Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

з напряму підготовки 121 «Інженерія програмного забезпечення»

На тему: *Розробка сервісного застосунку для розрахунку необхідних обсягів будівельно-оздоблювальних матеріалів*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-20-2
_____ *Школьний О.О.*

Керівник кваліфікаційної роботи _____ */ Гриценко А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2024

Криворізький національний університет

Факультет: *Інформаційних технологій*

Кафедра: *Моделювання та програмного забезпечення*

Освітньо-кваліфікаційний рівень: *бакалавр*

Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

студента групи ІПЗ-20-2 Школьний Олександр Олександрович

1. Тема: *Розробка сервісного застосунку розрахунку необхідних обсягів будівельно-оздоблювальних матеріалів* затверджено наказом по КНУ №___ від «__» лютого 2024 р.
2. Термін подання студентом закінченого проекту «10» червня 2024 р.
3. Вихідні дані по роботі: *Розроблений застосунок повинен виконувати розрахунок обсягів необхідних будівельних матеріалів.*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): .
Критичний аналіз інформаційних систем будівельної галузі, визначення мети та завдань, розробка архітектури системи, структури бази даних, розробка інтерфейсу. Тестування системи.
5. Перелік графічного демонстраційного матеріалу*: Приклади існуючих аналогів програм. Функціональні схеми режимів роботи,. Діаграма бази даних. Інтерфейси програми в режимах роботи.*

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*09.02.2024-17.02.2024*
2	*Аналіз інформаційних систем будівельної галузі*	*23.02.2024-11.03.2024*
3	*Визначення вимог до програмного продукту*	*13.03.2024-16.03.2024*
4	*Розробка функціональних схем*	*17.03.2024-24.03.2024*
5	*Розробка структури таблиць бази даних*	*26.03.2024-31.03.2024*
6	*Розробка програмного забезпечення*	*03.04.2024-15.04.2024*
7	*Тестування програмного забезпечення*	*16.05.2024-22.05.2024*
8	*Оформлення пояснювальної записки*	*22.05.2024-13.06.2024*
9	*Розробка демонстраційних матеріалів*	*13.06.2024-17.06.2024*

Дата видачі завдання: «_____»_____ 2024 р.
Студент _____ Школьний О.О.
Керівник роботи _____ Гриценко А. М.

РЕФЕРАТ

СЕРВІСНИЙ ЗАСТОСУНОК, ВАРТІСТЬ БУДІВЕЛЬНИХ МАТЕРІАЛІВ, ВІДДАЛЕНИЙ ДОСТУП, ПРОГРАМУВАННЯ, PHP.

Пояснювальна записка: 62 с., 35 рис., 8 джерел.

В кваліфікаційній роботі розроблено та створено програмний продукт для розрахунтку витрат матеріалів для будівництва та оздоблювальних робіт.

Проведено детальний аналітичний огляд систем будівельної галузі, з’ясовані їх переваги та недоліки.

Була спроектована архітектура розробки на основі шаблону MVC, створена база даних та розроблена серверна частина, яка відповідає за обробку запитів і реалізацію функціональності. Механізм міграції використовувався під час розробки з метою скорочення часу розробки. Розроблено графічний інтерфейс веб-застосунку, що відповідає предметній області

В результаті кваліфікаційної роботи було розроблено готовий застосунок, за допомогою якого користувачі можуть здійснювати розрахунки необхідних обсягів будівельно-оздоблювальних матеріалів і їх вартостей.

ABSTRACT

SERVICE APPLICATION, COST OF BUILDING MATERIALS, REMOTE ACCESS, PROGRAMMING, PNR.

Explanatory note: 62 pp., 35 pictures, 8 sources.

In the qualification work, a software product was developed and created for calculating the consumption of materials for construction and finishing works.

A detailed analytical review of construction industry systems was conducted, their advantages and disadvantages were clarified.

The development architecture was designed based on the MVC pattern, the database was created and the server part was developed, which is responsible for processing requests and implementing functionality. The migration mechanism was used during development to reduce development time. The graphical interface of the web application corresponding to the subject area has been developed

As a result of the qualification work, a ready-made application was developed, with the help of which users can calculate the necessary volumes of construction and finishing materials and their costs.

ЗМІСТ

ВСТУП

Сьогодні ефективне управління проектами є одним з найважливіших факторів успіху в будівельній галузі. Як наслідок, будівельні компанії все частіше використовують програмне забезпечення для аналізу економічної ефективності проектів.

У будівельній галузі важливість розробки програмного забезпечення для аналізу ефективності проектів полягає у зменшенні ризиків та витрат, а також у підвищенні якості та ефективності будівельного процесу. Програмне забезпечення може допомогти збирати, обробляти та аналізувати великі обсяги проектних даних, включаючи фінансові, технічні та графічні дані.

Крім того, за допомогою правильного програмного забезпечення можна ефективно контролювати роботу персоналу та вирішувати проблеми в режимі реального часу. Як результат, компанії можуть забезпечити дотримання термінів виконання проектів та економію ресурсів.

Тому розробка програмного забезпечення для аналізу виконання проектів у будівельній галузі є надзвичайно важливим питанням на сьогоднішній день. Це дозволить підвищити ефективність роботи в будівельній галузі та забезпечити успіх проектів.

Тому метою даної кваліфікаційної роботи є розробка веб-сайту для ефективного розрахунку витрат матеріалівв у будівельній галузі та візуалізації ключових показників ефективності вибору певних матеріалів.

