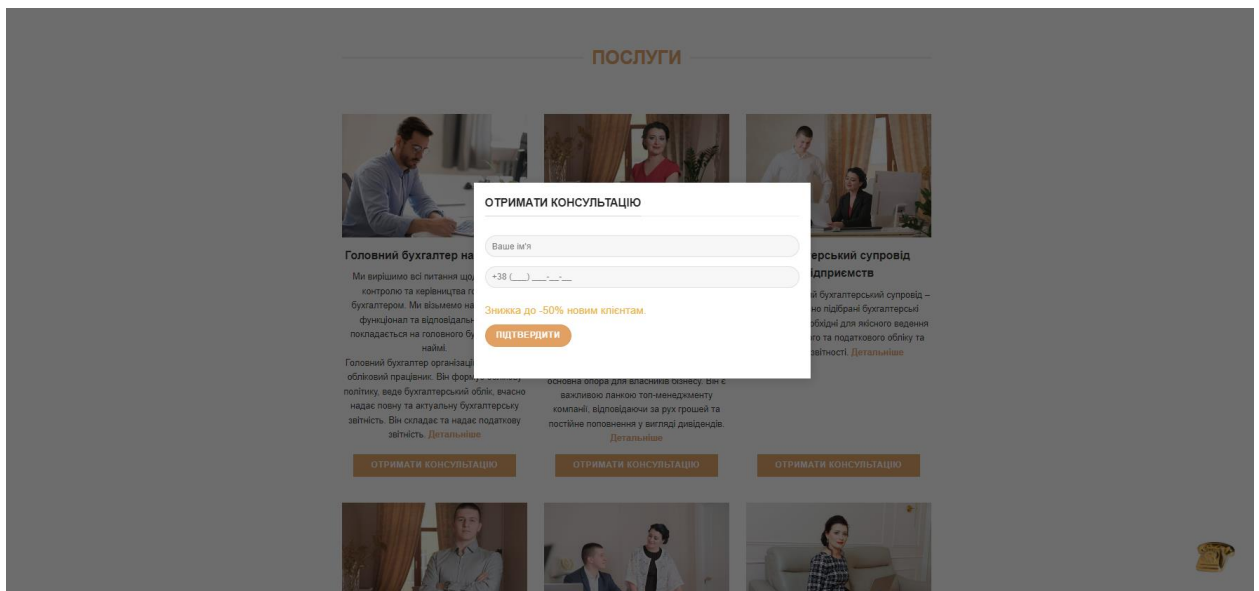


Рисунок 1.2 – Інформація про послуги та інтерактивна форма вебдодатку Platinum Finance

На сайті також є сторінка Блогу, де регулярно публікуються статті на актуальні теми фінансів, які можуть бути не тільки корисними, а і пізнавальними для відвідувачів додатку.

Вебсайт Platinum Finance має зручний та легкий у навігації інтерфейс, містить детально описані послуги та переваги компанії, що підвищує довіру нових клієнтів

Функціональність:

- Регулярні публікації на актуальні теми у сфері фінансів;
- Легкий доступ до консультаційних послуг компанії;

Другим аналогом є Paragraf[2] (рис. 1.3). Компанія пропонує велику різноманітність бухгалтерських послуг для бізнесу. Вебдодаток організації є достатньо зрозумілим та зручним у використанні. Інтерфейс сайту влаштований таким чином, щоб користувачі могли з легкістю знайти необхідну інформацію про послуги, які їх цікавлять.

На головній сторінці в розділі “Послуги” представлені варіанти з якими компанія може допомогти та проконсультувати.

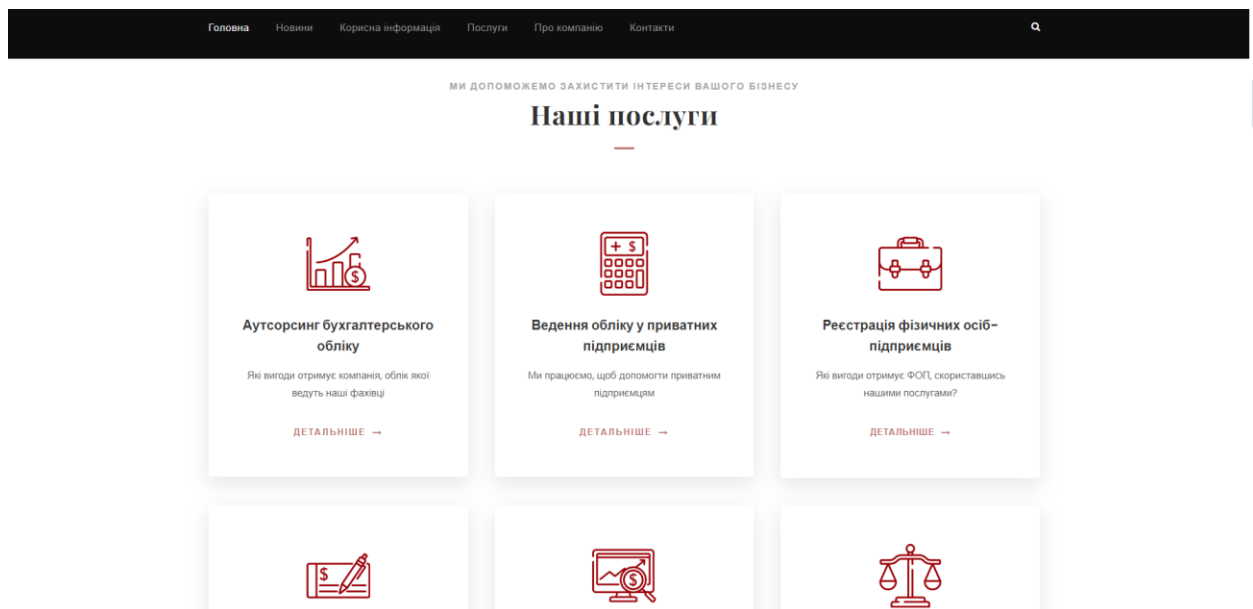


Рисунок 1.3 – Розділ “Послуги” у вебдодатку Paragraf

Одною з переваг додатку є сторінка “Корисна інформація” (див. рис. 1.4), де публікуються останні зміни в законодавстві пов’язані з фінансами, що може бути корисним для підприємців.

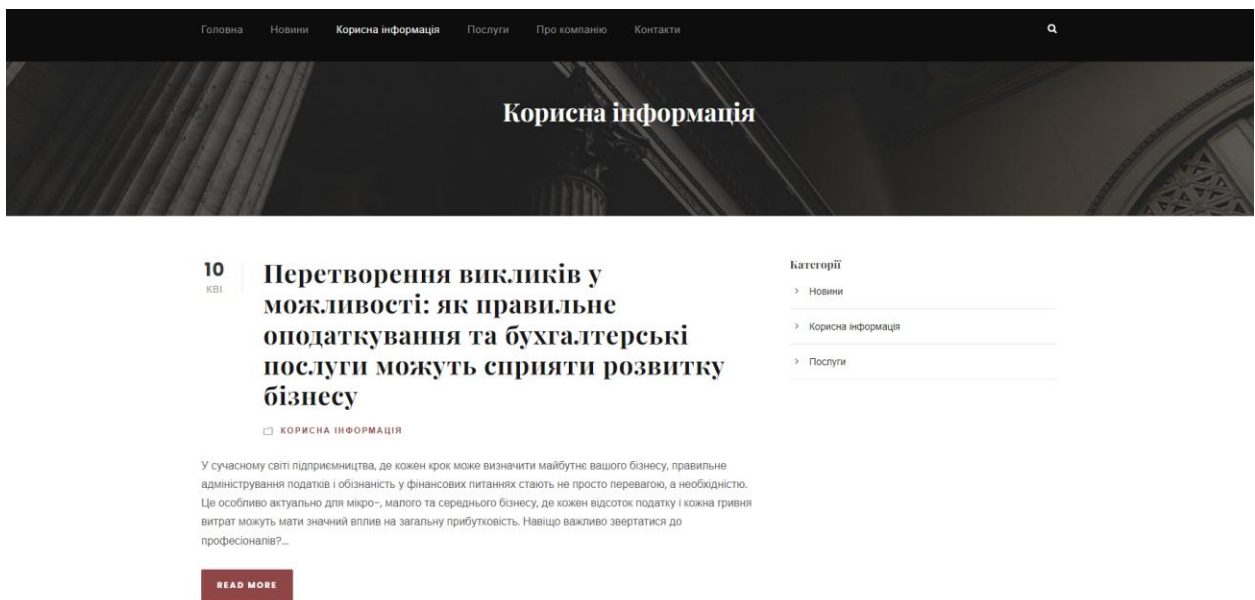


Рисунок 1.4 – Розділ “Корисна інформація” у вебдодатку Paragraf

Наявність у вебдодатку контактних форм є великим плюсом при користуванні і дозволяє швидко зв’язатись з представниками компанії для консультації.

Додаток Paragraf має зрозуміле і просте меню та навігацію, що дозволяє швидко знаходити необхідну інформацію. Функціональні можливості полегшують взаємодію користувачам.

Функціональність:

- Освітні матеріали для підприємців;
- Форми зворотного зв’язку;

Третім аналогом є Oblikuzh[3] (рис. 1.5). Компанія пропонує ведення бухгалтерського обліку, складання фінансової звітності, податкове планування та інше. Сайт хоч і має зручний у використанні та навігації інтерфейс, проте застарілий дизайн впливає на перше враження користувачів.

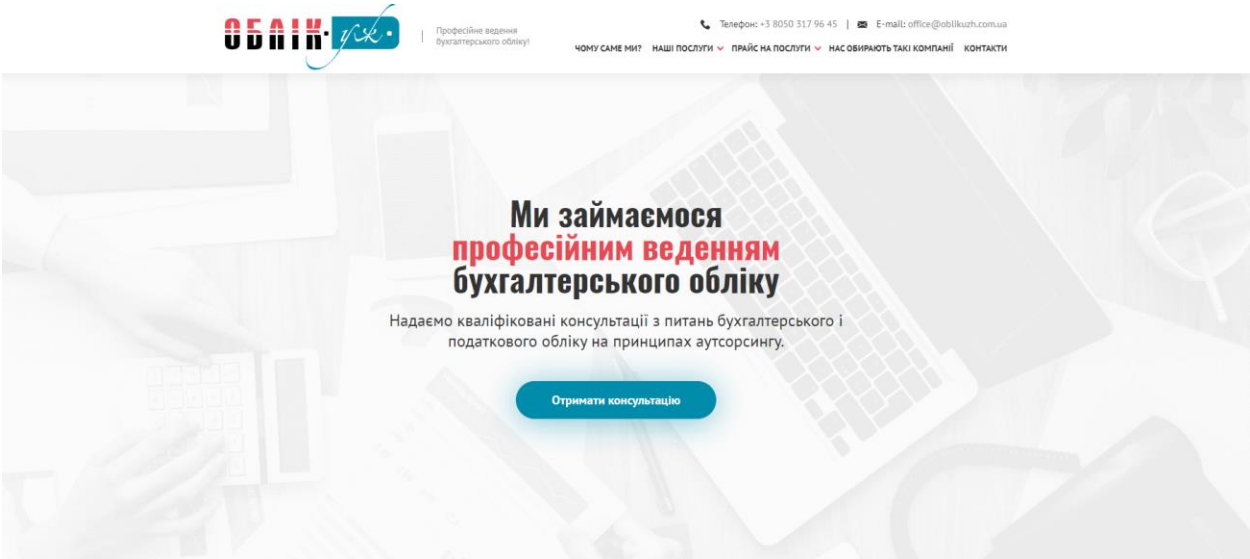


Рисунок 1.5 – Головна сторінка у вебдодатку Oblikuzh

Ще одним недоліком Oblikuzh у порівнянні з аналогічними додатками є відсутність форм зворотного зв’язку та можливості швидкої взаємодії з консультантами.

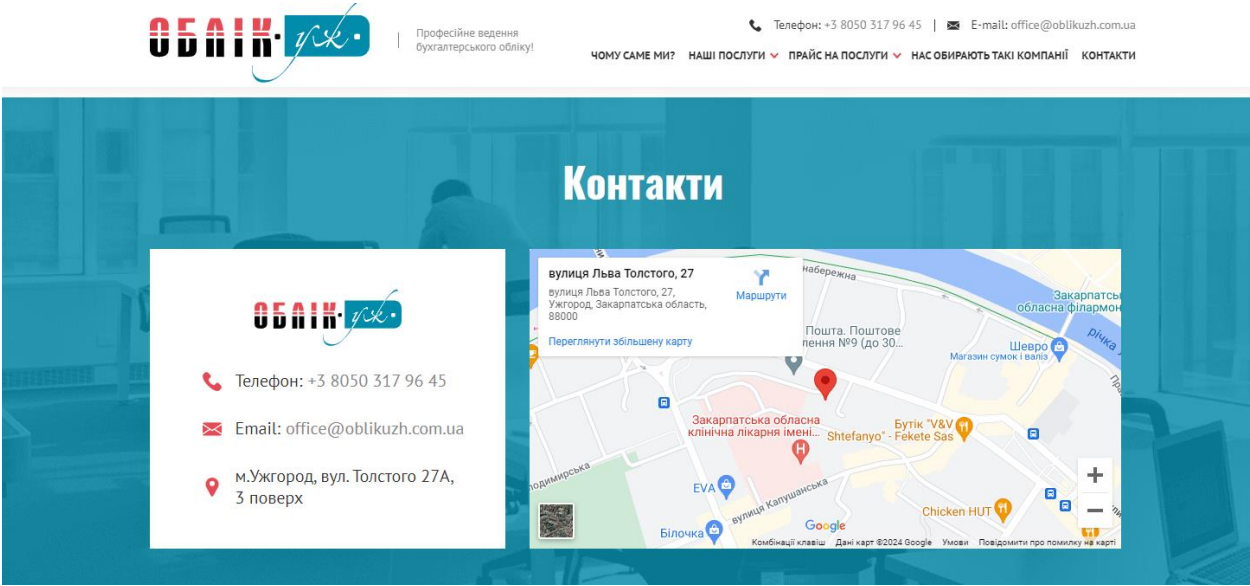


Рисунок 1.6 – Спосіб зв’язку у вебдодатку Oblikuzh

Додаток надає всю необхідну інформацію, яка потрібна користувачу, проте потребує суттєвого оновлення. Поліпшення дизайну, додавання інтерактивних елементів зробило б його більш привабливим для відвідувачів

Функціональність:

— Безкоштовний калькулятор податків по заробітній платі;

					ІПЗ.КР.Б-121-24-ПЗ	Арк.
						12

Для порівняння всіх описаних аналогів було створено таблицю (табл. 1.1)

Таблиця 1.1

Порівняння аналогів

Характеристика	Platinum Finance	Paragraf	Oblikuzh
Дизайн	Сучасний та привабливий	Сучасний та привабливий	Застарілий
Інтерфейс	Зручний	Зручний	Зручний
Наявність інтерактивних елементів	+	+	-
Адаптивність	+	+	+
Навігаційне меню	+	+	+
Наявність блог або статей	+	+	-
Відгуки клієнтів	+	-	-

Проаналізувавши таблицю (див. табл. 1.1), можна зробити висновок, що кожен з цих вебдодатків має свої недоліки, беручи до уваги цей аспект ми можемо включити найкращі механізми та можливості у власний додаток. Скомпонувавши всі необхідні функції на виході отримаємо додаток, який зможе задовільнити усі вимоги користувачів.

Таким чином, додаток повинен мати сучасний дизайн, інтуїтивно зрозумілий інтерфейс та легку навігацію. Також, значною перевагою буде наявність інтерактивних форм. На сторінці новин потрібно розміщувати не тільки актуальні новини, а і освітні матеріали для користувачів. Наявність сторінки з відгуками значно підвищать довіру потенційних клієнтів.

1.3 Визначення архітектури вебдодатку

Визначення архітектури вебдодатку є важливим пунктом у написанні кваліфікаційної роботи. Вибір архітектури повпливає на продуктивність, підтримку та безпеку додатку. Також, це повпливає на легкість його розширення і підтримки, ефективність додатку у виконанні своїх функцій напряду залежить від цього вибору.

Серед наявних архітектурних підходів розглянемо

1. Мікросервісна архітектура.
2. Модульна архітектура.
3. Монолітна архітектура.

Модель мікросервісної архітектури полягає в розбитті додатку на невеликі самостійні сервіси, кожен з яких виконує окрему функцію та може працювати незалежно. Кожен сервіс має свою власну кодову базу, базу даних та інтерфейс для спілкування з іншими сервісами, зазвичай через протоколи обміну повідомленнями. Це дозволяє розробникам працювати одночасно над різними сервісами, використовуючи різні технології, що сприяє швидшому розвитку та тестуванню. Ще одною перевагою мікросервісної архітектури є можливість масштабування. Кожне середовище можна масштабувати незалежно, що дозволяє ефективно управляти ресурсами, покращуючи продуктивність додатку. Крім того, збої в одному сервісі не впливають на роботу інших сервісів, це робить систему більш стійкою і гнучкою до змін. Однак ця архітектура має свої недоліки. Вона складна в управлінні, оскільки містить в собі багато різних сервісів, кожен з яких потребує координації і моніторингу. Зважаючи на те, що додаток бухгалтерської компанії для комунікації з клієнтами не буде настільки масштабним, ця архітектура є не доречною.

Розглядаючи модульну архітектуру можна зауважити, що вона розбиває додаток на модулі, які реалізують певну функціональність, розробляються та підтримуються незалежно. На відміну від мікросервісної архітектури, модулі працюють в одному процесі і спільно використовують ресурси, наприклад базу даних

чи файли конфігурації. При масштабній розробці додатку ця архітектура буде полегшувати розробку і тестування, оскільки кожен модуль можна тестувати та розробляти окремо. Основною перевагою модульної архітектури є швидкість реагування на бізнес-зміни і додавання нових функцій, оскільки модулі можна легко додавати, видаляти або змінювати без необхідності переписування всього додатку. Однак для додатку бухгалтерської компанії, який фокусується на комунікації з клієнтами, важлива безперебійна взаємодія між різними компонентами, такими як обробка запитів клієнтів, надання консультацій та фінансових звітів. Модульна архітектура, де кожен компонент реалізований у вигляді окремого модуля, може призвести до складностей в інтеграції та управлінні залежностями між модулями. Це може ускладнити забезпечення плавної і безперервної роботи додатку, що є критично важливим для комунікації з клієнтами.