1. АНАЛІЗ СУЧАСНОГО СТАНУ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Застосування програмних продуктів в будівельній сфері

Інформаційне суспільство змінюється, як і наше життя, особливо наші цінності. Матеріальність витісняє знання, час і культуру і рухається до них. В інформаційному суспільстві примат розумової праці супроводжується виробництвом і споживанням знань та інтелекту. Тому вміння творчо мислити відіграє важливу роль, що призвело до появи багатьох інформаційних технологій та їх широкого застосування в сферах життя[1].

У суспільстві інформаційні технології використовуються в багатьох сферах людської діяльності. Системи та програми виконують рутинні та відповідальні функції в освіті, економіці, бухгалтерії, економіці, медицині, та будівництві. Сьогодні будівельна галузь не є винятком і досягла значного прогресу з використанням інформаційних технологій. Відбуваються значні зміни з впровадженням інформаційних технологій в роботу будівельників, менеджерів, архітекторів та клієнтів. Комп'ютеризація допомагає від початкового етапу ідеї до створення та реалізації проекту, візуалізації результатів, підготовки розрахунків та кошторисів, побудови структури та управління об'єктами [1].

У будівельній галузі впровадження інформаційних технологій почалося з автоматизації обчислень, що до теперішнього часу еволюціонувало в системи управління великими та складними проектами, як загальнобудівельними, комунікаційниими, так і до автоматизованих систем контролю та управління об'єктами державного (спеціального) нагляду .

У сучасних масштабних будівельних проектах все частіше використовуються інформаційні технології та спеціалізоване програмне забезпечення. Першими з них є системи автоматизованого проектування (САПР). Що автоматизують виконання таких завдань, як:

- архітектурне планування;

- розв'язання задач проектного планування;
- розв'язання конструкторських задач;
- розрахунок механічних властивостей конструкцій;
- управління процесом будівництва [6].

Популярними програмними продуктами є AutoCAD, Allplan, Revit, ArchiCAD, та SCAD Office

ArchiCAD вважається найкращим додатком, що використовується в будівництві та архітектурному проектуванні; інформаційна технологія ArchiCAD дозволяє створювати віртуальні моделі реальних конструкцій за допомогою інструментів з реалістичними аналогами (колони, стіни, вікна, перекриття тощо) [7]. Наприклад, зміни плану будівлі автоматично відображаються в розрізах, специфікаціях та описах. Такий підхід значно скорочує час проектування. Також автоматично створюється документація [8]. Програма має деякі недоліки, які потребують доопрацювання:

- має обмежені можливості для створення об'єктів нестандартної складної форми (наприклад, скульптурне моделювання);
- не пропонує кілька варіантів дизайну;
- має високу вартість ($1540) [9].

Відомо про інформаційні системи, призначені для управління будівельними компаніями. До них відносяться "1С: Управління будівельною організацією", "1С: Будівельні підрядники. Управління фінансами". Ці системи допомагають створювати програми та керувати виконанням робіт. Вони також можуть обмінюватися даними з кошторисними та фінансовими програмами [10].

Тому моделювання відіграє важливу роль. Сучасні замовники хочуть не тільки високоякісні будівлі, але й довговічні, якісні та з мінімальними витратами. Для вирішення цих та інших питань використовується технологія інформаційного моделювання будівель (BIM) [11]. Переваги BIM значні. Відмінність від інших розробок у проектуванні будівельних об'єктів, які створюють геометричні зображення, BIM-моделювання вибудовує модель в

цифровій формі з повною інформацією про об'єкт і процес будівництва. Використання цієї методики проектування робить всі етапи прозорими, гарантує повний контроль і забезпечує високу якість проектних і будівельних робіт [13].

Слід зазначити, що все вищевказане відноситься до великих будівельних корпорацій, робота яких зосереджена з масштабним будівництвом. Для менших компаній, що займаються реставрацією та ремонтом будівель, потреби в інформаційних технологіях є іншими. Інформаційні технології в будівельній галузі можуть полегшити ці завдання, скоротити час підготовки та усунути помилки, оскільки дозволяють автоматично перевіряти розрахунки та готувати форми до друку [14].

На ринку існують системи, які можуть допомогти підготувати відповідні розрахунки. Деякі з найпопулярніших з них: MSmeta, JobNimbus, Будівельний калькулятор онлайн, SMETA AC-4, Будівельні технології, Estimates8, Construction Estimating & Job Costing Software.

Важливість використання веб-сайтів як рекламного продукту для будівельної галузі важко переоцінити. Адже веб-сайти - це сегмент інформаційної індустрії, що стрімко розвивається. Більшість представників компаній вже не уявляють своєї діяльності без цього інструменту. Сайти замінюють спілкування з клієнтами, які не тільки дізнаються про компанію з інформаційних блоків, що описують діяльність, досягнення та результати організації у вигляді фото- та відеозвітів, але й мають можливість ознайомитися з відгуками людей, які вже скористалися послугами компанії. Якщо у потенційних клієнтів виникають додаткові питання, вони можуть зв'язатися з компанією за номером телефону, який також вказаний в контактних даних. Поширеним варіантом є реєстрація та створення особистого кабінету, в якому вже відбувається процес введення замовлень та їх виконання. Подібні рішення позитивно впливає на статус компанії, розширює клієнтську базу та виступає рекламою її діяльності.