Монолітна архітектура передбачає розробку додатку як єдиного цілого, в якому всі компоненти тісно пов'язані і працюють разом. Де все від функціоналу до логіки додатку – реалізовано в одному монолітному блоці. Це спрощує процес розробки і розгортання, оскільки всі компоненти розміщені в одному проекті та розгортаються як єдиний додаток. Великою перевагою для невеликих за розмірами додатків є простота реалізації та управління. Цей підхід дозволяє швидко впроваджувати нові функції, бо немає необхідності в координуванні роботи багатьох модулів чи сервісів. Тому доцільним вибором для нашого додатку є саме ця архітектура, вона забезпечить простоту реалізації та управління та прискорить процес розробки.

Для реалізації додатку було обрано фреймворк [Symfony\[12\]](#) , що базується на монолітній архітектурі. Який ефективно справиться з поставленими задачами.

Отже, для додатку бухгалтерської компанії, орієнтованого на комунікацію з клієнтами, монолітна архітектура буде найбільш доцільною. Цей вибір забезпечить просту інтеграцію всіх функцій в єдиному кодовому блоці, що полегшить управління додатком та підвищить продуктивність. Використовуючи цю архітектуру процес розробки не займе надто багато часу та буде більш простим у виконанні.

1.4 Обґрунтування вибору інструментальних засобів.

Для кваліфікаційної роботи було використано такі інструментальні засоби: Symfony[12] та PHP[7] як основний фреймворк та мова програмування для розробки вебдодатків. PHPStorm[11] для зручного написання, відлагодження та управління кодом. JavaScript, HTML[20] та CSS[19] для створення динамічного та привабливого вебінтерфейсу. MySQL[18] в якості бази даних для зберігання та управління даними.

1. Symfony - це фреймворк для розробки вебдодатків, який базується на мові програмування PHP. Використання Symfony спростить процес розробки, так як пропонує готові рішення для багатьох типових задач. PHP є однією з найпопулярніших мов програмування для веброзробки. PHP дозволить створити вебдодаток, та забезпечить гнучкість та широкі можливості для втілення будь-яких ідей та функціональності.

2. PhpStorm є інтегрованим середовищем розробки (IDE), спеціально розробленим для роботи з мовою програмування PHP. Він надає широкий спектр інструментів та функціональності, які полегшують процес розробки, такі як автозаповнення коду, підтримка систем контролю версій та інші. Також, PhpStorm має вбудовані інструменти для роботи зі структурами проектів Symfony, автоматичне виявлення помилок та підтримку для функцій фреймворку, що значно спрощує розробку на Symfony. Крім того, PhpStorm забезпечує інтеграцію з іншими інструментами розробки, такими як системи контролю версій (наприклад, Git), що робить його універсальним інструментом для розробки вебдодатків. Це забезпечить зручність та ефективність управління проектами та посприятиме швидкій розробці додатку.

3. JavaScript використовується для реалізації динамічності на стороні користувача, дозволяючи взаємодіяти з ним та змінювати зовнішній вигляд сторінок без перезавантаження. Використання JavaScript дозволить створювати різноманітні функціональні елементи, такі як валідація форм, обробка подій та багато іншого.

HTML (HyperText Markup Language) є основною мовою розмітки для вебсторінок, яка визначає їхню структуру та контент. Використання HTML дозволить логічно та структуровано організувати вміст вебсторінок. CSS (Cascading Style Sheets) використовується для визначення зовнішнього вигляду вебсторінок, включаючи оформлення тексту, кольори, розташування елементів та багато чого іншого. Використання CSS дозволить створити привабливий та стилізований дизайн для додатку, що вплине на зовнішній вигляд та забезпечить єдність візуального стилю.

4. MySQL відомий своєю високою продуктивністю та швидкодією. Він здатен ефективно опрацьовувати великі обсяги даних та виконувати складні запити, що робить його ідеальним вибором для нашого додатку. Також, MySQL має широкий спектр можливостей, які дозволяють оптимізувати продуктивність сайту та зменшити час відповіді бази даних. MySQL підтримує широкий набір функцій та можливостей, найважливішою з яких є забезпечення безпеки даних. Гнучкість та масштабованість дозволять легко впровадити нові функціональні можливості, якщо це стане необхідно з часом.

Після вибору та обґрунтування інструментальних засобів було сформовано системні вимоги, які визначають мінімальну конфігурацію, при якій система працюватиме стабільно:

1. Вимоги до середовища виконання:

Мінімально необхідна конфігурація, при якій система працюватиме стабільно:

- Операційна система: Windows 7 або вище;
- Процесор: 32- або 64-розрядний з тактовою частотою 1.2 ГГц або вище;
- Оперативна пам'ять: 2 ГБ для 32-розрядних процесорів та 4 ГБ для 64-розрядних процесорів;
- Вільне місце на жорсткому диску: 14 ГБ для 32-розрядних систем або 18 ГБ для 64-розрядних систем;

2. Вимоги до СУБД та доступу до даних:

— У ядрі системи повинна бути представлена реляційна система управління базами даних (СУБД);

3. Вимоги до клієнтського пристрою

— Операційна система з підтримкою браузерів;

— 1 ГБ ОЗУ;

Висновки до першого розділу

У першому розділі було проведено аналіз вимог до програмного забезпечення та предметної області. Визначено завдання та етапи їх виконання, а також вибрано стек технологій для розробки додатку. Проведено детальний порівняльний аналіз аналогічних додатків, що дозволило виявити їхні переваги та недоліки. Це допомогло виділити найкращі аспекти кожного з аналогів, які можуть бути застосовані у власному проєкті бухгалтерської компанії для комунікації з клієнтами.

Також було розглянуто різні архітектурні підходи для розробки вебдодатків, серед яких мікросервісна, модульна та монолітна архітектури. На основі цього аналізу вирішено реалізувати додаток за допомогою монолітної архітектури, яка забезпечить стабільність та ефективність роботи системи. Було обрано фреймворк Symfony та мову програмування PHP для реалізації додатку, що дозволить досягти високої продуктивності та надійності.

Для розробки інтерфейсу користувача використовуватимуться JavaScript, HTML та CSS, що забезпечить створення динамічного та привабливого вебінтерфейсу. База даних MySQL буде використовуватися для зберігання та управління даними, забезпечуючи необхідну продуктивність і масштабованість системи. Вибір таких інструментальних засобів, як PHPStorm, сприятиме зручності розробки та відлагодження коду.

Обраний підхід надає можливість масштабувати додаток та розробляти новий функціонал, якщо це стане необхідно з часом, забезпечуючи стабільність та ефективність роботи системи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ БУХГАЛТЕРСЬКОЇ КОМПАНІЇ ДЛЯ КОМУНІКАЦІЇ З КЛІЄНТАМИ

2.1 Визначення варіантів використання та структури системи

Для забезпечення стабільного виконання всіх необхідних функцій та дотримання встановлених стандартів і специфікацій, сформульовано високорівневі вимоги для веборієнтованої системи бухгалтерської компанії для комунікації з клієнтами. Ці вимоги забезпечать безперебійну роботу системи, відповідність усім необхідним нормативам і специфікаціям, а також гарантують надійність, безпеку та ефективність у процесі взаємодії з клієнтами.

Бізнес вимоги:

- Основною ціллю є забезпечення ефективної та зручної комунікації між консультантом компанії та клієнтами;
- Представлення проекту - проект буде реалізовано у вигляді сайту, що містить усі необхідні функції для цієї задачі;

Вимоги користувачів:

- Надсилати прохання про зв'язок з собою (заповнивши форму з контактними даними);
- Надсилання запитань (заповнивши форму з контактними даними та власне питанням);
- Можливість залишити відгук про компанію;
- Перегляд відгуків про компанію;
- Відстежування статистики;
- Перегляд новин;
- Перегляд коментарів до новин;
- Коментування новин;

Вимоги персоналу вебдодатку:

- Перегляд та видалення запитань;
- Додавання, редагування, видалення новин;
- Перегляд, редагування, видалення та відповідь на коментар;
- Відстежування та аналіз статистики сайту;
- Перегляд, редагування, видалення відгуків про компанію;

Характеристика об'єкта комп'ютеризації:

За допомогою зручного та інтуїтивно зрозумілого інтерфейсу додатку користувач буде мати зручний доступ до необхідної інформації та послуг, що спростить процес взаємодії та покращить рівень обслуговування.

Функціональні вимоги:

- Реєстрація та авторизація користувачів: Можливість реєстрації нових облікових записів та авторизації існуючих користувачів з метою доступу до особистого кабінету;
- Комунікація з консультантами: Функціонал для можливості задати питання консультантам бухгалтерської компанії;
- Перегляд новин: Клієнти повинні мати можливість переглядати актуальні новини, повідомлення та оголошення, які розміщені бухгалтерською компанією;
- Коментування новин: Функція дозволяє клієнтам залишати коментарі під статтями, що дозволяє створити взаємодію між компанією та клієнтами;
- Відгуки: Можливість клієнтів залишати відгуки про надані послуги, які будуть видимі для інших користувачів додатку;

Нефункціональні вимоги:

- **Сприйняття:**

Час, необхідний для освоєння сайту звичайних користувачів – 5 хвилин, а для просунутих користувачів – 2-3 хвилини;

Час відгуку для типових задач – не більше 3 секунд, для складних завдань – не більше 5 секунд;

— **Надійність:**

Доступність – час, що витрачається на обслуговування системи не повинно перевищувати 5% від загального часу роботи;

Середній час безвідмовної роботи – 15 календарних днів;

Норма помилок та дефектів в роботі системи – 3 помилки на десять тисяч рядків коду;

— **Продуктивність:**

Система повинна підтримувати мінімум 50 одночасно працюючих користувачів, пов'язаних із загальною базою даних;

— **Масштабованість:**

Здатність системи ефективно працювати при зростанні обсягу даних та користувачів;

Для наглядної взаємодії з клієнтами була створена діаграма варіантів використання(рис. 2.1), яка ілюструє всі можливі дії, які клієнти можуть виконати за допомогою системи. Ця діаграма допомагає зрозуміти, як система взаємодіє з її користувачами, і служить важливим інструментом для аналізу та вдосконалення функціональності додатку.

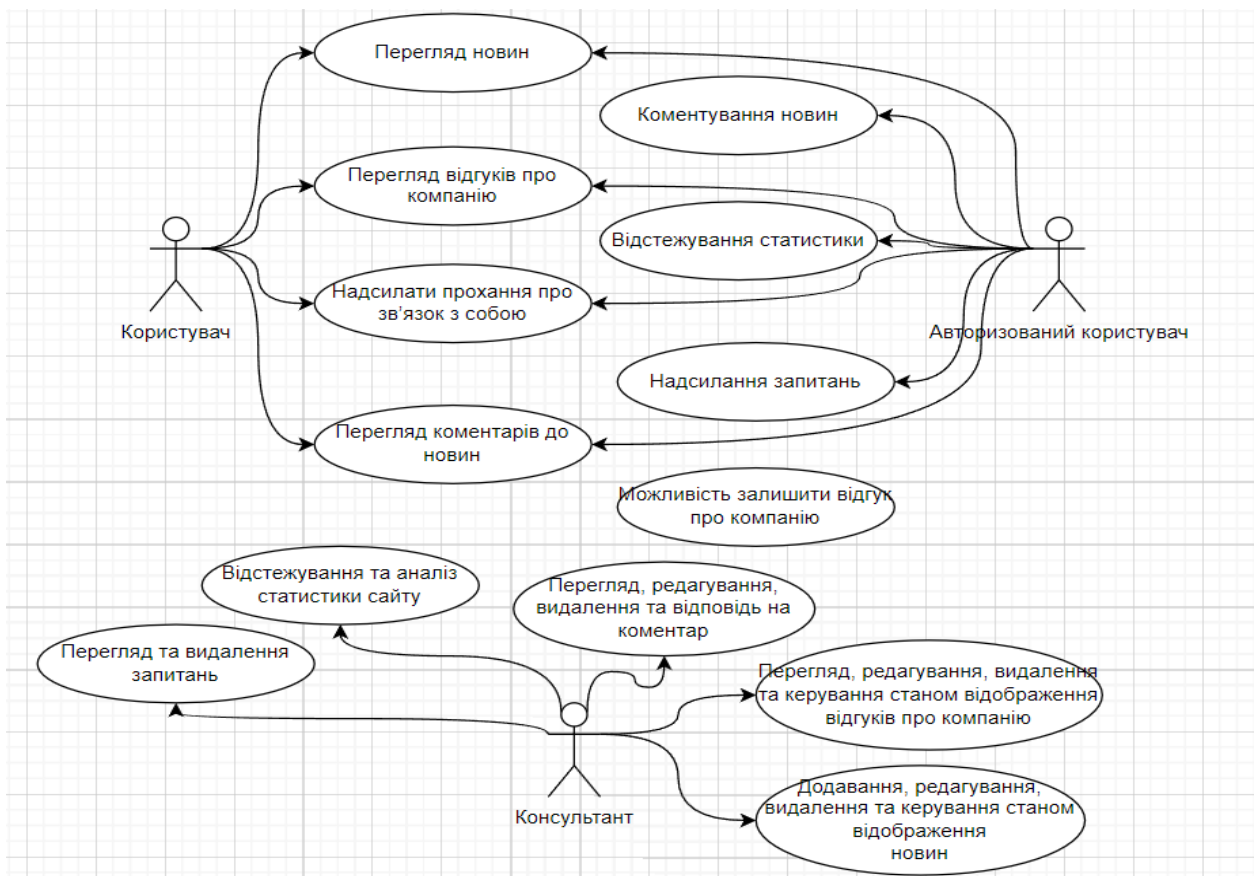


Рисунок 2.1 – Діаграма варіантів використання додатку

Під час аналізу функціональних вимог було виявлено ряд ключових сутностей, що визначають можливі варіанти використання системи. На основі цих функціональних вимог була розроблена діаграма класів сутностей (рис. 2.2), що забезпечать програмну реалізацію додатку.

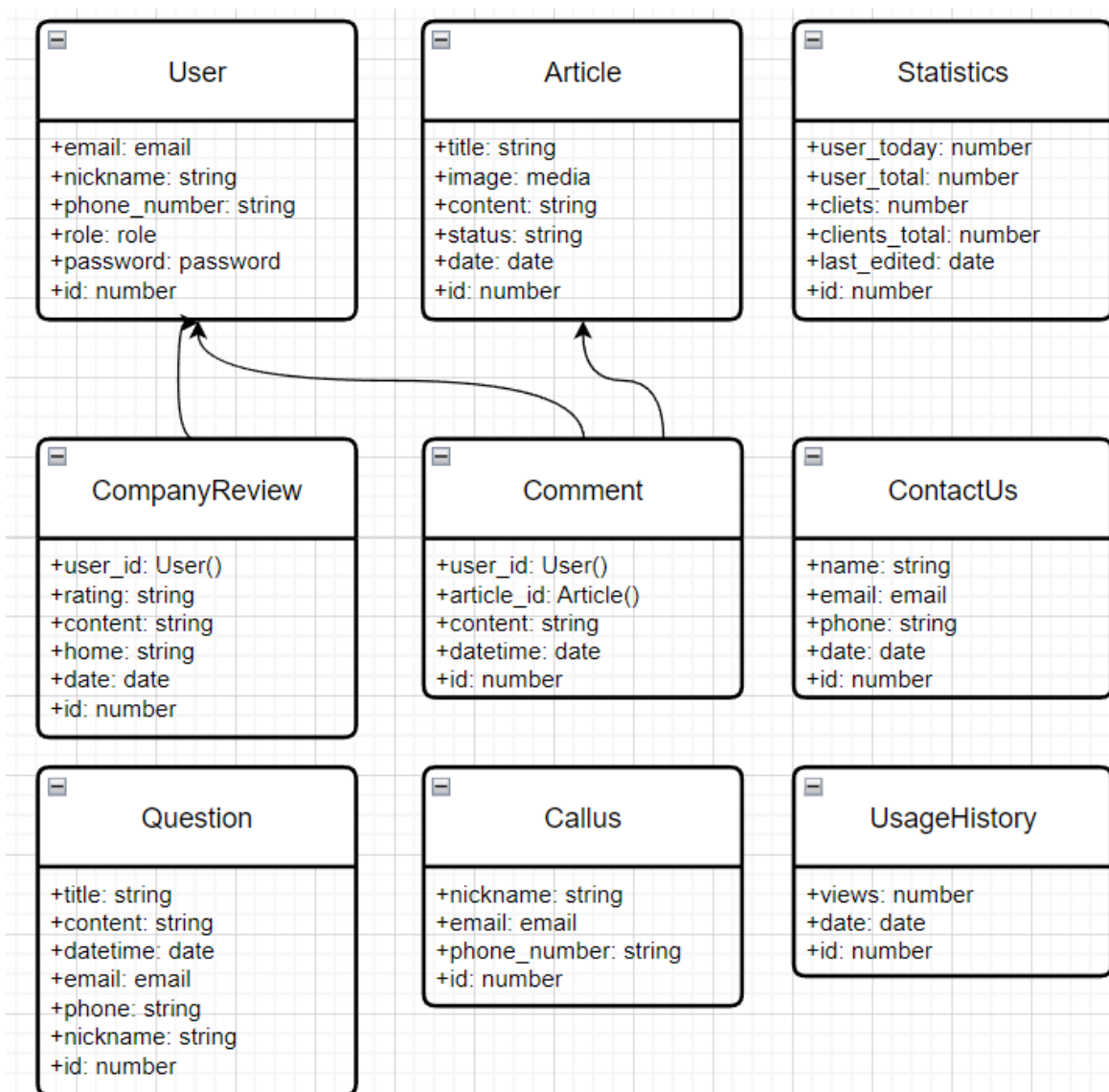


Рисунок 2.2 – Діаграма класів сутностей

Діаграма класів для додатку бухгалтерської компанії включає дев'ять основних сутностей, кожна з яких відіграє важливу роль у функціонуванні системи та взаємодії з користувачами.

Клас “User” зберігає дані користувача, включаючи унікальний ідентифікатор, ім'я користувача, номер телефона, електронну пошту, пароль та роль (наприклад, адміністратор або клієнт). Має зв'язки з більшістю інших сутностей, оскільки користувачі взаємодіють з різними аспектами системи.

Клас “Article” зберігає новини та освітні статті, що публікуються на сайті. Кожна стаття має унікальний ідентифікатор, заголовок, зміст, зображення, статус та

дату публікації. Статті можуть мати коментарі, які зберігаються в “Comment”. Цей клас містить унікальний ідентифікатор коментаря, ідентифікатор статті, до якої він належить, ідентифікатор користувача, що залишив коментар, текст коментаря та дату його створення.

Клас “Statistics” відслідковує дані про використання додатку кожним користувачем. Вона містить унікальний ідентифікатор статистики, ідентифікатор користувача, кількість відгуків та кількість залишених відгуків.

Відгуки про компанію зберігаються в “CompanyReview”, яка містить унікальний ідентифікатор відгуку, ідентифікатор користувача, що залишив відгук, оцінку, текст відгуку та дату його створення.

Запитання користувачів зберігаються в “Question”, який містить унікальний ідентифікатор запитання, ідентифікатор користувача, текст запитання та дату його створення. Ця сутність забезпечує можливість користувачам задавати питання адміністрації компанії. Звернення авторизованих користувачів зберігаються в сутності “CallUs”, “UsageHistory” зберігає дані історії використання додатку користувачами.

2.2 Розробка бази даних

Для реалізації СУБД вебдодатку бухгалтерської компанії було використано MySQL. Її було обрано через високу продуктивність, масштабованість та надійність, що забезпечує швидкий доступ до великих обсягів даних. Крім того, MySQL має потужні інструменти для забезпечення безпеки даних, що є критично важливим для захисту конфіденційної інформації клієнтів.

Діаграма бази даних включає десять таблиць, кожна з яких відповідає за зберігання різних типів даних, що взаємодіють між собою для забезпечення функціональності додатку та зображено на рисунку (рис. 2.3).

1. Users – таблиця, яка містить інформацію про користувачів, включаючи їхні облікові дані та профілі. Поля цієї таблиці включають userID, nickname, password, email, role та phonenumber, необхідні для управління користувачами.

Вона має зв'язки з кількома іншими таблицями для забезпечення повноцінної взаємодії користувачів з додатком.

2. Articles – зберігає дані про статті, опубліковані на вебсайті. Поля цієї таблиці включають articleID, title, content, image, status та Date. Ця таблиця має зв'язок із таблицею Comments, яка містить коментарі користувачів до статей.

3. Statistics – містить дані, що включають відвідуваність, активність користувачів. Поля цієї таблиці включають statisticID, usertoday, usertotal, clients та clientstotal.

4. CompanyReviews – використовується для зберігання відгуків про компанію, залишених користувачами. Поля включають reviewID, userID, rating, content та Date, і вона також має зв'язок з таблицею Users.

5. Comments – зберігає коментарі, залишені користувачами під статтями. Має поля commentID, articleID, userID, replytoid, content та date. Таблиця має зв'язки з таблицями 'Articles' та 'Users', що дозволяє зв'язувати коментарі з конкретними статтями та користувачами.

6. ContactUs – містить інформацію з форми зворотного зв'язку, яку користувачі заповнюють для зв'язку з компанією. Поля цієї таблиці включають contactID, name, email, phone та date.

7. Questions – зберігає запитання, що надходять від користувачів. Має поля questionID, title, content, date, email, phone та nickname.

8. CallUs – містить дані з форми, яку користувачі заповнюють для отримання зворотного дзвінка від компанії. Включаючи в себе поля callID, nickname, email та phonenumber.

9. UsageHistory – зберігає історію використання додатку користувачами. Містить поля historyID, views, та date.

10. Messenger_messages – таблиця для зберігання повідомлень. Має поля messageID, body, headers, queueName, createdat, availableat та deliveredat.

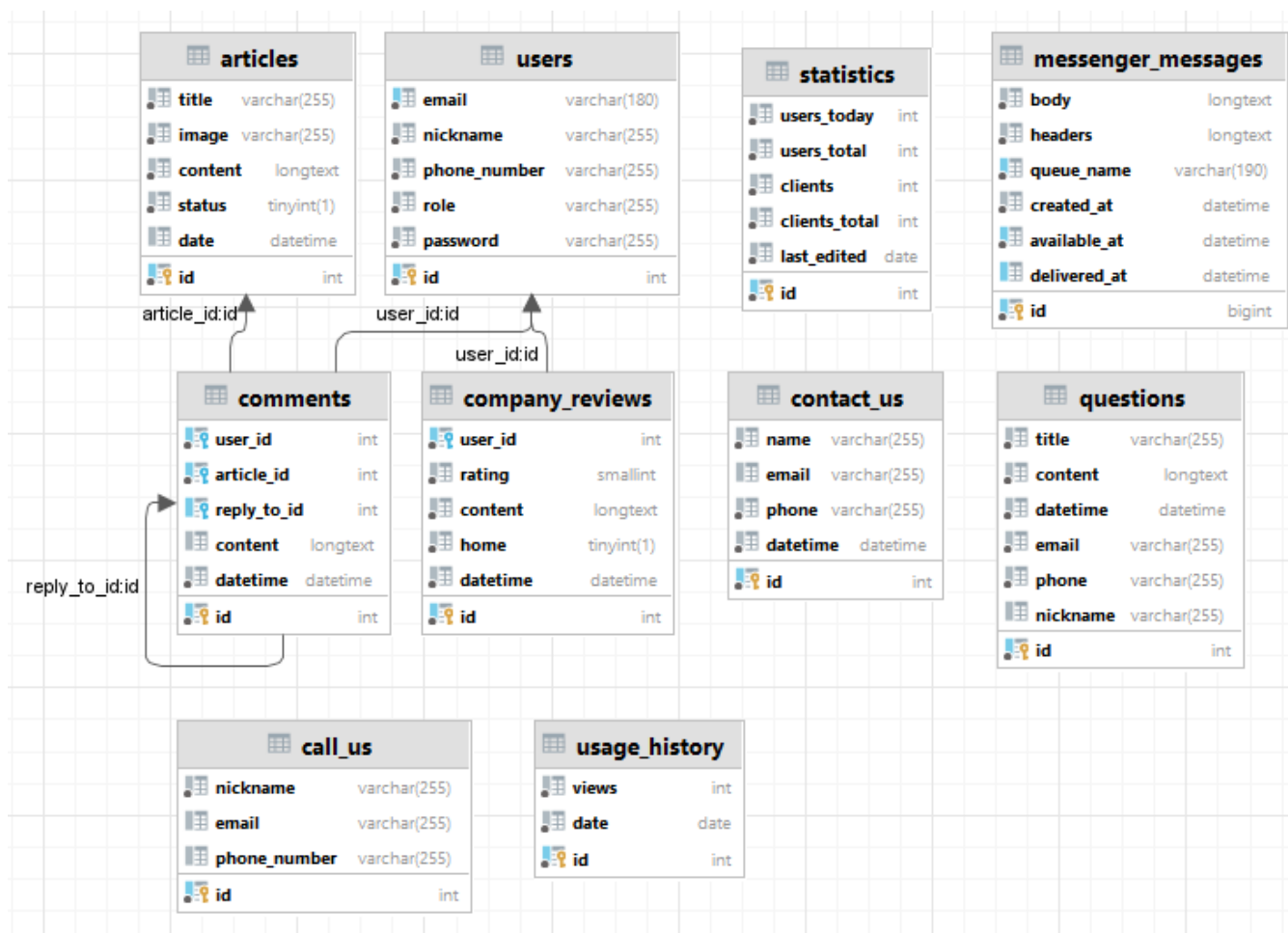


Рисунок 2.3 – Схема БД

Ця схема представляє набір таблиць та їх взаємозв'язки у базі даних. Таблиці включають інформацію про користувачів, статті, відвідуваність користувачів, відгуки про компанію, коментарі, запитання користувачів, інформацію з форм зворотного зв'язку.

2.3 Проектування та реалізація алгоритмів роботи додатку

Проектування та реалізація алгоритмів роботи додатку є комплексним процесом, що вимагає уваги до різних аспектів розробки програмного забезпечення. Цей процес потребує ретельного аналізу вимог до додатку, щоб зрозуміти, які саме завдання мають бути виконані та які функції повинні бути реалізовані. Під час цього етапу було визначено потреби користувачів, технічні обмеження та інші чинники, які можуть повпливати на архітектуру додатку.

Реалізуюючи алгоритм роботи можна зазначити, що додаток бухгалтерської компанії для комунікації з клієнтами спрямований на покращення обміну інформацією та надання консультацій клієнтам. Після реєстрації та авторизації користувач має доступ до основних функцій додатку.

Клієнт може звернутися до бухгалтерської компанії з питаннями або запитаннями через форму зворотного зв'язку. Він може задати питання про бухгалтерський облік, податки, звітність тощо. Після цього запит клієнта надходить до системи обробки запитань.

Спеціалісти бухгалтерської компанії отримують повідомлення про новий запит і розпочинають взаємодію з клієнтом. Вони можуть відповісти на запитання клієнта, надати консультацію або вирішити інші питання.

Такий підхід дозволяє клієнтам отримувати швидку та якісну консультацію від професіоналів у галузі бухгалтерського обліку, а компанії - ефективно вирішувати питання клієнтів та покращувати рівень обслуговування.

Алгоритм роботи з платформою зображено на рисунку (рис. 2.4), а також включає в себе наступні кроки, які детально описані та перелічені у списку.

Не зареєстровані користувачі повинні мати обмежені можливості такі як:

- Має змогу авторизуватись/створити обліковий запис;
- Має змогу переглядати статті та коментарі до них;
- Має змогу переглядати відгуки про компанію;
- Має змогу зв'язатись з консультантом за допомогою форми зворотного зв'язку вказавши свої контактні дані;

Зареєстровані користувачі повинні мати такі можливості такі як:

- Має всі можливості не авторизованого користувача;
- Має змогу коментувати статті та відповідати на вже існуючі коментарі під статтями;
- Має змогу залишити відгук про компанію;
- Має змогу залишити запитання;

Адміністратори повинні мати такі можливості:

- Перегляд та видалення запитань;
- Додавання, редагування, видалення новин;
- Перегляд, редагування, видалення та відповідь на коментар;
- Відстежування та аналіз статистики сайту;
- Перегляд, редагування, видалення відгуків про компанію;

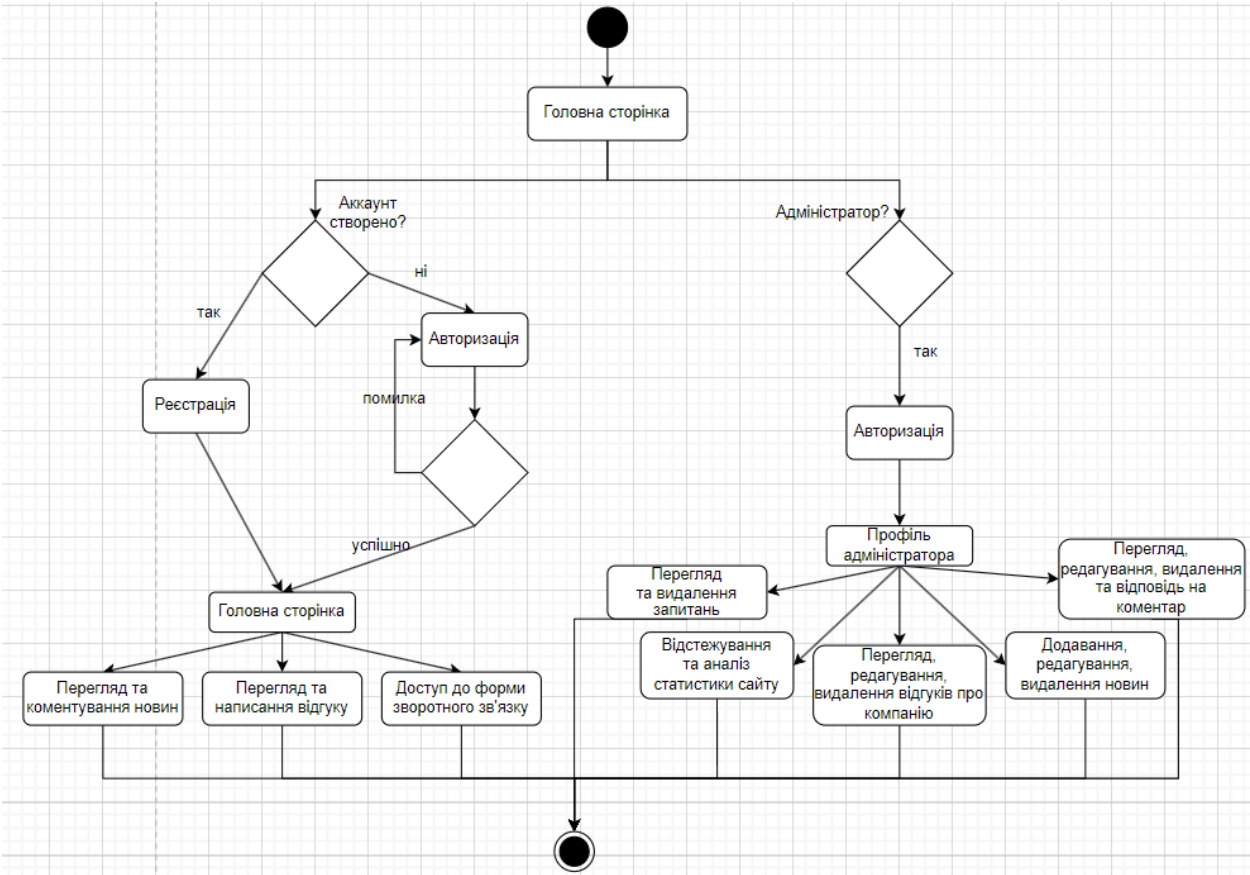


Рисунок 2.4 – Діаграма активності

2.4 Реалізація функціоналу додатку

Для початку реалізуємо модуль реєстрації в додатку, першим кроком буде створення відповідної форми, у якій користувачі зможуть ввести необхідну інформацію для реєстрації. Було додано елементи інтерфейсу, які підкажуть користувачам, які вимоги до пароля (мінімальна довжина).

Реалізація модулю реєстрації в додатку:

```

public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('email', EmailType::class)
        ->add('nickname', TextType::class)
        ->add('phone_number', TextType::class)
        ->add('agreeTerms', CheckboxType::class, [
            'mapped' => false,
            'constraints' => [
                new IsTrue([
                    'message' => 'Ви повинні погодитися з нашими умовами.',
                ]),
            ],
        ])
        ->add('plainPassword', PasswordType::class, [
            'mapped' => false,
            'attr' => ['autocomplete' => 'new-password'],
            'constraints' => [
                new NotBlank([
                    'message' => 'Будь ласка, введіть пароль.',
                ]),
                new Length([
                    'min' => 6,
                    'minMessage' => 'Ваш пароль має містити принаймні {{ limit }}
СИМВОЛІВ.',
                    'max' => 4096,
                ]),
            ],
        ])
    ];
}

public function configureOptions(OptionsResolver $resolver): void
{
    $resolver->setDefaults([
        'data_class' => User::class,
    ]);
}

```

Другим кроком є обробка надісланого запиту та перевірка, чи дані введені користувачем є коректними. Це виконується у наступному блоці коду:

```
if ($form->isSubmitted() && $form->isValid())
```

У цьому блоці виконується перевірка, чи форма була надіслана і чи дані, введені користувачем, є коректними. Якщо ці умови виконуються, виконується наступний код для збереження користувача в базі даних та перенаправлення користувача на сторінку входу:

```

$user->setPassword(
    $userPasswordHasher->hashPassword(
        $user,
        $form->get('plainPassword')->getData()
    )
);

```

```
$entityManager->persist($user);
$entityManager->flush();
```

У цьому коді пароль користувача хешується, використовуючи 'UserPasswordHasher'. Після цього об'єкт користувача зберігається в базі даних за допомогою 'EntityManager', а користувача перенаправляє на сторінку входу.

Наступним етапом буде реалізація форми зворотного зв'язку. Ця форма дозволить користувачам звертатися до адміністраторів з будь-якими питаннями або пропозиціями. Для початку буде створення HTML-форма, яка буде відображати анкету зворотного зв'язку. Вона міститиме поля для введення імені, електронної пошти, номера телефону та повідомлення.