Сучасні інформаційні технології набувають все більшого поширення в багатьох сферах життедіяльності людини. Різні форми і методи (комп'ютерні програми, веб-сайти, соціальні мережі) використовуються для вирішення серйозних і специфічних завдань IT також добре зарекомендували себе в будівництві та будівельній галузі. Сучасні сервіси відіграють роль помічників як для сформованих професіоналів, так і для студентів на етапі становлення. Замовники також використовують Інтернет для вирішення багатьох проблем, пов'язаних з ремонтом квартир і будинків, а IT-арсенал постійно вдосконалюється і набуває нових форм, які прискорюють роботу, покращують результати і знижують витрати [2].

1.2 Огляд програмних продуктів будівельної галузі

Вище розглянуто використання інформаційних технологій у будівельному секторі та визначено значні переваги цього. Розглянемо сектор малого будівництва, який виконує невеликі за обсягом роботи, але є більш поширеним для населення.

Будівельні проекти ніколи не розпочинаються без оцінки вартості будівництва, оскільки вартість матеріалів потрібно оцінити заздалегідь.

Для того, щоб отримати вартість, визначену шляхом переговорів, необхідно врахувати всі пов'язані з будівництвом роботи та ресурси. Кошторис повинен бути підготовлений відповідно до нормативних документів. Кошториси готуються кошторисниками, які пройшли професійну підготовку і засвоїли хоча б базові поняття бухгалтерії та нормування у будівельному секторі.

Розглянемо процес будівництва, що включає складання кошторисів на роботи та матеріали через веб-сайт (рис. 1.1).

Замовник звертається до компанії безпосередньо або через веб-сайт. Надсилається запит на будівництво, відбувається обмін інформацією щодо потреб та побажань клієнта, умов підрядника та цінової політики. Дані потенційного замовника вносяться до бази даних клієнтів.

Наступним етапом є огляд приміщення. Представники обох сторін обговорюють спеціальні роботи, які необхідно виконати на місці, та зміни в проектній документації. Прийнято, що, подібні проектні роботи можуть узгоджуватися декілька разів (наприклад до і після консультації з спеціалістами). Інформація надається клієнту електронною поштою. Замовник може вносити певні коригування в документацію.

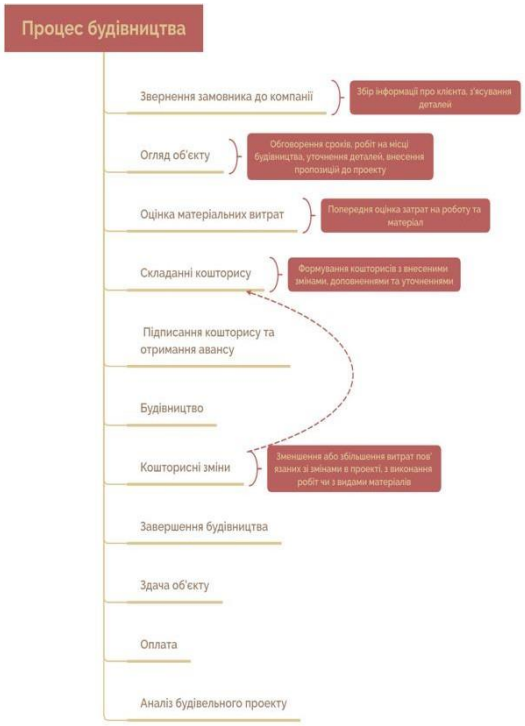


Рисунок 1.1 – Узагальнена схема узгоджень в процесі будівництва

Після внесення змін і доповнень обидві сторони підписують пропозицію, здійснюється перший внесок (передоплата) і починається будівництво. Під час виконання робіт повідомляється про відхилення від початкового проекту, оскільки існує ймовірність внесення додаткових коригувань до кошторису. Також приймаються рішення про авансові платежі та зміни в програмі будівництва [17].

Після завершення будівництва об’єкт передається клієнту, проводиться його інспекція і, за необхідності, виправляються забраковані роботи. Якщо проект схвалено, замовник і підрядник здійснюють розрахунки.

Ще одним важливим аспектом є аналіз завершеного будівельного процесу. На цьому етапі оцінюється якість, технологія, хід виконання робіт, безпека будівництва, матеріальні збитки та економія матеріальних витрат.

На рис. 1.2 показано двоступеневу інформаційну структуру для обробки даних у будівельній документації. Первинні та звітні документи посилаються на попередні та узгоджені кошториси робіт і матеріалів, які повинні автоматично генеруватися відповідно до обраного формату. Тому інформаційні системи повинні бути розроблені для ефективної та швидкої обробки даних.

Рисунок 1.2 – Структура інформаційних потоків при підготовці документації в будівельній сфері

Як відомо, системи документообігу, в тому числі в будівельній галузі, є складними. Це означає, що неможливо розпізнати всю інформацію та зв'язки між елементами системи в цілому. Тому необхідно розділити систему на більш прості частини, тобто інформаційні структури. Це зменшує складність,

забезпечує комунікацію та гнучкість і спрощує кожен етап процесу побудови та управління (рис. 1.2) [18, 19].