Реалізація форми зворотного зв'язку:

```
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('name', null, [
            'label' => false,
            'attr' => [
                'placeholder' => 'Ім\''я'
            ]
        ])
        ->add('email', null, [
            'label' => false,
            'attr' => [
                'placeholder' => 'Ел. пошта'
            ]
        ])
        ->add('phone', PhoneNumberType::class, [
            'label' => false,
            'attr' => [
                'placeholder' => 'Номер телефону'
            ],
            'format' => PhoneNumberFormat::E164,
            'default_region' => 'UA',
            'constraints' => [
                new PhoneNumber([
                    'message' => 'Це значення не є дійсним номером телефону'
                ])
            ]
        ])
        ->add('add', SubmitType::class, [
            'label' => 'Залишити контакт',
            'attr' => [
                'class' => 'radius30 btn-outline-success',
                'style' => 'width: 100%;'
            ]
        ])
    ];
}
```

```
public function configureOptions(OptionsResolver $resolver): void
```

```
{
    $resolver->setDefaults([
        'data_class' => ContactUs::class,
    ]);
}
```

Після створення HTML-структури форми, наступним етапом буде її обробка та збереження даних з форми у відповідному контролері.

```
class ContactUsController extends AbstractController
{
    public function __construct(private ManagerRegistry $doctrine) {}
    #[Route('/contact', name: 'app_contact_us')]
    public function index(Request $request)
    {
        $contact = new ContactUs(); // Створення нового об'єкту ContactUs

        $form = $this->createForm(ContactUsType::class, $contact); // Створення
        форми для введення даних зворотного зв'язку

        $form ->handleRequest($request); // Обробка надісланого запиту

        if($form->isSubmitted() && $form->isValid())// Перевірка, чи дані форми були
        надіслані та чи вони валідні
        {
            // Отримання поточної дати та часу
            $now = new \DateTime('@' . strtotime('now'));
            $now->setTimezone(new \DateTimeZone('Europe/Kiev'));
            // Встановлення дати та часу в об'єкті ContactUs
            $contact->setDatetime($now);
            // Отримання та форматування номеру телефону
            $phoneNumber = $contact->getPhone();
            $cleanedNumber = "+" . preg_replace("/[^0-9]/", "", $phoneNumber);
            $contact->setPhone($cleanedNumber);
            // збереження даних в базу даних
            $em = $this->doctrine->getManager();
            $em->persist($contact);
            $em->flush();
            // Перенаправлення користувача на головну сторінку після успішного збереження
            return new RedirectResponse($this->generateUrl('Main.Index'));
        }
        // Відображення сторінки з формою зворотного зв'язку
        return $this->render(view: 'contact_us/index.html.twig', parameters: [
            'form'=>$form,
        ]);
    }
}
```

Функція index:

- Відповідає за обробку запитів на шляху `/contact`;
- Спочатку створюється новий об'єкт `ContactUs`, який буде використовуватися для збереження даних, введених у форму зворотного зв'язку;

- Потім створюється форма за допомогою методу `createForm` контролера. Форма використовує клас `ContactUsType` для визначення своєї структури;
- Після цього викликається метод `handleRequest`, який обробляє надісланий запит і пов'язує дані форми з введеними користувачем даними;
- Якщо форма була надіслана і дані є валідними, то:
 - Зберігається поточна дата та час у об'єкті `ContactUs`;
 - Номер телефону очищується від всіх нечислових символів і форматується в міжнародний формат;
 - Об'єкт `ContactUs` зберігається в базі даних;
 - Користувач перенаправляється на головну сторінку додатку;
- Якщо форма не була надіслана або дані не є валідними, то:
 - Відображається сторінка з формою зворотного зв'язку для подальшого заповнення користувачем;

Після того, як користувач відправить своє повідомлення через форму зворотного зв'язку, адміністратор може надати відповідь, використовуючи панель адміністрування. Це передбачає збереження відповіді в базі даних разом з інформацією про відповідне повідомлення, щоб мати можливість відстежувати історію комунікації з користувачем.

Наступним кроком буде реалізація функціоналу відгуків, що дозволить користувачам залишати свої оцінки компанії. Цей модуль включатиме створення форми для відгуків, обробку та збереження даних у базі даних, а також відображення залишених відгуків на відповідній сторінці. Крім того, для забезпечення достовірності відгуків буде реалізовано функціонал видалення та зміни, що дозволить адміністраторам перевіряти відгуки. Це допоможе підтримувати високу якість контенту та забезпечити позитивний досвід для всіх користувачів.

Реалізація системи відгуків:

```
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('rating', TextType::class, [
```

```

        'label' => false,
        'attr' => [
            'class' => 's',
            'step' => 1,
        ],
    ])
    ->add('content', null, [
        'label' => false,
        'attr' => [
            'placeholder' => 'Коментар ...',
            'style' => 'height: 10vh;'
        ]
    ])
    ->add('add', SubmitType::class, [
        'label' => 'Додати відгук',
        'attr' => [
            'class' => 'radius30 btn-outline-success',
            'style' => 'width: 100%;'
        ]
    ])
    ];
}

public function configureOptions(OptionsResolver $resolver): void
{
    $resolver->setDefaults([
        'data_class' => CompanyReview::class,
    ]);
}

```

Після створення HTML-структури, наступним етапом буде її обробка та збереження даних, що буде виконуватись у відповідному контролері.

Реалізація контролера для системи відгуків:

Отримання поточного користувача

```
$user = $this->getUser();
```

Перевірка авторизації користувача:

— Якщо користувач не авторизований, відображається сторінка з повідомленням про необхідність авторизації;

```

if($user == null)
{
    return $this->render(view: 'error.html.twig', parameters: [
        'error'=>"Спочатку потрібно пройти авторизацію.",
    ]);
}

```

Перевірка, чи користувач вже залишав відгук:

— Якщо користувач вже залишав відгук, відображається сторінка з повідомленням про те, що повторний відгук залишити не можна;

```

if ($user->getCompanyReview() != null)
{
    return $this->render(view: 'error.html.twig', parameters: [

```

```

        'error'=>"Ви вже залишали відгук.",
    });
}

```

Створення нового відгуку та форми:

— Якщо користувач авторизований і ще не залишав відгук, створюється новий об'єкт `CompanyReview` та форма для його заповнення;

```

else {
    $review = new CompanyReview();

    $form = $this->createForm(CompanyReviewType::class, $review);

    $form->handleRequest($request);

```

Перевірка валідності форми та збереження відгуку:

— Якщо форма надіслана і є валідною, встановлюються додаткові поля (`user`, `home`, `datetime`) і відгук зберігається у базу даних;

```

if ($form->isSubmitted() && $form->isValid()) {

    $now = new \DateTime('@' . strtotime('now'));
    $now->setTimezone(new \DateTimeZone('Europe/Kiev'));
    $user = $this->getUser();
    $review->setUser($user);
    $review->setHome(false);
    $review->setDatetime($now);
    $em = $this->doctrine->getManager();
    $em->persist($review);
    $em->flush();
    return new RedirectResponse($this->generateUrl('Main.Index'));
}

```

Цей метод контролера обробляє запити від користувачів на залишення відгуків, перевіряє їхню авторизацію, унікальність відгуку та зберігає коректні відгуки у базу даних.

Розглянемо метод `reviewRemove`, який відповідає за видалення відгуків користувачів:

```

public function reviewRemove(ManagerRegistry $doctrine, CompanyReview $review)
{
    $em = $doctrine->getManager();

    $em->remove($review);
    $em->flush();
    $this->addFlash(type: 'success', message: 'Відгук видалено.');
```

Метод `reviewRemove` видаляє відгук користувача з бази даних, додає повідомлення про успішне видалення і перенаправляє адміністратора на сторінку зі

списком відгуків. Цей метод забезпечує простий і ефективний спосіб для адміністратора управляти відгуками, дозволяючи легко видаляти неприйнятні відгуки.

Ще один інструмент для редагування відгуків - метод `reviewEdit` дозволяє адміністраторам редагувати відгуки користувачів. Він обробляє запит, зберігає зміни у базі даних та відображає форму для редагування відгуку.

```
public function reviewEdit(ManagerRegistry $doctrine, CompanyReview $review, Request
$request, CompanyReviewRepository $reviewRepository)
{
    $reviewOne = $reviewRepository->find($review);

    $form = $this->createForm(type: CompanyReviewType::class, data: $reviewOne);

    $form->handleRequest($request);

    if ($form->isSubmitted()) {
        $em = $doctrine->getManager();
        $em->flush();

        $this->addFlash(type: 'success', message: 'Відгук оновлено.');
        return $this->redirect($this->generateUrl(route: 'Admin.reviewsGet'));
    }

    return $this->render(view: 'admin_panel/reviews/edit.html.twig', parameters: [
        'form' => $form->createView()
    ]);
}
```

Розглянемо метод `buildForm`, який відповідає за побудову форми для створення новин. Цей метод додає поля до форми, встановлює атрибути та мітки для кожного з них.

Додавання поля заголовку:

```
$builder
->add('title', TextType::class, [
    'attr'=> array(
        'class'=>'form-control p-2',
        'placeholder'=>'Заголовок новини'
    ),
    'label'=>'Заголовок'
])
```

Додавання поля для зображення:

```
->add('image', FileType::class, array(
    'required'=> false,
    'mapped' => false,
```

```
'label'=>'Зображення'
))
```

Додавання поля для контенту:

```
->add('content', TextareaType::class, [
    'attr'=> array(
        'class'=>'form-control p-2 h-100',
        'placeholder'=>'Контент'
    ),
    'label'=>'Вміст новини'
])
```

Додавання поля статусу:

```
->add('status', CheckboxType::class, [
    'label'=>'Опубліковано',
    'required' => false
])
```

Додавання поля дати публікації:

```
->add('date', DateType::class, [
    'label'=>'Дата публікації',
])
```

buildForm визначає структуру форми для новин, додаючи до неї необхідні поля та встановлюючи атрибути для кожного поля. Це забезпечує зручний інтерфейс для введення та редагування новин на сайті. Кожне поле має свої специфічні налаштування, які допомагають створити інтуїтивно зрозумілу та функціональну форму.

Розглянемо клас ArticleController, який створений для обробки основної сторінки зі списком статей та перегляду окремої статті разом із коментарями. Цей контролер забезпечує функціональність для сортування статей, перегляду детальної інформації про статтю та додавання коментарів.

```
class ArticleController extends AbstractController
{
    #[Route('/', name: 'index', methods: ['GET'])]
    public function index(Request $request, ArticleRepository $articleRepository):
    Response
    {
        $sortBy = $request->query->get('article-sort-by');

        if ($sortBy == 'asc-title') {
            $articles = $articleRepository->findBy([], ['title' => 'asc']);
        } else if ($sortBy == 'desc-title') {
            $articles = $articleRepository->findBy([], ['title' => 'desc']);
        } else if ($sortBy == 'asc-date') {
            $articles = $articleRepository->findBy([], ['date' => 'asc']);
        } else if ($sortBy == 'desc-date') {
            $articles = $articleRepository->findBy([], ['date' => 'desc']);
        } else {
```

```

        $articles = $articleRepository->findAll();
    }

    return $this->render('article/index.html.twig', [
        'articles' => $articles
    ]);
}

#[Route('/{id}', name: 'view')]
public function view(Request $request, ArticleRepository $articleRepository,
    CommentRepository $commentRepository, ManagerRegistry $doctrine, $id): Response
{
    $article = $articleRepository->find($id);
    $comments = $commentRepository->findBy(['article' => $article], ['datetime'
=> 'ASC']);

    $user = $this->getUser();

    $otherArticles = $doctrine->getRepository(Article::class)->findBy([], ["date"
=> "DESC"], 5);

    $now = new \DateTime('@' . strtotime('now'));
    $now->setTimezone(new \DateTimeZone('Europe/Kiev'));

    $comment = new Comment();
    $form = $this->createForm(type: CommentsFormType::class, data: $comment);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $comment->setUser($user);
        $comment->setArticle($article);
        $comment->setDatetime($now);

        $em = $doctrine->getManager();
        $em->persist($comment);
        $em->flush();

        return $this->redirect($this->generateUrl(route: 'article.view',
parameters: ['id' => $id]));
    }

    if ($user) {
        return $this->render(view: 'article/view.html.twig', parameters: [
            'article' => $article,
            'comments' => $comments,
            'otherArticles' => $otherArticles,
            'user' => $user->getNickname(),
            'form' => $form->createView()
//            'form_answer' => $form_answer->createView()
        ]);
    } else {
        return $this->render(view: 'article/view.html.twig', parameters: [
            'article' => $article,
            'comments' => $comments,
            'otherArticles' => $otherArticles
        ]);
    }
}

```

Метод index:

					ИПЗ.КР.Б-121-24-ИЗ	Арк.
						37

1. Отримує параметр сортування з запиту.
2. Залежно від параметра сортує статті за заголовком або датою в порядку зростання або спадання.
3. Відображає сторінку зі списком статей.

Метод view:

1. Знаходить статтю за ідентифікатором.
2. Знаходить коментарі до статті.
3. Отримує поточного користувача.
4. Отримує останні 5 статей.
5. Створює новий об'єкт коментаря і форму для його введення.
6. Обробляє форму, зберігає коментар, якщо форма валідна.
7. Відображає сторінку з деталями статті, коментарями, іншими статтями і, якщо користувач авторизований, з формою для додавання коментаря.

Контролер ArticleController забезпечує основну функціональність для роботи зі статтями та коментарями на сайті. Він дозволяє користувачам переглядати список статей із можливістю сортування, а також детально переглядати статті та додавати до них коментарі. Це забезпечує інтерактивність вебдодатку, дозволяючи користувачам взаємодіяти з контентом і залишати свої відгуки.

Для спрощення публікацій статей та актуалізації інформації в них існує метод `editArticle`. Коли адміністратор хоче змінити зміст статті, він звертається до цього методу, передаючи ідентифікатор статті як параметр. Метод отримує статтю з бази даних за цим ідентифікатором і створює форму для редагування на основі даних знайденої статті. Після обробки запиту, якщо форма була подана і пройшла валідацію, метод перевіряє, чи було додано нове зображення. Якщо так, зображення завантажується в папку `/public/uploads`, і шлях до нього оновлюється в об'єкті статті. Якщо зображення не додано, оновлюються тільки текстові дані статті, такі як заголовок, зміст, дата публікації та статус. Після цього всі зміни зберігаються в базі даних. Використання методу `editArticle` забезпечує актуальність і коректність

					ІПЗ.КР.Б-121-24-ПЗ	Арк.
						38

інформації, оскільки дозволяє адміністраторам вносити необхідні правки до статей, включаючи оновлення тексту та зображень.

```
public function editArticle(ArticleRepository $articleRepository, $id, Request
$request, SluggerInterface $slugger): Response
{
    $article = $articleRepository->find($id);
    $form = $this->createForm(ArticleFormType::class, $article);

    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()) {
        $imagePath = $form->get('image')->getData();
        if ($imagePath) {
            $destination = $this->getParameter('kernel.project_dir') .
'/public/uploads';

            $originalFilename = pathinfo($imagePath->getClientOriginalName(),
PATHINFO_FILENAME);
            $safeFilename = $slugger->slug($originalFilename);
            $newFilename = $safeFilename . '-' . uniqid() . '.' . $imagePath-
>guessExtension();

            $imagePath->move(
                $destination,
                $newFilename
            );

            $article->setImage($newFilename);
        } else {
            $article->setTitle($form->get('title')->getData());
            $article->setContent($form->get('content')->getData());
            $article->setDate($form->get('date')->getData());
            $article->setStatus($form->get('status')->getData());
        }

        $this->em->persist($article);
        $this->em->flush();

        return $this->redirectToRoute('Admin.articles');
    }

    return $this->render('admin_panel/articles/edit.html.twig', [
        'article' => $article,
        'form' => $form->createView()
    ]);
}
```

Висновки до другого розділу

Другий розділ був присвячений детальному аналізу та реалізації різних аспектів системи. Початковим етапом стало визначення варіантів використання та структура системи. У цьому контексті було визначено бізнес-вимоги, вимоги користувачів і

персоналу, а також функціональні та нефункціональні вимоги. Це дозволило встановити чіткі цілі, що є критично важливим для подальшого успішного проектування та реалізації.

Наступним важливим етапом стала розробка бази даних. Було використано MySQL для бази даних, що включає 10 таблиць: Users, Articles, Statistics, CompanyReviews, Comments, ContactUs, Questions, CallUs, UsageHistory та Messenger_messages. Кожна таблиця має свою роль та функції, забезпечуючи зберігання необхідної інформації для підтримки роботи системи. Цей структурований підхід дозволяє ефективно керувати даними і забезпечувати їх цілісність та доступність.

Проектування та реалізація алгоритмів роботи додатку становили наступний етап розробки. Було визначено можливості для різних типів користувачів, включаючи авторизованих користувачів, не авторизованих користувачів та адміністраторів. Для кращого розуміння та візуалізації процесів була створена діаграма активності, яка відображає основні взаємодії між компонентами системи.