Першочергово звітна документація оформлюється на основі наданих користувачем інформації. Враховуючи, що інформація може бути введена безпосередньо на будівельному майданчику під час розміщення замовлення, необхідно розробити веб-інформаційну систему розрахунку витрат матеріалів та вартості будівництва для забезпечення віддаленого доступу до даних.

1.3 Аналіз програмних продуктів для розрахунку витрат матеріалів на будівництві.

1) COCONSTRUCT - це програмне забезпечення преміум-класу для оцінки будівництва. Вона пропонує найбільше функцій та переваг, та має сприйнятливу вартість. Тарифні плани включають базову щомісячну плату, та відсутнє обмеження кількості користувачів. Високо оцінюється клієнтський сервіс, тобто якісні консультації щодо роботи програми. Якщо у вас виникли проблеми, ви можете отримати підтримку за телефоном або надіслати електронного листа спеціалісту. Також наявний зворотній зв'язок з службою підтримки та пропонує великий набір навчальний курсів для практичний навичок роботи та велику колекцію відеоматеріалів з уроками. [23].

2) STACK – недорогий та потужний програмний продукт для оцінки вартості будівництва. Програма безкоштовного використання стане у пригоді малому бізнесу. STACK також надає велику кількість занять тренінгів та підтримку незалежно від плану обслуговування. На додаток до безкоштовних планів обслуговування, існують також платні плани; STACK пропонує чотири плани обслуговування (один безкоштовний і три платні). Платні плани передбачають щорічну плату, яка залежить від кількості користувачів. Безкоштовний план включає наступні можливості на одного користувача на рік: два

одночасних проекти, сім днів доступу до проекту, десять метрик на проект і багатокористувацька співпраця в режимі реального часу. Платні плани включають наступні функції: послуги зі створення власних елементів, навчання на місці, а також спільна робота, яка спрощується завдяки можливості доступу до STACK з будь-якого підключеного до Інтернету комп'ютера, планшета або мобільного пристрою STACK - це хмарний додаток, тому оновлення та вдосконалення відбуваються автоматично, не вимагаючи від вас жодних дій.

STACK має деякі недоліки. Безкоштовний план обслуговування має ряд обмежень, що призводить до вимушеного придбання платних тарифів.

3) Acumatica Cloud ERP - хмарна та мобільна технологічна платформа. Яка націлена на клієнтів середнього ринку, вона надає повний огляд їхнього бізнесу в режимі реального часу будь-де, будь-коли, на будь-якому гаджеті, де працює браузер. Пакет додатків Acumatica включає управління фінансами, дистрибуцію, виробництво, планування та управління часом і витратами, та інше.

Всі функції повноцінно інтегровані і побудовані з використанням стандартних інструментів Microsoft з самообслуговуванням звітності, документообігу та робочих процесів затвердження. Вони в змозі налаштовувати робочі області та візуальні інтерфейси. Також можуть розгорнути проект на віддаленому сервері або перенести на сучасний хмарний додаток [25, 26].

Основні порівняльні характеристики розглянутих програмних продуктів будівельної галузі наведено в табл. 1.1

Таблиця 1.1. Порівняльна характеристик програмних продуктів

	CoConstruct	**Acumatica Cloud ERP**	**STACK**
Безкоштовне використання	Demo-версия	Demo-версия	Demo-версия
Безкоштовні версії	Ні	Ні	Так
Вартість	$49 60 днів 199-299$ за місяць	15000-40000$	$999 до $5999 999-5999$
Обмеження в кількості користувачів	Так	Ні	Ні
Позподілене використанння	Cloud, WWW	Cloud, WWW	Cloud, WWW
Інтеграція	QuickBooks	Business, Dropbox, Microsoft	
Простота використання	Потребує навчання користуванню	Потребує навчання користуванню	

Отже, сучасне програмне забезпечення для будівництва є дуже складним і гнучким. Всі вони мають багато спільних рис. Переважна більшість яких орієнтовані на великі корпорації, що беруть участь у масштабних будівельних проектах. Звичайно, такі проекти вимагають безлічі ретельних і точних розрахунків.

Великі проекти вимагають ретельних і точних розрахунків, які безпосередньо впливають на прибуток компанії та її рейтинг серед конкурентів. Тому не в їхніх інтересах економити на таких програмах, хоча ціна такого програмного забезпечення досить висока. Але для малого бізнесу ця ціна є неприйнятною.

Особливо цього року малий та середній бізнес зазнав значних збитків. Звичайно, дозволяти використовувати таке будівельне програмне забезпечення з широким доступом до різних частково використовуваних функцій не є практичним рішенням. Використання демонстраційних або безкоштовних версій не вирішує проблему для невеликих будівельних компаній, оскільки вони обмежені як у часі, так і в функціональності [27].

Впровадження масштабного програмного комплексу, що автоматизує широкий спектр видів робіт у будівельній компанії, є набагато проблематичнішим і вимагає багато часу та коштів, а також досить високого рівня компетенції користувачів. Враховуючи ціни на вищезгадані продукти, бажано використовувати програмні продукти, які можуть бути створені для конкретних організацій. Тому основним рішенням для досягнення оптимального результату є самостійна розробка системи, орієнтована на конкретного користувача з його невеликим проектом.

1.4 Мета та завдання розробки

Метою кваліфікаційної роботи є розробка централізованої інформаційної системи для створення будівельних кошторисів.

Оптимальним рішенням є розробка веб-сайту. Для менеджерів будівельних проектів важливо мати можливість готувати попередні кошториси в польових умовах разом зі своїми клієнтами, заощаджуючи їх час і підкреслюючи свою кваліфікацію.

Переважною більшістю користувачами централізованої інформаційної системи є, зацікавлені у будівельних та ремонтних послугах потенційні клієнти, а також, підрядники (будівельники, бухгалтери), фахівці, які готують і використовують кошториси, та адміністратори, які оперують базою даних і керують сайтом.

Сервісний застосунок повинен бути доступним через мережу Інтернет і простим у використанні як для підрядників, так і для клієнтів. Оскільки

інформаційна система реалізовується з використанням технологій HTML, CSS, JS, Vue.js (клієнтська частина), PHP, Laravel (серверна частина), та MySQL (база даних), в якій зберігаються всі дані, адміністратори повинні мати спеціальні технічні навички. З іншого боку, користувачам достатньо загальних навичок роботи з комп'ютером і веб-браузерами.

Веб-сайт повинен мати права доступу і бути розділений на дві частини: клієнтську та адміністраторську.

Сайт для клієнтів повинен бути інформативним і рекламним. Це означає, що при відвідуванні сайту потенційні клієнти дізнаються про діяльність компанії, результати виконаних робіт, а також відвідають сторінки з реальними відгуками про надані послуги. Зацікавлені клієнти можуть скористатися контактними даними, такими як номер телефону, адреса електронної пошти або відвідати офіс за конкретною адресою за допомогою Google Maps.

На рисунку 1.3 наведено карту-схему сайту, на якій показано зв'язки між розділами та підрозділами.

Люди, які хочуть скористатися послугами компанії, створюють власний обліковий запис і реєструються на сайті самостійно або за допомогою представника компанії. На рисунку 1.4 зображено схематично функціонал облікового запису клієнта. Доступні наступні меню (інформаційна панель):

- налаштування;
- мої проекти;
- зворотній зв'язок;
- ціни на матеріали.