Ще одним етапом стала реалізація функціоналу додатку. Було розроблено та впроваджено різні модулі, включаючи реєстрацію, зворотний зв'язок, управління відгуками та інші ключові компоненти. Це дозволило створити повноцінний додаток, який відповідає всім визначеним вимогам і забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ БУХГАЛТЕРСЬКОЇ КОМПАНІЇ ДЛЯ КОМУНІКАЦІЇ З КЛІЄНТАМИ

3.1 Порядок встановлення та налаштування параметрів додатку

Встановлення та налаштування параметрів додатку бухгалтерської компанії для комунікації з клієнтами включає в себе роботу з низкою складових, кожна з яких потребує окремого розгортання та налаштування. Для зручності відображення цього процесу була створена діаграма розгортання (рис. 3.1), що дозволяє відобразити взаємозв'язки між компонентами:

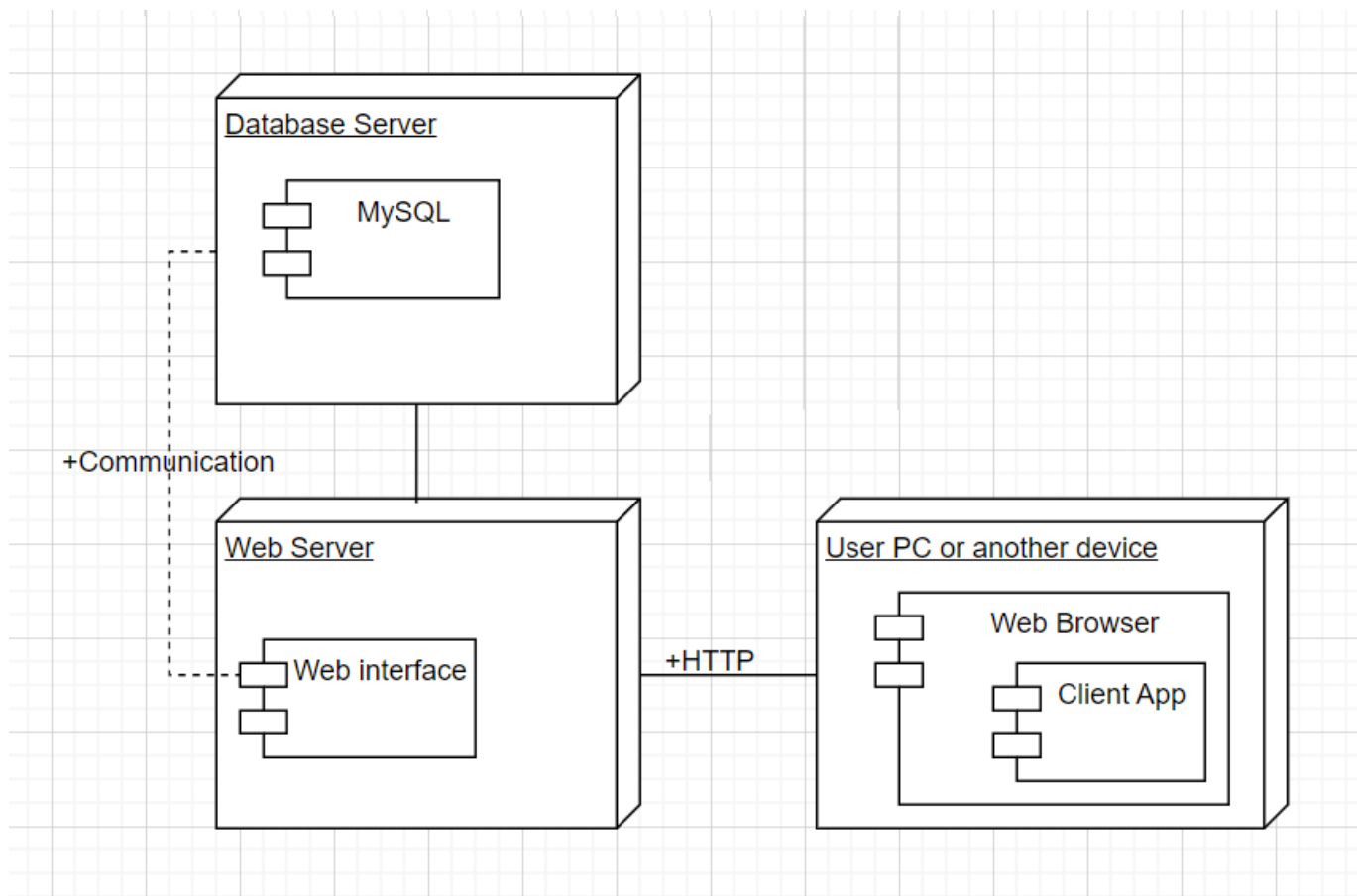


Рисунок 3.1 – Діаграма розгортання

Опис діаграми розгортання:

На даній діаграмі розгортання представлено систему, яка складається з трьох вузлів: вебсерверу, серверу бази даних та клієнтського пристрою. Зв'язок між клієнтським пристроєм та вебсервером відбувається через протокол HTTP, що

реалізує модель запит-відповідь, засновану на стандартах TCP, забезпечуючи передачу даних без втрат. Взаємодія між вебсервером та сервером бази даних здійснюється через протокол TCP/IP, що гарантує стабільний та безперервний обмін інформацією.

3.2 Структура інтерфейсу вебдодатку

Взаємодія користувачів з сайтом починається з відкриття головної сторінки (рис. 3.2). Перше, що вони побачать це банер “Financial Guardians” з кнопкою, за допомогою якої можна звернутись до компанії з будь-яким питанням, що стосується бухгалтерського обліку.

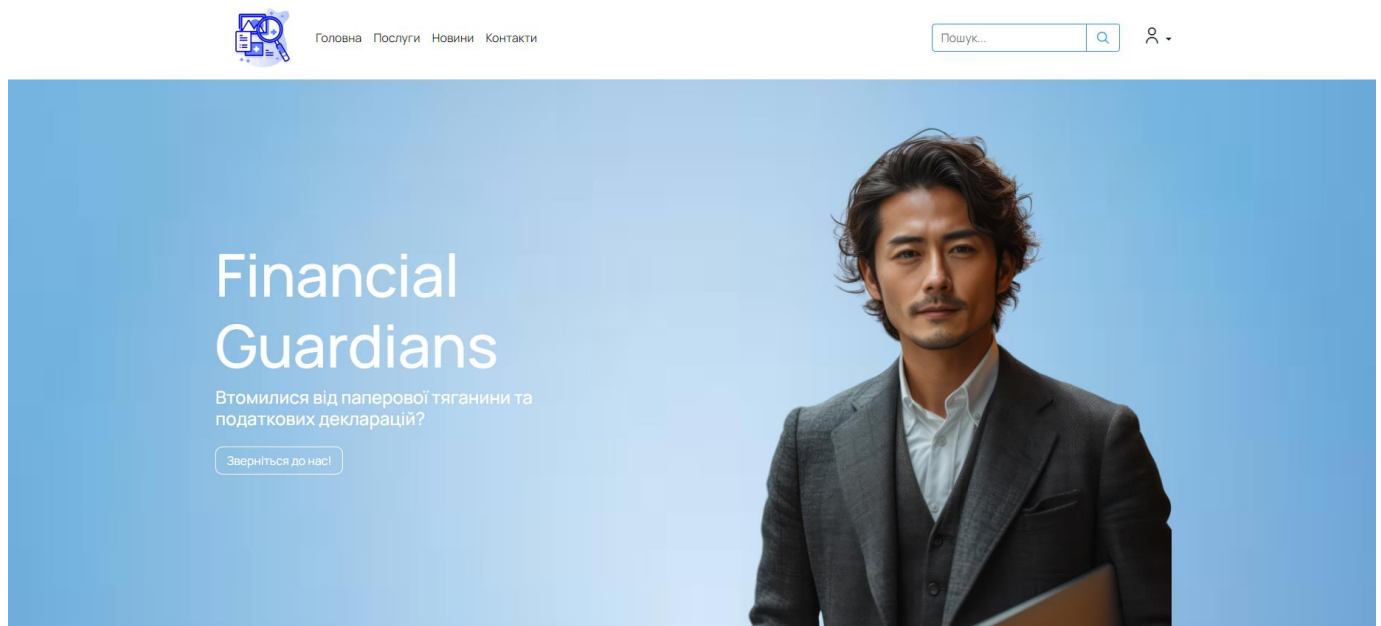


Рисунок 3.2 – Головна сторінка

У шапці додатку можна побачити навігаційне меню, що містить основні розділи сайту: “Головна”, “Послуги”, “Новини” і “Контакти” (рис. 3.3). Справа від меню розташована стрічка пошуку, що допоможе користувачу швидко знайти потрібну інформацію. Також в цій частині міститься кнопка для входу на сайт (рис. 3.4).



Рисунок 3.3 – Навігаційне меню

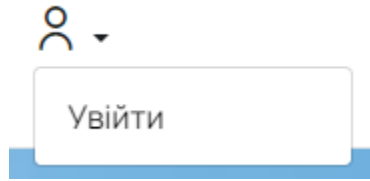


Рисунок 3.4 – Кнопка входу

Натиснувши на кнопку “Увійти”, у користувача відкриється сторінка де йому буде запропоновано ввести свої дані для входу на сайт, при умові, що він уже має зареєстрований обліковий запис (рис. 3.5)

Рисунок 3.5 – Форма для входу в обліковий запис

У випадку необхідності зареєструвати обліковий запис користувач натискає кнопку “Зареєструватися” у низу форми (рис. 3.5). При реєстрації відвідувачу буде слід вказати своє ім'я, пошту, номер телефону та придумати пароль (рис. 3.6).

Реєстрація

☐ Ви приймаєте умови?

[Зареєструватися](#)

Вже є аккаунт? [Увійти](#)

Рисунок 3.6 – Форма для реєстрації облікового запису

Внизу сторінки розміщена кнопка, що відкриває чат-бота, який може надавати різноманітні послуги, такі як відповіді на найпоширеніші запитання, містить корисну інформацію про послуги компанії та правила сайту (рис. 3.7).

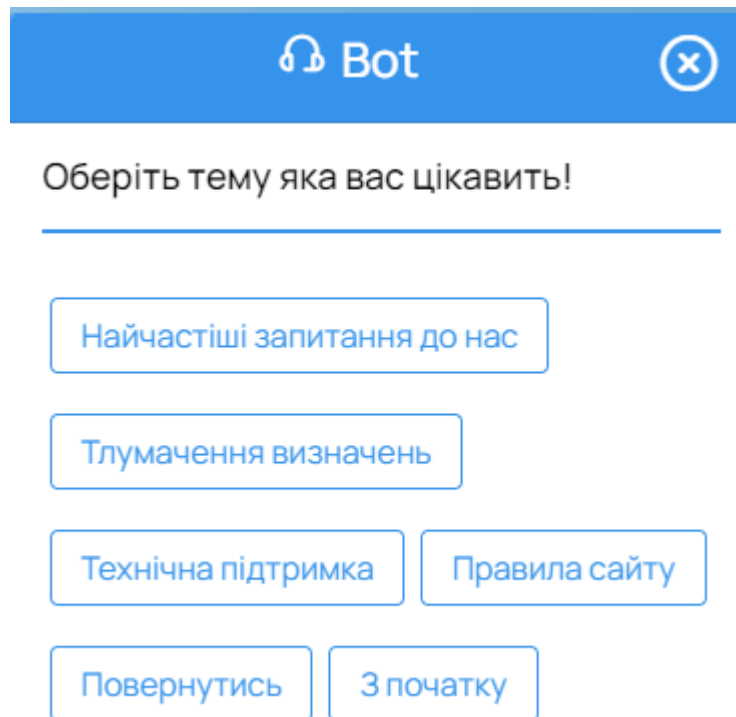


Рисунок 3.7 – Чат-бот

Додаток містить дві ролі: користувач і адміністратор. Користувачі, які відвідують сайт, можуть скористатися різноманітними послугами компанії. Адміністратори, у свою чергу, мають доступ до адміністративної панелі, де вони можуть керувати контентом сайту.

3.2.1 Інтерфейс панелі адміністратора

Для отримання ролі адміністратора потрібно звернутися до вже існуючого адміністратора і запросити цю роль. Адміністратор може надати роль через інтерфейс вебдодатку, змінивши роль у базі даних. При отриманні ролі адміністратора, надається доступ до адміністративної панелі (рис. 3.8). Панель надає можливість керувати різними аспектами вебдодатку, такими як управління користувачами, контентом, налаштуваннями та іншими параметрами, які впливають на функціонування та безпеку системи.

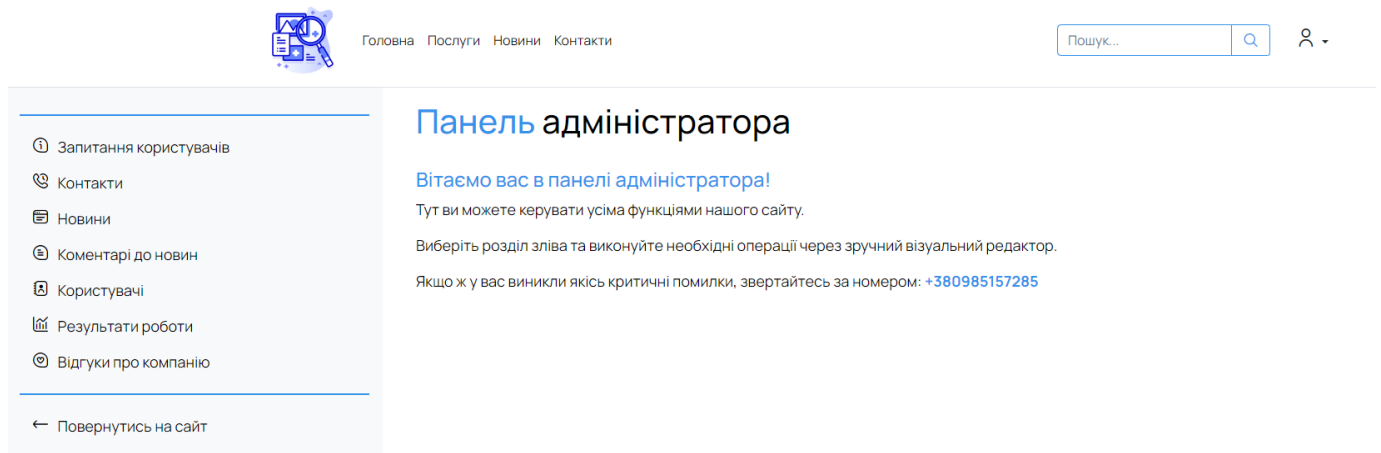


Рисунок 3.8 – Панель адміністратора

Адміністратор має повний доступ до запитань користувачів (рис. 3.9) і може їх видаляти (рис. 3.10). Він може переглядати всі запитання, що надійшли від користувачів, включно з їх вмістом.

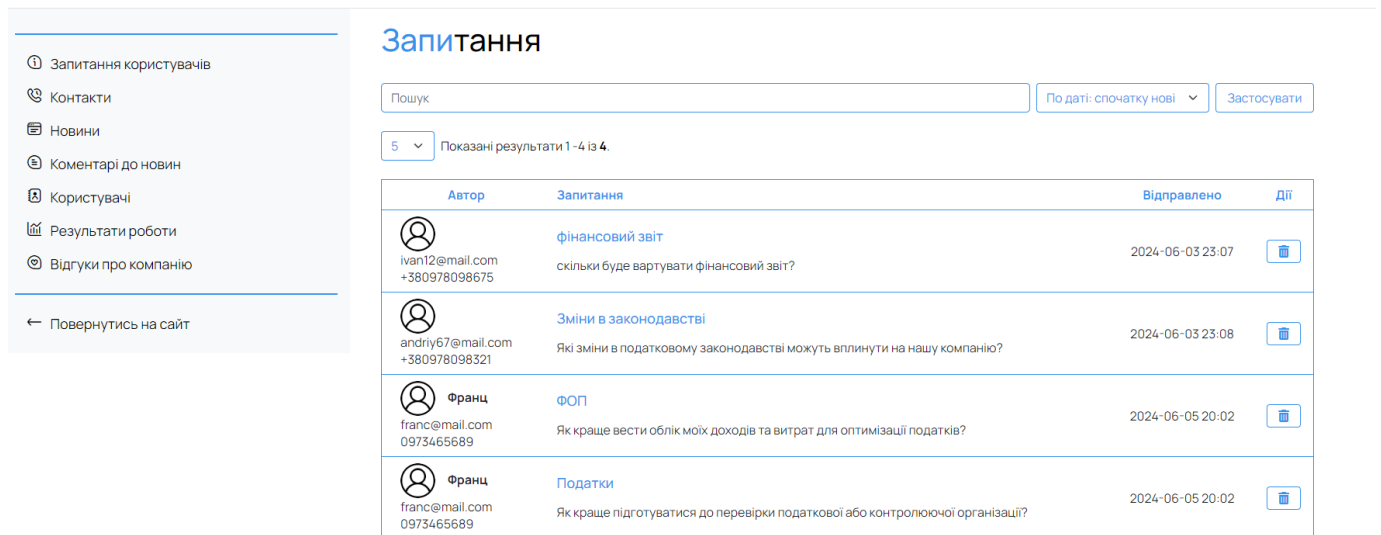


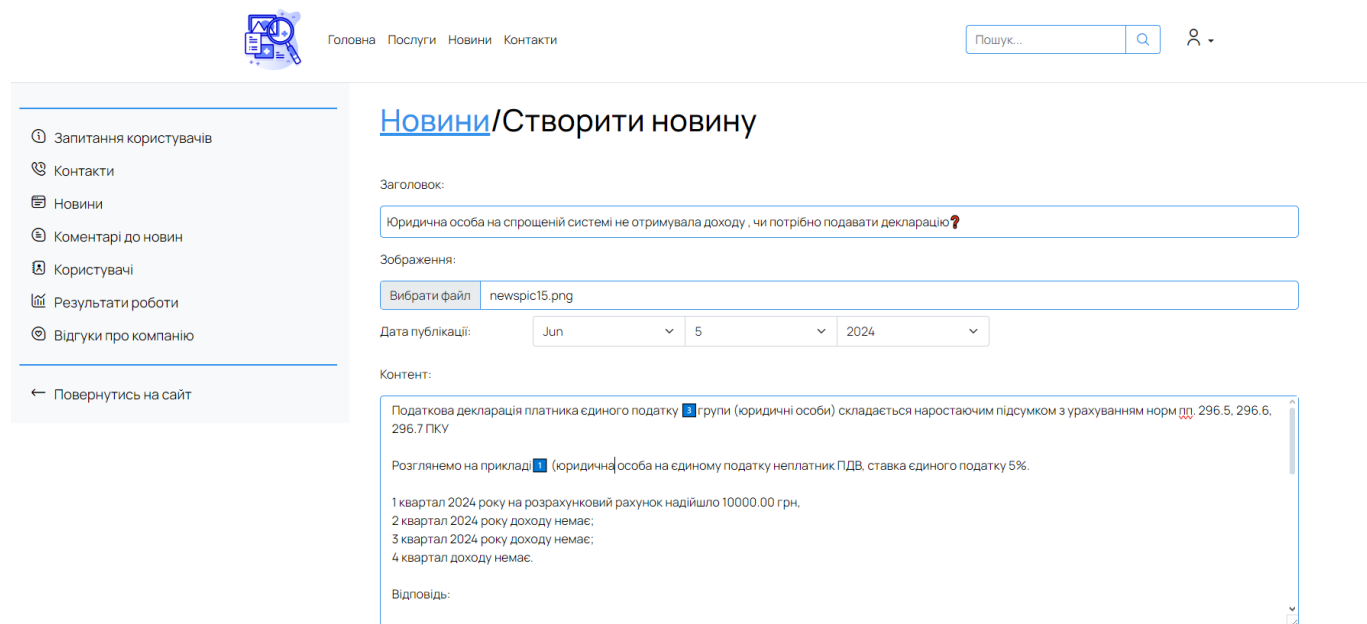
Рисунок 3.9 – Розділ запитань

Запитання видалено.

Рисунок 3.10 – Успішне видалення запитань

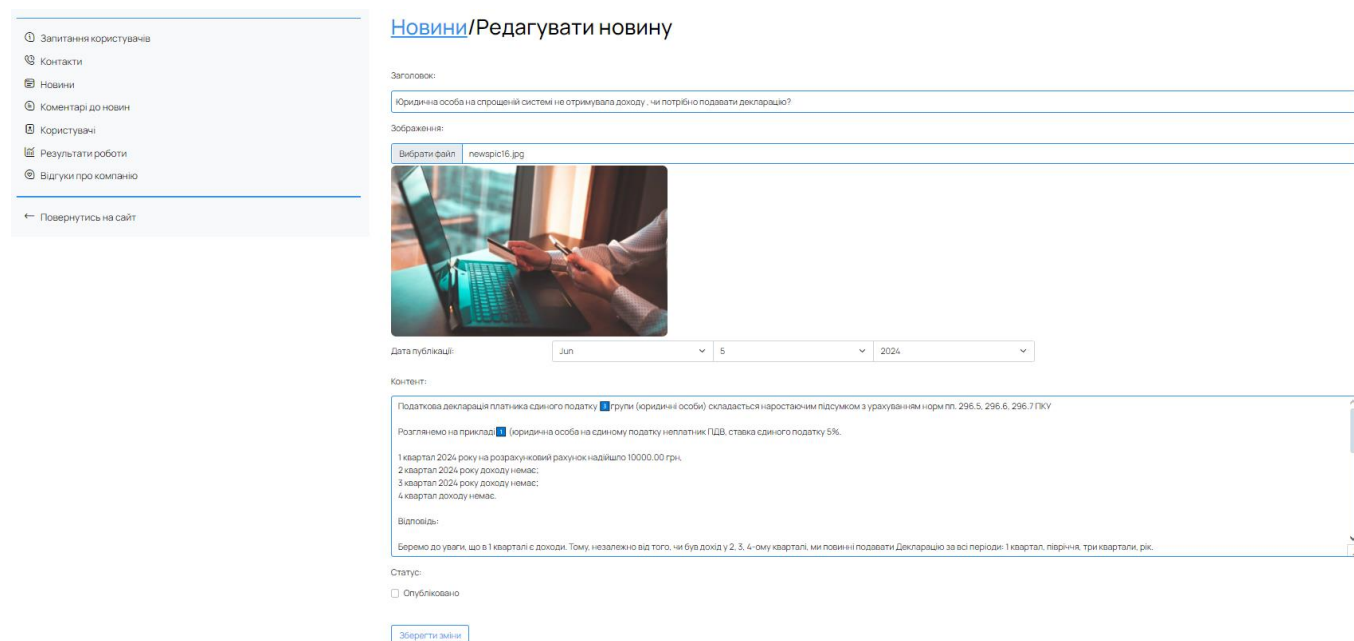
Адміністратор має можливість додавати новини на сайт(див. рис. 3.11), редагувати(рис. 3.12) та видаляти(рис. 3.13) їх за необхідності. Цей функціонал дозволяє зручно керувати контентом сайту та швидко оновлювати інформацію.

Редагування і видалення новин дозволяє адміністратору редагувати або видаляти стару інформацію, що втратила актуальність або потребує корекції.



The screenshot shows the 'Новини/Створити новину' (News/Create News) form. On the left is a sidebar with navigation links: 'Запитання користувачів', 'Контакти', 'Новини', 'Коментарі до новин', 'Користувачі', 'Результати роботи', 'Відгуки про компанію', and a 'Повернутись на сайт' button. The main form area has a title 'Новини/Створити новину'. It includes a 'Заголовок:' (Title) field with the text 'Юридична особа на спрощеній системі не отримувала доходу, чи потрібно подавати декларацію?'. Below is a 'Зображення:' (Image) section with a 'Вибрати файл' (Choose file) button and the filename 'newspic15.png'. A 'Дата публікації:' (Publication date) section shows 'Jun', '5', and '2024'. The 'Контент:' (Content) area contains a text editor with the following text: 'Податкова декларація платника єдиного податку [1] групи (юридичні особи) складається наростаючим підсумком з урахуванням норм пп. 296.5, 296.6, 296.7 ПКУ. Розглянемо на прикладі [1] (юридична) особа на єдиному податку неплатник ПДВ, ставка єдиного податку 5%. 1 квартал 2024 року на розрахунковий рахунок надійшло 10000.00 грн. 2 квартал 2024 року доходу немає; 3 квартал 2024 року доходу немає; 4 квартал доходу немає. Відповідь:'. At the bottom left of the form is a 'Повернутись на сайт' button.

Рисунок 3.11 – Форма для створення статті



The screenshot shows the 'Новини/Редагувати новину' (News/Edit News) form. The sidebar is identical to the previous form. The main form area has a title 'Новини/Редагувати новину'. It includes a 'Заголовок:' (Title) field with the text 'Юридична особа на спрощеній системі не отримувала доходу, чи потрібно подавати декларацію?'. Below is a 'Зображення:' (Image) section with a 'Вибрати файл' (Choose file) button and the filename 'newspic16.jpg'. A 'Дата публікації:' (Publication date) section shows 'Jun', '5', and '2024'. The 'Контент:' (Content) area contains a text editor with the following text: 'Податкова декларація платника єдиного податку [1] групи (юридичні особи) складається наростаючим підсумком з урахуванням норм пп. 296.5, 296.6, 296.7 ПКУ. Розглянемо на прикладі [1] (юридична) особа на єдиному податку неплатник ПДВ, ставка єдиного податку 5%. 1 квартал 2024 року на розрахунковий рахунок надійшло 10000.00 грн. 2 квартал 2024 року доходу немає; 3 квартал 2024 року доходу немає; 4 квартал доходу немає. Відповідь: Беремо до уваги, що в 1 кварталі є доходи. Тому, незалежно від того, чи був дохід у 2, 3, 4-ому кварталі, ми повинні подавати Декларацію за всі періоди: 1 квартал, півріччя, три квартали, рік. Статус: ☐ Опубліковано. Зберегти зміни'. At the bottom left of the form is a 'Повернутись на сайт' button.

Рисунок 3.12 – Форма для редагування статті



Рисунок 3.13 – Кнопка для видалення статті

Також присутній функціонал для перегляду коментарів користувачів (рис. 3.14). Адміністратор може переглядати всі залишені коментарі, вносити зміни (рис. 3.15) або видаляти (рис. 3.16) їх при необхідності.

Автор	Коментар	Відправлено	Дії
 Олена olenamzur@mail.com	Відповідь на коментар: Цікава інформація Цікава та корисна!	2024-06-05 23:18	 
 Олена olenamzur@mail.com	Чудова стаття! Дуже детально розписано всі аспекти.	2024-06-05 23:18	  
 Олена olenamzur@mail.com	Відповідь на коментар: Корисно Погоджуюсь	2024-06-05 23:16	 
 Олена olenamzur@mail.com	Ця інформація мені допомогла!	2024-06-05 23:16	  
 Франц franc@mail.com	Корисно	2024-06-05 23:12	  
 Франц franc@mail.com	Цікава інформація)	2024-06-05 23:12	  
 Франц franc@mail.com	Дякую за поради!	2024-06-05 23:12	  
 Влад vlad344@mail.com	Відповідь на коментар: Супер так	2024-06-04 00:12	 
 Влад vlad344@mail.com	Супер	2024-06-04 00:12	  

Рисунок 3.14 – Список останніх коментарів

Редагування коментарю

Чудова стаття! Дуже детально розписано всі аспекти.

Рисунок 3.15 – Форма для редагування коментарів

Коментар видалено

Рисунок 3.16 – Повідомлення про успішне видалення коментарю

В додатку представлений функціонал для перегляду списку користувачів (рис. 3.17), редагування їхніх даних (рис. 3.18) та видалення облікових записів (рис. 3.19). Це дозволяє адміністратору ефективно керувати користувацькою базою, забезпечуючи актуальність та коректність інформації про кожного користувача. В разі необхідності, адміністратор може змінювати особисті дані користувачів або видаляти облікові записи, що більше не є активними або порушують правила платформи.

Користувачі

Пошук				Застосувати
10	Показані результати 1 - 6 із 6.			
Автор	Пошта	Номер телефону	Дії	
 admin ADMIN	admin@admin.com	0000		
 Vova ADMIN	vovask@mail.com	0972064581		
 Микола	mukola@mail.com	0972064511		
 Влад	vlad344@mail.com	0972064444		
 Франц	franc@mail.com	0973465689		
 Олена	olenamzur@mail.com	0674512093		

Рисунок 3.17 – Список користувачів

Користувачі/Редагувати дані

Нікнейм:

Микола

Електронна пошта користувача:

mukola@mail.com

Номер телефону:

0972064511

Роль:

ROLE_USER

Зберегти зміни

Рисунок 3.18 – Форма для редагування даних користувачів



Рисунок 3.19 – Кнопка для видалення акаунту користувача

Для перегляду всіх відгуків користувачів (рис. 3.20), їх редагування (рис. 3.21) та видалення (рис. 3.22) у адміністратора присутній функціонал. Кожен відгук підлягає схваленню адміністратором перед тим, як потрапити на сайт. Це дозволяє забезпечити відповідність контенту стандартам платформи, видаляючи небажані або недоречні коментарі.

Відгуки

Пошук

По даті: спочатку нові

Всі

Застосувати

5

Показані результати 1 -5 із 10.

Автор	Текст відгуку	Відправлено	Рейтинг	Дії
Віталій	Завжди на зв'язку і готові допомогти!	2024-06-07 23:39	★★★★★	
Дмитро	Чудова компанія! Допомогли вирішити всі питання з податками швидко та професійно. Рекомендую!	2024-06-07 23:28	★★★★★	
Володимир	Допомогли у вирішенні проблем з податками, швидко та якісно!	2024-06-06 15:20	★★★★★	
Сергій	Ваші послуги перевершили всі мої очікування. Дякую за чудову роботу!	2024-06-06 14:17	★★★★★	
Максим	Чудовий сервіс і привітний персонал. Дякую за професійну роботу!	2024-06-06 14:15	★★★★★	

Рисунок 3.20 – Список всіх відгуків

Редагування відгуку

★★★★★

Компанія справді піклується про клієнтів!

Додати відгук

Рисунок 3.21 – Форма для редагування відгуків

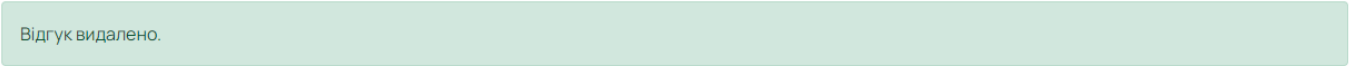


Рисунок 3.22 – Повідомлення про успішне видалення

3.2.2 Інтерфейс клієнта додатка

Так, як інтерфейс з процесом реєстрації та входу в обліковий запис вже було розглянуто, перейдемо до огляду додатку коли користувач вже здійснив вхід до акаунту. На банері головної сторінки представлена кнопка за допомогою якої можна залишити своє питання (рис. 3.23).

З'явилися запитання?

Отримайте відповідь від спеціаліста,
залишивши свої дані

Наші соціальні мережі



Рисунок 3.23 – Форма зворотного зв'язку

Прогорнувши сторінку будуть представлені відгуки інших користувачів (рис. 3.24). Крім того, буде запропоновано оцінити роботу компанії, поділившись своїм досвідом за допомогою кнопки “Залишити відгук” (рис. 3.25).

Що говорять про нас клієнти?

Франц ★★★★★ Допомогли у вирішенні питання 2024-06-05 23:56	Ангеліна ★★★★★ Послуги на найвищому рівні. Рекомендую всім!" 2024-06-06 00:26	Анастасія ★★★★★ Швидко, якісно і надійно. Користуюсь послугами компанії вже кілька років і завжди задоволена. 2024-06-06 00:28
Максим ★★★★★ Чудовий сервіс і привітний персонал. Дякую за професійну роботу! 2024-06-06 14:15	Сергій ★★★★★ Ваші послуги перевершили всі мої очікування. Дякую за чудову роботу! 2024-06-06 14:17	Микола ★★★★☆ Хороша компанія! 2024-06-03 22:57
Олена ★★★★☆ Чудові статті! 2024-06-05 23:59		

Залишити відгук

Рисунок 3.24 – Список існуючих відгуків

Залиште відгук

про нашу компанію



Допомогли у вирішенні проблем з податками, швидко та якісно!

Додати відгук

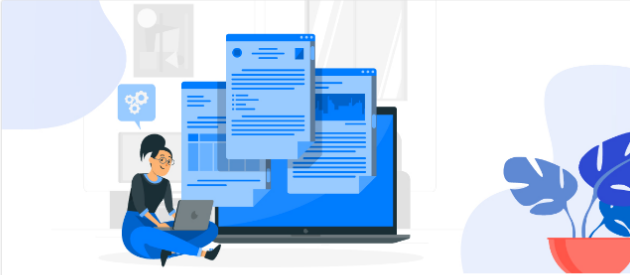
Наші соціальні мережі



Рисунок 3.25 – Форма для написання відгуку

Прогорнувши ще нижче, користувач побачить блок з двома останніми новинами (рис. 3.26). Щоб переглянути всі новини (рис. 3.27)„ треба скористатись кнопкою “Показати ще” або натиснувши на кнопку “Новини” у шапці.

Останні новини



Закон про електронні трудові книжки було опубліковано
Він набрав чинності 10 червня 2021 року. Законом 1217 – IX “Про внесення змін до деяких законодавчих актів України щодо обліку трудової діяльності...”

[Перейти...](#)



Найважливішими ресурсами є час та інформація
Тайм-менеджмент є критично важливим навиком у сучасному світі, де швидкість і ефективність визначають успіх. Ефективне управління часом дозволяє нам з...

[Перейти...](#)

[Показати ще](#)

Рисунок 3.26 – Форма для написання відгуку

[За останніми](#) [Відсортувати](#)



Найважливішими ресурсами є час та інформація
03.06.2024
Тайм-менеджмент є критично важливим навиком у сучасному світі, де швидкість і ефективність визначають успіх. Ефективне управління часом дозволяє нам зосередитися на пріоритетних завданнях, зменшити рівень стресу і підвищ...

[Читати](#)



Як правильно визначити доходи, витрати та прибуток?
02.06.2024
Ви, як директор, часто задаєте бухгалтеру такі питання: “Скільки я заробив? Який мій прибуток? Залишок на рахунку і є моїм прибутком?” Спробую дати відповіді. Надходження на рахунок бувають: ☒ у вигляді грош...

[Читати](#)



Який КВЕД краще обрати для психолога?
30.05.2024
☒ 86.90 – інша діяльність в сфері охорони здоров'я. Цей клас включає в себе діяльність психоаналітиків, психологів і психотерапевтів. ☒ 88.99 – Надання іншої соціальної допомоги без забезпечення проживання. ...

[Читати](#)



Закон про електронні трудові книжки було опубліковано
05.06.2024

Рисунок 3.27 – Форма для написання відгуку

При натисканні кнопки “Читати” відкривається сторінка із повною статтею (рис. 3.28). Також на сторінці розташовані два блоки: перший містить інші новини, які

можуть зацікавити користувача (рис. 3.29), а другий надає можливість залишити коментар під статтею (рис. 3.30).

Найважливішими ресурсами є час та інформація

Опубліковано 03.06.2024



Тайм-менеджмент є критично важливим навиком у сучасному світі, де швидкість і ефективність визначають успіх. Ефективне управління часом дозволяє нам зосередитися на пріоритетних завданнях, зменшити рівень стресу і підвищити продуктивність. Це допомагає досягати цілей швидше і з меншою витратою ресурсів. Вміння організовувати свій час дозволяє знаходити баланс між роботою та особистим життям, що вкрай важливо для загального добробуту і задоволення від життя.

Рисунок 3.28 – Сторінка новини

Рекомендації

Закон про електронні трудові книжки було опубліковано

Він набрав чинності 10 червня 2021 року.
Законом 1217 – IX 'Про внесення змін до деяких законода...

Найважливішими ресурсами є час та інформація

Тайм-менеджмент є критично важливим навиком у сучасному світі, де швидкість і ефективність визначають...

Реєстрація ФОП та ТОВ

Для реєстрації ФОП або ТОВ важливо мати чітке розуміння процесу та вимог, що стосуються створення бі...

Рисунок 3.29 – Форма для написання відгуку

Коментарі

Останні коментарі



Володимир

Введіть текст коментаря ...

Залишити коментар



Франц - 2024-06-05 23:12

Корисно

Відповісти



Олена - 2024-06-05 23:16

Погоджуюсь

Відповісти



Олена - 2024-06-05 23:16

Ця інформація мені допомогла!

Відповісти

Рисунок 3.30 – Коментарі

Також представлений функціонал відповіді на вже існуючий коментар (рис. 3.31).

Автор	Email	Коментар	Відправлено	
	Франц	franc@mail.com	Корисно	2024-06-05 23:12

Введіть текст коментаря ...

Залишити коментар

Рисунок 3.31 – Відповідь на коментар

Користувачу доступна стрічка пошуку, яка дозволяє знаходити потрібну інформацію за ключовими словами (рис. 3.32). Користувач може ввести ключове слово у стрічку пошуку, і отримає результати, які відповідають його запиту.



Результати пошуку



Який КВЕД краще обрати для психолога?

30.05.2024

86.90 – інша діяльність в сфері охорони здоров'я. Цей клас включає в себе діяльність психоаналітиків, психологів і психотерапевтів. 88.99 – Надання іншої соціальної допомоги без забезпечення проживання.

Читати

Рисунок 3.32 – Відповідь на коментар

При виході з облікового запису користувач втрачає можливість залишати відгуки (рис. 3.33) та коментарі (рис. 3.34) на сайті.

Щоб залишити відгук, увійдіть

Рисунок 3.33 – Повідомлення про необхідність входу

Коментарі

Останні коментарі

Для того, щоб залишити коментар, потрібно бути зареєстрованим!