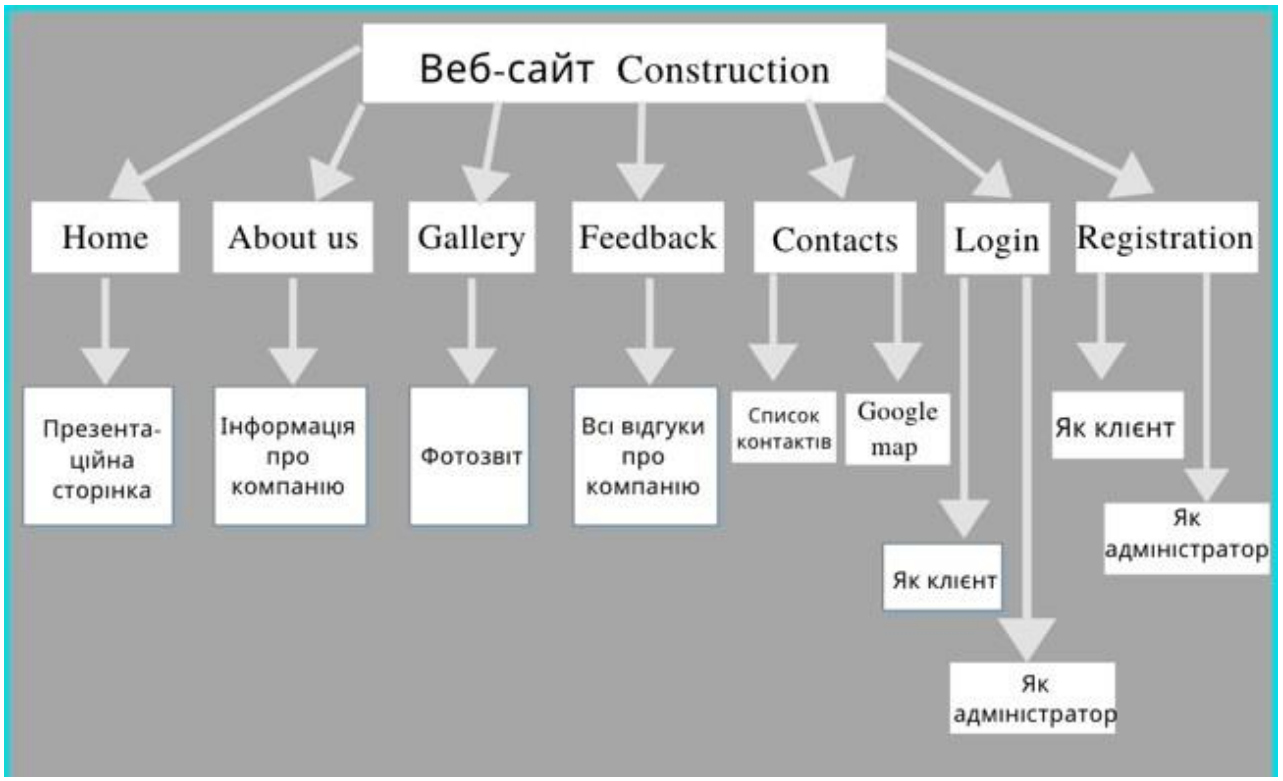


Рисунок 1.3 – Схема веб-сайту

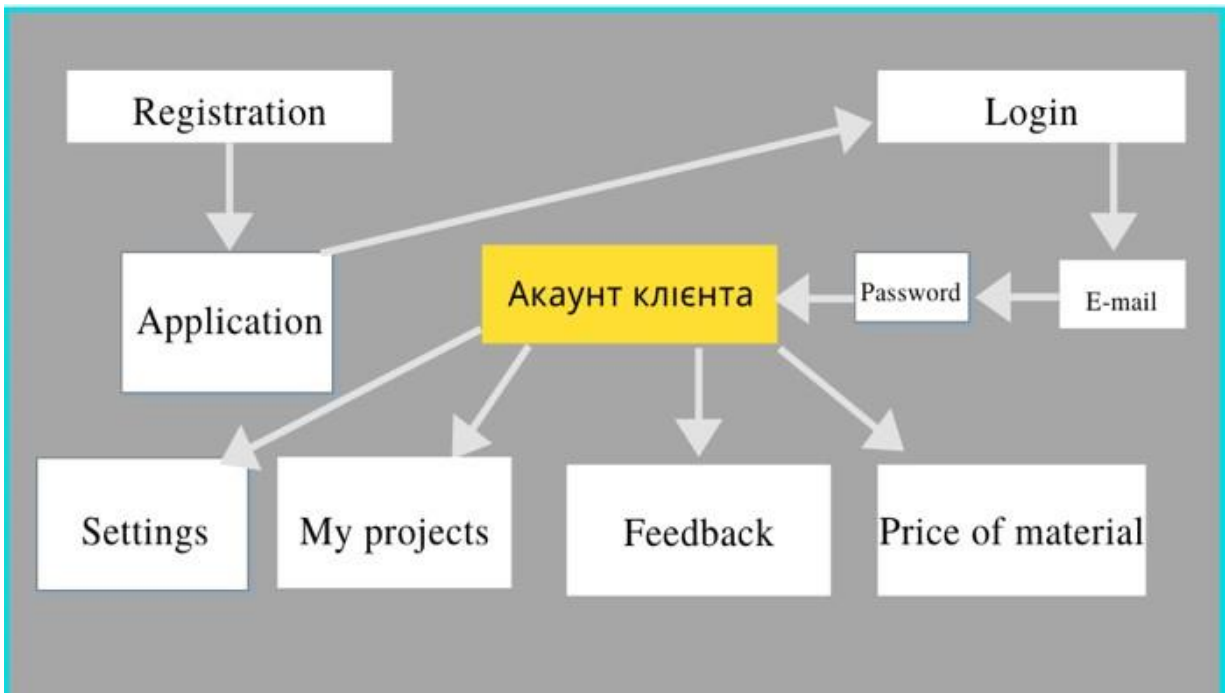


Рисунок 1.4 – Схема клієнтського аккаунту

Окрім облікового запису клієнта, передбачено також створити обліковий запис адміністратора. Основний процес створення кошторису відбувається в обліковому записі адміністратора. На рисунку 1.5 показано карту облікового запису адміністратора, яким може виступати менеджер

виробника будівельних матеріалів;

Рисунок 1.5 – Схема аккаунту адміністратора

2 ПРОЕКТУВАННЯ СЕРВІСНОГО ЗАСТОСУНКУ ПІДРАХУНКУ ВИТРАТ БУДІВЕЛЬНИХ МАТЕРІАЛІВ

2.1 Підбір компонентів реалізації централізованої інформаційної системи проектування витрат будівництва

Розробка централізованої інформаційної системи для створення будівельних кошторисів включає в себе інтерфейс розроблений з використанням HTML з CSS та JavaScript, фреймворкVue.js), та бекенд (PHP, з фреймворком Laravel) а також СУБД MySQL. Vue.js - це фреймворк JavaScript, який призначений для створення адаптивних інтерфейсів користкувача і досить складних односторінкових сайтів-додатків. Однією з його основних переваг є легка інтеграція з іншими проєктами та бібліотеками, включаючи бекенд-фреймворки. Vue.js базується на JavaScript і спрощує інтеграцію з іншими бібліотеками. Використання архітектурного патерну Model-View-ViewModel (MVVM) є типовим для побудови веб-інтерфейсів [29]. Цей фреймворк може значно зменшити навантаження на проєкт і прискорити завантаження веб-сторінок [31], а також ідеально підходить для створення рішень, що використовують зовнішні API для обробки даних. Узагальнюючи, Vue.js має гнучкий підхід до роботи з шаблонами, має простий синтаксис, швидко рендериться, має невеликі розміри, підтримує розширення для інструментів розробника та може бути поступово адаптований [32].

Для написання бекенду було використано мову PHP, фреймворк Laravel та базу даних MySQL. PHP є однією з найпопулярніших мов для розробки серверного програмного забезпечення; на основі цієї мови було створено безліч фреймворків. Фреймворки надають структуру для побудови додатків та включають різноманітні допоміжні функції, що сприяють у процесі розробки. Найпопулярнішими з них є Zend, Laravel, Symphy, CodeIgniter, Yii, CakePHP, Phalcon та FuelPHP [21]. Вони мають схожий функціонал, але відрізняються за підходами до розробки. Використання фреймворків дозволяє отримати безліч

переваг, таких як прискорення процесу розробки, спрощення обслуговування додатків, покращення рівня безпеки, автоматизація базових завдань у процесі розробки та раціональне використання баз даних [34].

Тож Laravel - це фреймворк на базі PHP з відкритим вихідним кодом. Його архітектурні патерни переважно ґрунтуються на Symfony [35]. Laravel займає провідне місце за рівнем використання фреймворку, з більш ніж удвічі більшою кількістю користувачів [33]. Після додавання інтерфейсу командного рядка (Artisan), підтримки СУБД та міграцій став користуватися попитом. Серед ключових переваг Laravel можна виділити:

1) Безпека на досить високому рівні;

2) Аутентифікація. Надаючи готовий функціоналсистем аутентифікації та авторизації;

3) Відкритий вихідний код. Що дозволяє модернізувати фреймворк під задані потреби;

4) Підтримує MVC та об'єктно-орієнтований підхід. Laravel є повністю сумісним з MVC і надає вбудовану підтримку принципів інверсії управління, є шаблоном і допомагає легко змінювати і модифікувати код;

5) Artisan. Звільняє розробників від необхідності створювати власні каркаси коду.

Фреймворк Laravel володіє найкращим об'єктно-реляційним відображенням порівняно з іншими фреймворками. Ця функціональність дозволяє взаємодіяти з об'єктами та зв'язками бази даних шляхом розумного синтаксису. Додатково, механізм шаблонів Blade стандартизує та уможливлює повторне використання шаблонів у різних частинах програми. Laravel також надає підтримку планування завдань, тестування та міграції баз даних, дозволяючи легко змінювати структури баз даних. Також важливими особливостями є можливість інтеграції з поштовими сервісами, швидкість розробки, багатомовна підтримка та об'єктно-орієнтовані бібліотеки, які легко взаємодіють між собою та дозволяють уникнути дублювання коду. Проте слід

зазначити, що англомовна офіційна документація може становити перешкоду для окремих користувачів, також виявлено проблеми з регулярним зворотним зв'язком між різними версіями фреймворку Laravel, а також не завжди логічно організовані каталоги та файли у порівнянні з іншими фреймворками, наприклад, Yii. Крім того, варто відзначити відсутність вбудованого конструктора сторінок для перегляду, редагування та видалення записів у базі даних порівняно з іншими фреймворками, такими як Ruby on Rails та Django. Ця обмеженість може бути розв'язана за допомогою сторонніх інструментів.

Щодо середовища розробки, PhpStorm є вибором, призначеним для цих завдань. Його функціонал включає редактори PHP, HTML, CSS і JavaScript, навігацію по коду, автоматичне завершення коду, підтримку утиліт та можливість встановлення плагінів. Крім того, спеціальні інструменти для роботи з Laravel, такі як Laravel IDE Helper і відповідний плагін, надають корисні підказки та рекомендації при написанні коду.

2.2 Розробка структури проекту та його функціональне моделювання

Моделювання структури та функцій веб-сайту для підрахунку кількості матеріалів представлене на діаграмі IDEF0 (рис. 2.1). Ця діаграма відображає призначення системи та її взаємодію із зовнішнім середовищем.. Вона складається з функціонального блоку, що відповідає за певний процес, зображений в центрі прямокутнику, та стрілок, з різних боків. Основна функція веб-додатку полягає у підрахунку витрат матеріалів. Об’єкти, що зумовлюють вплив на процес, позначаються стрілками.