Рисунок 3.34 – Повідомлення про необхідність входу

3.3 Тестування роботи вебдодатку

Тестування вебдодатку є критично важливим етапом розробки, що дозволяє гарантувати його коректну роботу, зручність використання та безпеку. Тестування допомагає виявити та виправити потенційні помилки, забезпечуючи відповідність вимогам та очікуванням користувачів.

Важливо проводити тестування під час розробки додатку, щоб виявляти та виправляти помилки на ранніх етапах, забезпечуючи стабільність роботи. Крім того, після завершення розробки необхідно провести повне тестування всього функціоналу, щоб переконатися, що всі компоненти працюють коректно та відповідають вимогам користувачів. Після виявлення та виправлення помилок додаток піддається повторному тестуванню, що гарантує надійність та відсутність нових проблем.

Для тестування вебдодатку було обрано мануальне тестування, оскільки цей додаток є не дуже масштабним, то така техніка дозволить швидко виявити та виправити більшість проблем, а автоматизація може потребувати більше часу та ресурсів. Мануальне тестування передбачає ручне виконання тестових сценаріїв без використання автоматизованих інструментів. Це дає змогу адаптувати тестування до змін, що виникають під час розробки та ретельно перевірити кожну функцію додатку.

В результаті тестування виявлено недоліки в додатку:

Передумови: Адміністратор бажає додати новину на сайт.

Кроки:

- Адміністратор входить до адмін-панелі;
- Адміністратор натискає на кнопку “Додати новину”;
- Адміністратор вводить заголовок новини у відповідне поле;
- Адміністратор вводить текст новини у редактор;
- Адміністратор додає зображення, вибираючи файл зі свого комп'ютера;
- Адміністратор натискає кнопку “Зберегти”;

Очікуваний результат: Новина успішно додається та відображається у списку новин на сайті.

Фактичний результат: Новина не відображається у списку новин на сайті, хоча адмін-панель показує, що новина була успішно додана.

У процесі розробки функціоналу додатку було створено набір тест-кейсів, призначених для перевірки правильності функціонування його основних можливостей.

1. Реєстрація нового користувача:

Передумови: Користувач бажає створити новий обліковий запис.

Кроки:

- Користувач відкриває модальне вікно з реєстрацією;
- Користувач вводить свою електронну пошту, пароль та інші необхідні дані;

— Користувач натискає кнопку “Реєстрація”;

Очікуваний результат: Користувач успішно реєструється в системі та зберігається у сесії зі своїм унікальним токеном.

Фактичний результат: Користувач успішно реєструється в системі та зберігається у сесії зі своїм унікальним токеном.

2. Відправка повідомлення адміністратору:

Передумови: Користувач бажає відправити повідомлення адміністратору через форму зворотного зв'язку.

Кроки:

— Користувач відкриває форму зворотного зв'язку;

— Користувач вводить текст повідомлення та натискає кнопку “Відправити”;

Очікуваний результат: Повідомлення успішно надсилається адміністратору, користувач отримує підтвердження про успішну відправку.

Фактичний результат: Повідомлення успішно надсилається адміністратору, користувач отримує підтвердження про успішну відправку.

3. Додавання нового відгуку:

Передумови: Користувач бажає залишити відгук про компанію.

Кроки:

— Користувач гортає до розділу з відгуками;

— Користувач натискає кнопку “Залишити відгук”;

— Користувач вводить текст відгуку та натискає кнопку “Додати відгук”;

Очікуваний результат: Відгук успішно додається до системи та відображається після модерації адміністратором.

Фактичний результат: Відгук успішно додається до системи та відображається після модерації адміністратором.

4. Перегляд списку новин:

Передумови: Користувач бажає переглянути останні новини компанії.

					ІПЗ.КР.Б-121-24-ПЗ	Арк.
						58

Кроки:

- Користувач відкриває розділ “Новини” у шапці сайту;
- Користувач переглядає список останніх новин;

Очікуваний результат: Користувач бачить список новин, з можливістю перегляду повних статей.

Фактичний результат: Користувач бачить список новин, з можливістю перегляду повних статей.

5. Додавання коментаря до статті:

Передумови: Користувач бажає залишити коментар до конкретної новини.

Кроки:

- Користувач відкриває розділ “Новини” у шапці сайту;
- Користувач гортає до блоку з коментарями;
- Користувач вводить текст коментаря у відповідне поле та натискає кнопку “Відправити”;

Очікуваний результат: Коментар успішно додається та відображається під статтею.

Фактичний результат: Коментар успішно додається та відображається під статтею.

Висновки до третього розділу

У третьому розділі було детально розглянуто різні аспекти розробки вебдодатку для бухгалтерської компанії. Була представлена діаграма розгортання, яка наочно демонструє архітектуру системи.

Далі було описано інтерфейс користувача, включаючи елементи навігації, чат-бота, а також ролі користувачів та адміністраторів. Було детально розглянуто, як користувачі можуть переглядати та залишати відгуки, коментарі, читати новини та взаємодіяти з іншими елементами сайту. Окрему увагу було приділено панелі

адміністратора, де описано функціонал для управління новинами, коментарями, відгуками та користувачами.

Після цього було проведено тестування додатку, що дозволило виявити та виправити можливі помилки, забезпечуючи надійну роботу системи. Були представлені приклади тест-кейсів для перевірки основних функцій системи, таких як авторизація, редагування даних користувачів та додавання новин адміністратором.

					ІПЗ.КР.Б-121-24-ПЗ	Арк.
						60

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебсайт бухгалтерської компанії для комунікації з клієнтами. Проведено детальний аналіз вимог до програмного забезпечення та предметної області. Визначено завдання та етапи їх виконання, а також проведено порівняльний аналіз аналогічних додатків. На основі проведеного аналізу було обрано архітектуру, що забезпечить стабільність та ефективність системи.

В процесі детального аналізу та реалізації різних аспектів системи, було визначено варіанти використання та структуру системи, що включало вимоги користувачів і персоналу, а також функціональні та нефункціональні вимоги. Для кращого розуміння та візуалізації процесів була створена діаграма, яка відображає основні взаємодії між компонентами системи. Це дозволило визначити можливості для різних типів користувачів, включаючи авторизованих користувачів, не авторизованих користувачів та адміністраторів.

Було детально розглянуто різні аспекти розробки вебдодатку, описано інтерфейс користувача, включаючи елементи навігації, а також ролі користувачів та адміністраторів.

Загалом, під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано систему бухгалтерської компанії для комунікації з клієнтами, яка відповідає всім вимогам додатку. Система забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів, стабільну та ефективну роботу, а також можливість подальшого масштабування та розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Platinumfinance [Електронний ресурс] - Режим доступу до ресурсу: <https://platinumfinance.com.ua>
2. Paragraf [Електронний ресурс] - Режим доступу до ресурсу: <https://paragraf.com.ua>
3. Oblikuzh [Електронний ресурс] - Режим доступу до ресурсу: <https://oblikuzh.com.ua>
4. Bootstrap [Електронний ресурс] - Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
5. Bootstrap [Електронний ресурс] - Режим доступу до ресурсу: <https://getbootstrap.com/docs/4.1/content/reboot/>
6. Bootstrap [Електронний ресурс] - Режим доступу до ресурсу: <https://getbootstrap.com/docs/4.1/utilities/borders/>
7. PHP [Електронний ресурс] - Режим доступу до ресурсу: <https://www.php.net/docs.php>
8. PHP / Constants [Електронний ресурс] - Режим доступу до ресурсу: <https://www.php.net/manual/en/language.constants.php>
9. PHP / Classes [Електронний ресурс] - Режим доступу до ресурсу: <https://www.php.net/manual/en/language.oop5.php>
10. PHP / Functions [Електронний ресурс] - Режим доступу до ресурсу: <https://www.php.net/manual/en/language.functions.php>
11. PhpStorm [Електронний ресурс] - Режим доступу до ресурсу: <https://www.jetbrains.com/help/phpstorm/configuring-php-development-environment.html>
12. Symfony / Setup [Електронний ресурс] - Режим доступу до ресурсу: <https://symfony.com/doc/current/setup.html>

13. Symfony / Forms [Электронный ресурс] - Режим доступа до ресурсу: <https://symfony.com/doc/current/forms.html>
14. Symfony / Validation [Электронный ресурс] - Режим доступа до ресурсу: <https://symfony.com/doc/current/validation.html>
15. Symfony / Controller [Электронный ресурс] - Режим доступа до ресурсу: <https://symfony.com/doc/current/controller.html>
16. Symfony / Password [Электронный ресурс] - Режим доступа до ресурсу: <https://symfony.com/doc/current/security/passwords.html/>
17. Symfony / Template [Электронный ресурс] - Режим доступа до ресурсу: <https://symfony.com/doc/current/templates.html>
18. MySQL [Электронный ресурс] - Режим доступа до ресурсу: <https://dev.mysql.com/doc/>
19. CSS [Электронный ресурс] - Режим доступа до ресурсу: <https://devdocs.io/css/>
20. HTML [Электронный ресурс] - Режим доступа до ресурсу: <https://devdocs.io/html/>
21. TWIG [Электронный ресурс] - Режим доступа до ресурсу: <https://twig.symfony.com/doc/3.x/>
22. Nixon R. Learning PHP, MySQL & JavaScript. Boston, USA, 2021. 709 p.
23. Majida L. A comparative study of laravel and symfony PHP frameworks. Tetouan, Morocco, 2019. 9 p.
24. Sotnik S. Manakov V. Lyashenko V. Overview: PHP and MySQL Features for Creating Modern Web Projects. Washington, USA, 2023. 17p.
25. M'hamed Rahmouni. Model-driven architecture: generating models from Symfony framework. Rabat, Morocco, 2023. 10 p.
26. Gunnard Engebret. Introduction to Symfony. Madison, USA, 2022. 551 p.

ДОДАТКИ

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						64

```

#[Route('/admin', name: 'Admin.')]
class AdminPanelController extends AbstractController
{
    private $em;

    //private $chartBuilder;
    public function __construct(EntityManagerInterface $em,)
    {
        $this->em = $em;
    }

    #[Route('/', name: 'Index')]
    public function index(): Response
    {
        return $this->render('admin_panel/index.html.twig', [
            'controller_name' => 'AdminPanelController',
        ]);
    }

    /** STATISTICS */

    #[Route('/statistics', name: 'StatisticsGet', methods: ["GET"])]
    public function statisticsGet(ManagerRegistry $doctrine): Response
    {
        $stats = $doctrine->getRepository(Statistics::class)->getInstance();
        $statsForm = $this->createForm(StatisticsFormType::class);

        $comments = count($doctrine->getManager()->getRepository(Comment::class)-
>findAll());
        $reviews = count($doctrine->getManager()-
>getRepository(CompanyReview::class)->findAll());

        return $this->statisticsRender($statsForm, $stats, $comments, $reviews);
    }

    #[Route('/statistics', name: 'StatisticsPost', methods: ["POST"])]
    public function statisticsPost(ManagerRegistry $doctrine, Request $request):
Response
    {
        $form = $this->createForm(StatisticsFormType::class);
        $form->handleRequest($request);

        $rep = $doctrine->getManager()->getRepository(Statistics::class);
        $stats = $rep->getInstance();

        if ($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $rep->update($data['clients'], $data['clientsTotal']);
            //...
        }

        $comments = count($doctrine->getManager()->getRepository(Comment::class)-
>findAll());

```

```

        $reviews = count($doctrine->getManager()-
>getRepository(CompanyReview::class)->findAll());

        return $this->statisticsRender($form, $stats, $comments, $reviews);
    }

    private function statisticsRender(FormInterface $form, Statistics $stats, int
$commentsCount, int $reviewsCount): Response
    {
        $chartData = $this->em->getRepository(UsageHistory::class)->last7Days();

        $labels = [];
        $views = [];
        foreach ($chartData as $item) {
            $labels[] = $item->getDate()->format("Y-m-d");
            $views[] = $item->getViews();
        }

        return $this->render('admin_panel/statistics/statistics.html.twig', [
            'form' => $form->createView(),
            'statistics' => $stats,
            'comments' => $commentsCount,
            'reviews' => $reviewsCount,
            'chart' => [
                'labels' => $labels,
                'data' => [
                    'views' => $views
                ]
            ]
        ]);
    }

    /** COMMENTS */

    #[Route('/comments', name: 'commentsGet', methods: ['GET'])]
    public function commentGet(CommentRepository $commentRepository, Request
$request, EntityManagerInterface $em): Response
    {
        $search = $request->query->get('search');

        $sortBy = $request->query->get('comment-sort', 'DESC');
        $page = $request->query->get('page', 1);
        $limit = $request->query->get('limit', 5);
        $offset = ($page - 1) * $limit;
        $comments = $commentRepository->search($search, $sortBy, $limit, $offset,
$totalItems);
        $Pages = ceil($totalItems / $limit);
        return $this->render('admin_panel/comments/index.html.twig', [
            'route' => '',
            'comments' => $comments,
            'search' => $search,
            'sortBy' => $sortBy,
            'Pages' => $Pages,
            'totalItems' => $totalItems,
            'viewedItems' => count($comments),
            'page' => $page,
            'limit' => $limit,

```

```

    });
}

#[Route('/comments/{id}', name: 'comments.id')]
public function commentById(CommentRepository $commentRepository, Request
$request, EntityManagerInterface $em, $id)
{
    $search = $request->query->get('search');
    $sortBy = $request->query->get('comment-sort', 'DESC');
    $page = $request->query->get('page', 1);
    $limit = $request->query->get('limit', 5);
    $offset = ($page - 1) * $limit;
    $comments = [$commentRepository->find($id)];
    $totalItems = 1;
    $Pages = ceil($totalItems / $limit);

    return $this->render('admin_panel/comments/index.html.twig', [
        'route' => 'id',
        'comments' => $comments,
        'search' => $search,
        'sortBy' => $sortBy,
        'Pages' => $Pages,
        'totalItems' => $totalItems,
        'viewedItems' => count($comments),
        'page' => $page,
        'limit' => $limit,
    ]);
}

#[Route('/comments/answer/{id}', name: 'comments.answerTo')]
public function answerTo(ManagerRegistry $doctrine, Request $request,
CommentRepository $commentRepository, $id): Response
{
    $comment = $commentRepository->find($id);
    $user = $this->getUser();

    $answerTo = new Comment();

    $now = new \DateTime('@' . strtotime('now'));
    $now->setTimezone(new \DateTimeZone('Europe/Kiev'));

    $form = $this->createForm(type: CommentsFormType::class, data: $answerTo);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $answerTo->setUser($user);
        $answerTo->setReplyTo($comment);
        $answerTo->setArticle($comment->getArticle());
        $answerTo->setDatetime($now);

        $em = $doctrine->getManager();
        $em->persist($answerTo);
        $em->flush();

        $this->addFlash(type: 'success', message: 'Прокоментовано');

        return $this->redirect($this->generateUrl(route: 'Admin.commentsGet'));
    }
}

```

```

        return $this->render(view: 'admin_panel/comments/answerTo.html.twig',
parameters: [
    'form' => $form->createView(),
    'comment' => $comment
]);
    }

    #[Route('/comments/delete/{id}', name: 'comments.delete')]
    public function remove(ManagerRegistry $doctrine, CommentRepository
$commentRepository, $id)
    {
        $comment = $commentRepository->find($id);
        $this->deleteCommentAndReplies($comment, $doctrine);

        $this->addFlash(type: 'success', message: 'Коментар видалено');

        return $this->redirect($this->generateUrl(route: 'Admin.commentsGet'));
    }

    #[Route('/comments/edit/{id}', name: 'comments.edit')]
    public function edit(ManagerRegistry $doctrine, Request $request,
CommentRepository $commentRepository, $id)
    {
        $commentOne = $commentRepository->find($id);

        $form = $this->createForm(type: CommentsFormType::class, data: $commentOne);
        $form->handleRequest($request);

        if ($form->isSubmitted()) {
            $em = $doctrine->getManager();
            $em->flush();

            $this->addFlash(type: 'success', message: 'Коментар оновлено');

            return $this->redirect($this->generateUrl(route: 'Admin.commentsGet'));
        }

        return $this->render(view: 'admin_panel/comments/edit.html.twig', parameters:
[
    'form' => $form->createView()
]);
    }

    private function deleteCommentAndReplies(Comment $comment, ManagerRegistry
$doctrine)
    {
        $entityManager = $doctrine->getManager();

        foreach ($comment->getReplies() as $reply) {
            $this->deleteCommentAndReplies($reply, $doctrine);
        }

        $entityManager->remove($comment);
        $entityManager->flush();
    }

    /** REVIEWS */

```

```

#[Route('/reviews', name: 'reviewsGet', methods: ['GET'])]
public function reviewGet(CompanyReviewRepository $companyReviewRepository,
Request $request, EntityManagerInterface $em): Response
{
    $searchByNickname = $request->query->get('searchByName');
    $sortBy = $request->query->get('review-sort');
    $search = $request->query->get('searchByName');
    $sortBy = $request->query->get('review-sort', 'DESC');
    $reviewHome = $request->query->get('review-home');
    $page = $request->query->get('page', 1); // Get the current page, default to
1
    $limit = $request->query->get('limit', 5); // Get the limit, default to 10
    $offset = ($page - 1) * $limit;
    $reviewHome = ($reviewHome !== "1" && $reviewHome !== "0") ? null :
$reviewHome;
    $reviews = $companyReviewRepository->search($search, $reviewHome, $sortBy,
$limit, $offset, $totalItems);

    $Pages = ceil($totalItems / $limit); // Calculate the total number of pages
    return $this->render('admin_panel/reviews/index.html.twig', [
        'reviews' => $reviews,
        'searchByName' => $search,
        'sortBy' => $sortBy,
        'reviewHome' => $reviewHome,
        'Pages' => $Pages,
        'totalItems' => $totalItems,
        'viewedItems' => count($reviews),
        'page' => $page,
        'limit' => $limit,
    ]);
}

#[Route('/reviews/delete/{id}', name: 'reviews.delete')]
public function reviewRemove(ManagerRegistry $doctrine, CompanyReview $review)
{
    $em = $doctrine->getManager();

    $em->remove($review);
    $em->flush();
    $this->addFlash(type: 'success', message: 'Відгук видалено.');
```

```

    return $this->redirect($this->generateUrl(route: 'Admin.reviewsGet'));
}

#[Route('/reviews/main/{id}', name: 'reviews.main')]
public function reviewMain(ManagerRegistry $doctrine, CompanyReview $review)
{
    $review->setHome(!$review->getHome());
    $em = $doctrine->getManager();
    $em->persist($review);
    $em->flush();
    $this->addFlash(type: 'success', message: 'Відгук змінено.');
```

```

    return $this->redirect($this->generateUrl(route: 'Admin.reviewsGet'));
}

#[Route('/reviews/edit/{id}', name: 'reviews.edit')]
public function reviewEdit(ManagerRegistry $doctrine, CompanyReview $review,
Request $request, CompanyReviewRepository $reviewRepository)
{

```

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						69

```

        $reviewOne = $reviewRepository->find($review);

        $form = $this->createForm(type: CompanyReviewType::class, data: $reviewOne);

        $form->handleRequest($request);

        if ($form->isSubmitted()) {
            $em = $doctrine->getManager();
            $em->flush();

            $this->addFlash(type: 'success', message: 'Відгук оновлено.');
```

[

```
            return $this->redirect($this->generateUrl(route: 'Admin.reviewsGet'));
        }

        return $this->render(view: 'admin_panel/reviews/edit.html.twig', parameters:
            'form' => $form->createView()
        );
    ]

    /** contacts */

    #[Route('/contacts', name: 'contactsGet', methods: ['GET'])]
    public function contactsGet(ContactUsRepository $contactUsRepository, Request
$request, EntityManagerInterface $em): Response
    {

        $search = $request->query->get('search');
        $sortBy = $request->query->get('contact-sort', 'DESC');
        $page = $request->query->get('page', 1); // Get the current page, default to
1
        $limit = $request->query->get('limit', 5); // Get the limit, default to 10
        $offset = ($page - 1) * $limit; // Calculate the offset
        $contacts = $contactUsRepository->search($search, $sortBy, $limit, $offset,
$totalItems);

        $Pages = ceil($totalItems / $limit); // Calculate the total number of pages
        return $this->render('admin_panel/contacts/index.html.twig', [
            'contacts' => $contacts,
            'search' => $search,
            'sortBy' => $sortBy,
            'Pages' => $Pages,
            'totalItems' => $totalItems,
            'viewedItems' => count($contacts),
            'page' => $page,
            'limit' => $limit,
        ]);
    }

    #[Route('/contacts/delete/{id}', name: 'contacts.delete')]
    public function contactRemove(ManagerRegistry $doctrine, ContactUs $contact)
    {
        $em = $doctrine->getManager();

        $em->remove($contact);
        $em->flush();
        $this->addFlash(type: 'success', message: 'Контакт видалено.');
```

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						70

```

        return $this->redirect($this->generateUrl(route: 'Admin.contactsGet'));
    }

    #[Route('/contacts/edit/{id}', name: 'contacts.edit')]
    public function contactEdit(ManagerRegistry $doctrine, ContactUs $contactUs,
Request $request, ContactUsRepository $contactUsRepository)
    {
        $reviewOne = $contactUsRepository->find($contactUs);
        $form = $this->createForm(type: ContactUsEditType::class, data: $reviewOne);

        $form->handleRequest($request);

        if ($form->isSubmitted()) {
            $em = $doctrine->getManager();
            $em->flush();

            $this->addFlash(type: 'success', message: 'Відгук оновлено.');
```

return \$this->redirect(\$this->generateUrl(route: 'Admin.contactsGet'));

```

        }

        return $this->render(view: 'admin_panel/contacts/edit.html.twig', parameters:
[
            'form' => $form->createView()
]);
    }

    //Question
    #[Route('/questions', name: 'questionGet', methods: ['GET'])]
    public function questionGet(QuestionRepository $questionRepository, Request
$request): Response
    {
        $search = $request->query->get('search');
        $sortBy = $request->query->get('question-sort');
        $page = $request->query->get('page', 1); // Get the current page, default to
1
        $limit = $request->query->get('limit', 5); // Get the limit, default to 10
        $offset = ($page - 1) * $limit; // Calculate the offset

        $questions = $questionRepository->search($search, $sortBy, $limit, $offset,
$totalItems);

        $Pages = ceil($totalItems / $limit); // Calculate the total number of pages
        return $this->render('admin_panel/question/index.html.twig', [
            'questions' => $questions,
            'search' => $search,
            'sortBy' => $sortBy,
            'Pages' => $Pages,
            'totalItems' => $totalItems,
            'viewedItems' => count($questions),
            'page' => $page,
            'limit' => $limit,
        ]);
    }

    /** USERS */
    #[Route('/users', name: 'users', methods: ['GET'])]
    public function usersShow(Request $request, UserRepository $userRepository):

```

```

Response
{
    $search = $request->query->get('search');
    $page = $request->query->get('page', 1); // Get the current page, default to
1    $limit = $request->query->get('limit', 5); // Get the limit, default to 10
    $offset = ($page - 1) * $limit; // Calculate the offset

    $users = $userRepository->search($search, $limit, $offset, $totalItems);

    $Pages = ceil($totalItems / $limit); // Calculate the total number of pages
    return $this->render('admin_panel/users/users.html.twig', [
        'users' => $users,
        'search' => $search,
        'Pages' => $Pages,
        'totalItems' => $totalItems,
        'viewedItems' => count($users),
        'page' => $page,
        'limit' => $limit,
    ]);
}

#[Route('/users/{id}', name: 'user', methods: ['GET'])]
public function usersView(UserRepository $userRepository, $id): Response
{
    $users = $userRepository->find($id);
    return $this->render('admin_panel/users/view.html.twig', [
        'users' => $users
    ]);
}

#[Route('/users/edit/{id}', name: 'edit_user')]
public function editUser(UserRepository $userRepository, $id, Request $request):
Response
{
    $user = $userRepository->find($id);
    $form = $this->createForm(UserFormType::class, $user);

    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $user->setNickname($form->get('nickname')->getData());
        $user->setPhoneNumber($form->get('phone_number')->getData());
        $user->setEmail($form->get('email')->getData());
        $user->setRoles($form->get('roles')->getData());

        $this->em->flush();
        return $this->redirectToRoute('Admin.users');
    }

    return $this->render('admin_panel/users/edit.html.twig', [
        'user' => $user,
        'form' => $form->createView()
    ]);
}

/** ARTICLES
 * @param ArticleRepository $articleRepository

```

```

    * @param Request $request
    * @param EntityManagerInterface $em
    * @param $totalItems
    * @return Response
    */
    #[Route('/articles', name: 'articles', methods: ['GET'])]
    public function articleList(ArticleRepository $articleRepository, Request
    $request, EntityManagerInterface $em): Response
    {
        $articles = $articleRepository->findBy([], ["date"=>"DESC"]);

        $search = $request->query->get('search');
        $sortBy = $request->query->get('article-sort', 'DESC');
        $articleStatus = $request->query->get('article-status', null);
        $page = $request->query->get('page', 1); // Get the current page, default to
1
        $limit = $request->query->get('limit', 5); // Get the limit, default to 10
        $offset = ($page - 1) * $limit; // Calculate the offset

        $articleStatus = ($articleStatus !== "1" && $articleStatus !== "0") ? null :
        $articleStatus;
        $articles = $articleRepository->new_search($search, $articleStatus, $sortBy,
        $limit, $offset, $totalItems);
        $pages = ceil($totalItems / $limit); // Calculate the total number of pages

        return $this->render('admin_panel/articles/articles.html.twig', [
            'articles' => $articles,
            'search' => $search,
            'sortBy' => $sortBy,
            'articleStatus' => $articleStatus,
            'pages' => $pages,
            'totalItems' => $totalItems,
            'viewedItems' => count($articles),
            'page' => $page,
            'limit' => $limit,
        ]);
    }

    #[Route('/articles/create', name: 'create_article')]
    public function createArticle(Request $request, SluggerInterface $slugger):
    Response
    {
        $article = new Article();
        $form = $this->createForm(ArticleFormType::class, $article);

        $form->get("date")->setData(new DateTime()); //today by default

        $form->handleRequest($request);
        if ($form->isSubmitted() && $form->isValid()) {
            $newArticle = $form->getData();
            $imagePath = $form->get('image')->getData();
            if ($imagePath) {

                $originalFilename = pathinfo($imagePath->getClientOriginalName(),
                PATHINFO_FILENAME);
                // this is needed to safely include the file name as part of the URL
                $safeFilename = $slugger->slug($originalFilename);
                $newFilename = $safeFilename . '-' . uniqid() . '.' . $imagePath->

```

```

>guessExtension();

        // Move the file to the directory where brochures are stored
        try {
            $imagePath->move(
                $this->getParameter('kernel.project_dir') .
'/public/uploads',
                $newFilename
            );
        } catch (FileNotFoundException $e) {
        }
        $article->setImage($newFilename);
    }
    $this->em->persist($newArticle);
    $this->em->flush();

    return $this->redirectToRoute('Admin.articles');
}

return $this->render('/admin_panel/articles/create.html.twig', [
    'form' => $form->createView()
]);
}

#[Route('/articles/delete/{id}', name: 'delete_article', methods: ['GET',
'DELETE'])]
public function deleteArticle(ArticleRepository $articleRepository, $id):
Response
{
    $article = $articleRepository->find($id);
    $this->em->remove($article);
    $this->em->flush();

    return $this->redirectToRoute('Admin.articles');
}

#[Route('/articles/{id}', name: 'article', methods: ['GET'])]
public function articleView(ArticleRepository $articleRepository, $id): Response
{
    $article = $articleRepository->find($id);
    return $this->render('admin_panel/articles/view.html.twig', [
        'article' => $article
    ]);
}

#[Route('/articles/edit/{id}', name: 'edit_article')]
public function editArticle(ArticleRepository $articleRepository, $id, Request
$request, SluggerInterface $slugger): Response
{
    $article = $articleRepository->find($id);
    $form = $this->createForm(ArticleFormType::class, $article);

    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()) {
        $imagePath = $form->get('image')->getData();
        if ($imagePath) {
            $destination = $this->getParameter('kernel.project_dir') .
'/public/uploads';

```

```

        $originalFilename = pathinfo($imagePath->getClientOriginalName(),
PATHINFO_FILENAME);
        $safeFilename = $slugger->slug($originalFilename);
        $newFilename = $safeFilename . '-' . uniqid() . '.' . $imagePath-
>guessExtension();

        $imagePath->move(
            $destination,
            $newFilename
        );

        $article->setImage($newFilename);
    } else {
        $article->setTitle($form->get('title')->getData());
        $article->setContent($form->get('content')->getData());
        $article->setDate($form->get('date')->getData());
        $article->setStatus($form->get('status')->getData());
    }

    $this->em->persist($article);
    $this->em->flush();

    return $this->redirectToRoute('Admin.articles');
}

return $this->render('admin_panel/articles/edit.html.twig', [
    'article' => $article,
    'form' => $form->createView()
]);
}

#[Route('/questions/delete/{id}', name: 'question.delete')]
public function questionRemove(ManagerRegistry $doctrine, Question $question)
{
    $em = $doctrine->getManager();

    $em->remove($question);
    $em->flush();
    $this->addFlash(type: 'success', message: 'Запитання видалено.');
```

Додаток Б

```

#[Route('/', name: 'Main.')]
class MainController extends AbstractController
{
    private EntityManager;

    public function __construct(EntityManagerInterface $entityManager)
    {
        $this->entityManager = $entityManager;
    }

    /** HEADER SECTION */
    public function sectionHeader(): Response
    {
        return $this->render('header.html.twig', [

```

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						75

```

        'user' => 'MainController',
    ]);
}

/** USER VISITOR */
#[Route('/', name: 'VisitorRegister', methods: ["POST"])]
public function visitorRegister(ManagerRegistry $doctrine, Request $request,
UserRepository $userRepository)
{
    $rep = $doctrine->getManager()->getRepository(Statistics::class);

    $user = $this->getUser();
    $question = new Question();

    if ($user == null) {
        $questionForm = $this->createForm(QuestionType::class, $question);
    } else {
        $questionForm = $this->createForm(AuthorizedQuestionType::class,
$question);
    }

    $questionForm->handleRequest($request);
    if ($questionForm->isSubmitted() && $questionForm->isValid()) {
        $now = new \DateTime('@' . strtotime('now'));
        $now->setTimezone(new \DateTimeZone('Europe/Kiev'));
        $question->setDatetime($now);

        if ($user == null) {
            $phoneNumber = $question->getPhone(); // phone number in
international format
            $cleanedNumber = "+" . preg_replace("/[^\d]/", "", $phoneNumber);
            $question->setPhone($cleanedNumber);
        } else {
            $question->setEmail($user->getEmail());
            $question->setPhone($user->getPhoneNumber());
            $question->setNickname($user->getNickname());
        }
        $this->entityManager->persist($question);
        $this->entityManager->flush();

        return new RedirectResponse($this->generateUrl('Main.Index'));
    }

    $contact = new ContactUs();
    $contactForm = $this->createForm(ContactUsType::class, $contact);
    $contactForm->handleRequest($request);
    if ($contactForm->isSubmitted() && $contactForm->isValid()) {

        $now = new \DateTime('@' . strtotime('now'));
        $now->setTimezone(new \DateTimeZone('Europe/Kiev'));
        $contact->setDatetime($now);
        $phoneNumber = $contact->getPhone(); // phone number in international
format
        $cleanedNumber = "+" . preg_replace("/[^\d]/", "", $phoneNumber);
        $contact->setPhone($cleanedNumber);
        $this->entityManager->persist($contact);
        $this->entityManager->flush();
    }
}

```

```

        return new RedirectResponse($this->generateUrl('Main.Index'));
    }

    if ($user != null && $user->getCompanyReview() != null) {
        $review = $user->getCompanyReview();
    } else {
        $review = new CompanyReview();
    }

    $reviewForm = $this->createForm(CompanyReviewType::class, $review);
    $reviewForm->handleRequest($request);
    if ($reviewForm->isSubmitted() && $reviewForm->isValid()) {

        $now = new \DateTime('@' . strtotime('now'));
        $now->setTimezone(new \DateTimeZone('Europe/Kiev'));
        $user = $this->getUser();

        $review->setUser($user);
        $review->setHome(false);
        $review->setDatetime($now);
        $this->entityManager->persist($review);
        $this->entityManager->flush();

        return new RedirectResponse($this->generateUrl('Main.Index'));
    }

    return new RedirectResponse($this->generateUrl('Main.Index'));
}

/** ROUTES */

#[Route('/', name: 'Index', methods: ["GET"])]
public function index(ManagerRegistry $doctrine): Response
{
    $data = $doctrine->getManager()->getRepository(Statistics::class)-
>getInstance();
    $articles = $doctrine->getRepository(Article::class)->findBy([], ["date" =>
"DESC"], 3);
    $reviews = $doctrine->getRepository(CompanyReview::class)->findBy(["home" =>
"1"], ["rating" => "DESC"]);
    ///
    $user = $this->getUser();
    $question = new Question();
    if ($user == null) {
        $questionForm = $this->createForm(QuestionType::class, $question);
    } else {
        $questionForm = $this->createForm(AuthorizedQuestionType::class,
$question);
    }
    ///
    $contact = new ContactUs();
    $contactForm = $this->createForm(ContactUsType::class, $contact);
    ///
    $review = new CompanyReview();
    $reviewForm = $this->createForm(CompanyReviewType::class, $review);

    return $this->render('main/index.html.twig', [
        'questionForm' => $questionForm,

```

```

        'contactForm' => $contactForm,
        'reviewForm' => $reviewForm,
        'statistics' => $data,
        'articles' => $articles,
        'reviews' => $reviews,
        'isLoggedIn' => $this->getUser() instanceof
\Symfony\Component\Security\Core\User\UserInterface,
    ]);
}

#[Route('/about', name: 'About', methods: ["GET"])]
public function about(TestService $service): Response
{
    $service->SomeFunc();
    return $this->render('main/about.html.twig', [
        'controller_name' => 'MainController',
    ]);
}

#[Route('/search', name: 'SearchHandler', methods: ["GET"])]
public function searchHandler(ManagerRegistry $doctrine, Request $request):
Response
{
    $pageSize = 2; //amount of articles per page

    $query = $request->query->get('q');
    $page = intval($request->query->get('p'));
    if ($page <= 1)
        $page = 1;

    $articles = $doctrine->getRepository(Article::class)->search($query,
$pageSize, $page);

    return $this->render('main/search.html.twig', [
        'articles' => $articles,
        'search' => $query,
        'page' => $page,
        'pageSize' => $pageSize
    ]);
}

//TODO: Obsolete
public function searchBar(): Response
{
    $form = $this->createForm(SearchFormType::class);

    return $this->render('parts/forms/searchBar.html.twig', [
        'searchBar' => $form->createView(),
    ]);
}

}

public function __construct(private ManagerRegistry $doctrine) {}
#[Route('/contact', name: 'app_contact_us')]

```

Додаток В

					ІІЗ.КР.Б-121-24-ІІЗ	Арк.
						78

```

public function index(Request $request)
{
    $contact = new ContactUs();

    $form = $this->createForm(ContactUsType::class, $contact);

    $form ->handleRequest($request);

    if($form->isSubmitted() && $form->isValid())
    {
        $now = new \DateTime('@' . strtotime('now'));
        $now->setTimezone(new \DateTimeZone('Europe/Kiev'));
        $contact->setDatetime($now);
        $phoneNumber = $contact->getPhone(); // phone number in international format
        $cleanedNumber = "+" . preg_replace("/[^0-9]/", "", $phoneNumber);
        $contact->setPhone($cleanedNumber);
        $em = $this->doctrine->getManager();
        $em->persist($contact);
        $em->flush();
        return new RedirectResponse($this->generateUrl('Main.Index'));
    }

    return $this->render(view: 'contact_us/index.html.twig', parameters: [
        'form'=>$form,
    ]);
}

```