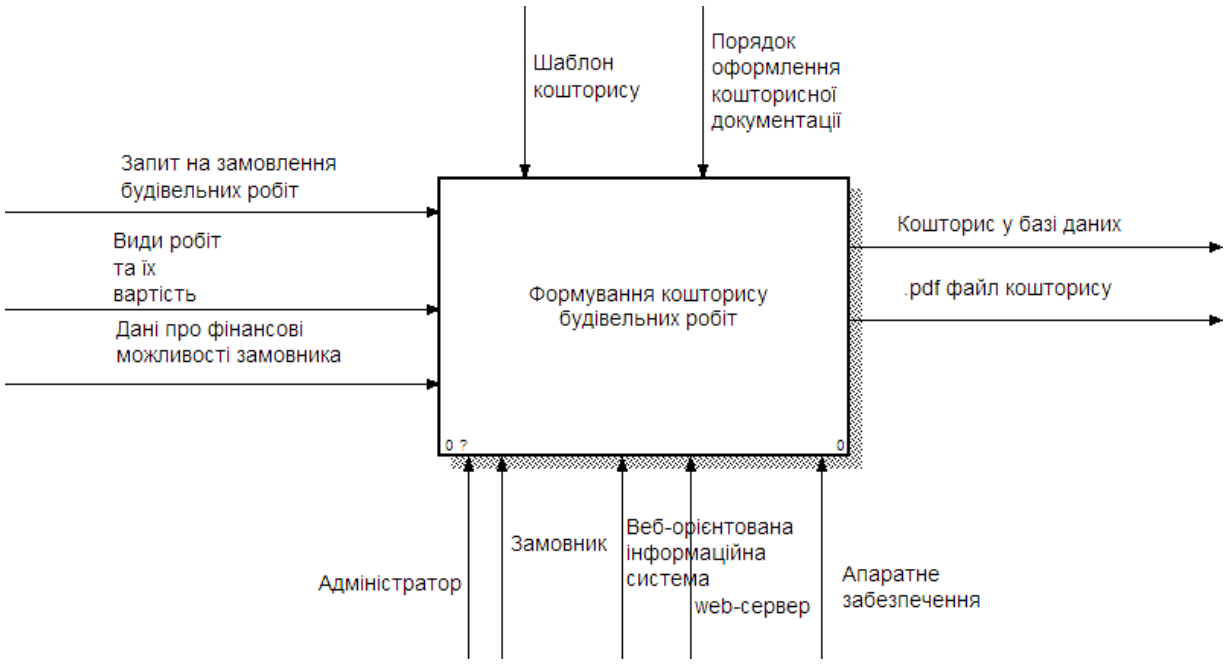


Рисунок 2.1 – Діаграма інформаційних потоків

Після опису основної функції в контекстній діаграмі відбувається її розкладання. На цьому етапі важливо визначити функції, що впливають на головну. Потім функції можуть бути розкладені на підфункції до досягнення необхідного рівня деталізації аналізованого процесу. Діаграми, що описують певну підфункцію конкретної системи, мають назву дочірніх. У розкладанні моделі основної функції веб-сайту визначено п'ять блоків:

- створення проекту для підрахунку;
- заповнення форми переліком матеріалів;
- перегляд замовлення;
- підтвердження підрахунку проекту;
- формування звітної документації.

Стрілки мають таке саме значення що й в контекстній діаграмі, але створюються і нові, що виходячи з першого блоку надходять в інший, це більш наглядно розкріває функціонування веб-сайту.

Отримана діаграма декомпозиції наведена на рис. 2.2.

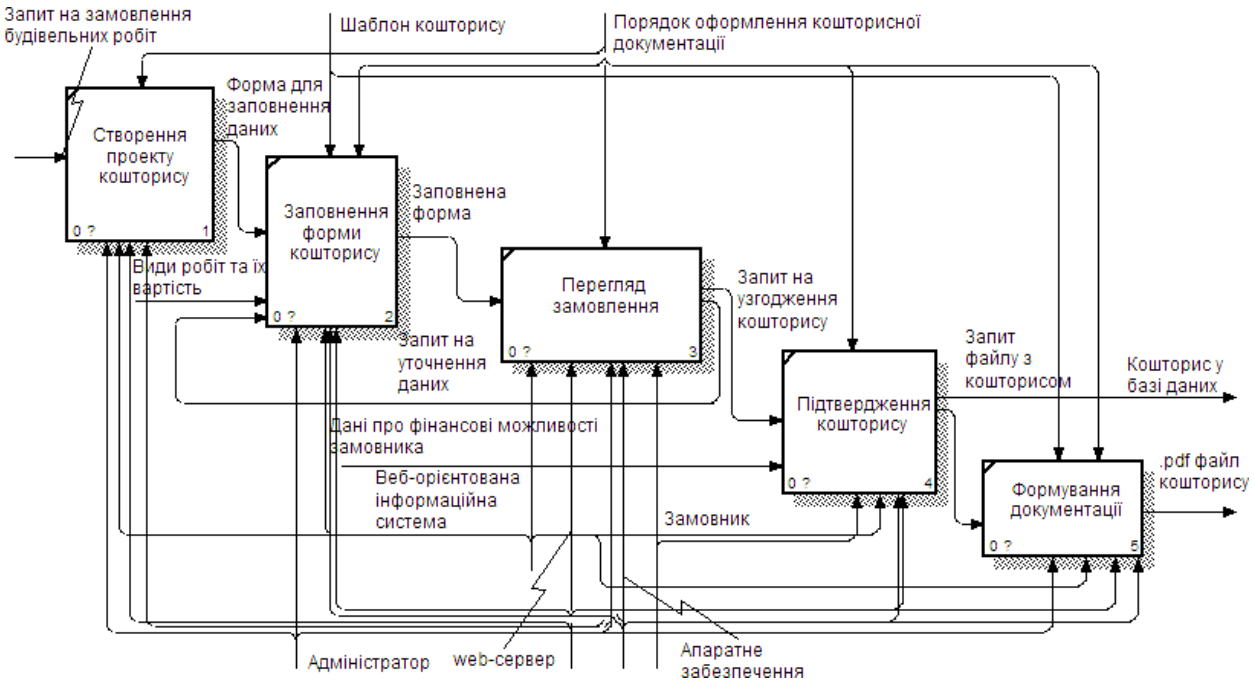


Рисунок 2.2 – Кінцева діаграма декомпозиції централізованої системи

2.3 Моделювання процесу використання централізованої інформаційної системи

Методологія використання (Use Case, від англ. використання) застосовується для формулювання необхідної поведінки системи, яку розробляють, без надання опису її конкретної реалізації. Це дозволяє забезпечити більш впевнені архітектурні рішення і перевірити систему під час розробки. Діаграми використання (Use Case) складають основний тип діаграм моделювання поведінки систем. Кожна з них демонструє набір можливих варіантів використання та акторів з їх взаємодіями. Діаграми використання мовою UML застосовуються для візуальної демонстрації поведінки системи з метою надання зрозумілості користувачам щодо способу взаємодії з конкретним елементом і розробникам – способу його реалізації. Взаємодію акторів із системою зображено на рис. 3.3: Адміністратор – користувач, якому надано доступ до адміністративної частини системи, має повноваження

управляти базами даних, обробляти заявки, формувати проекти підрахунку обсягів матеріалів. Клієнт – користувач, що може ознайомлюватися з інформацією, може пройти реєстрацію/авторизацію, створювати запити та залишати відгуки.

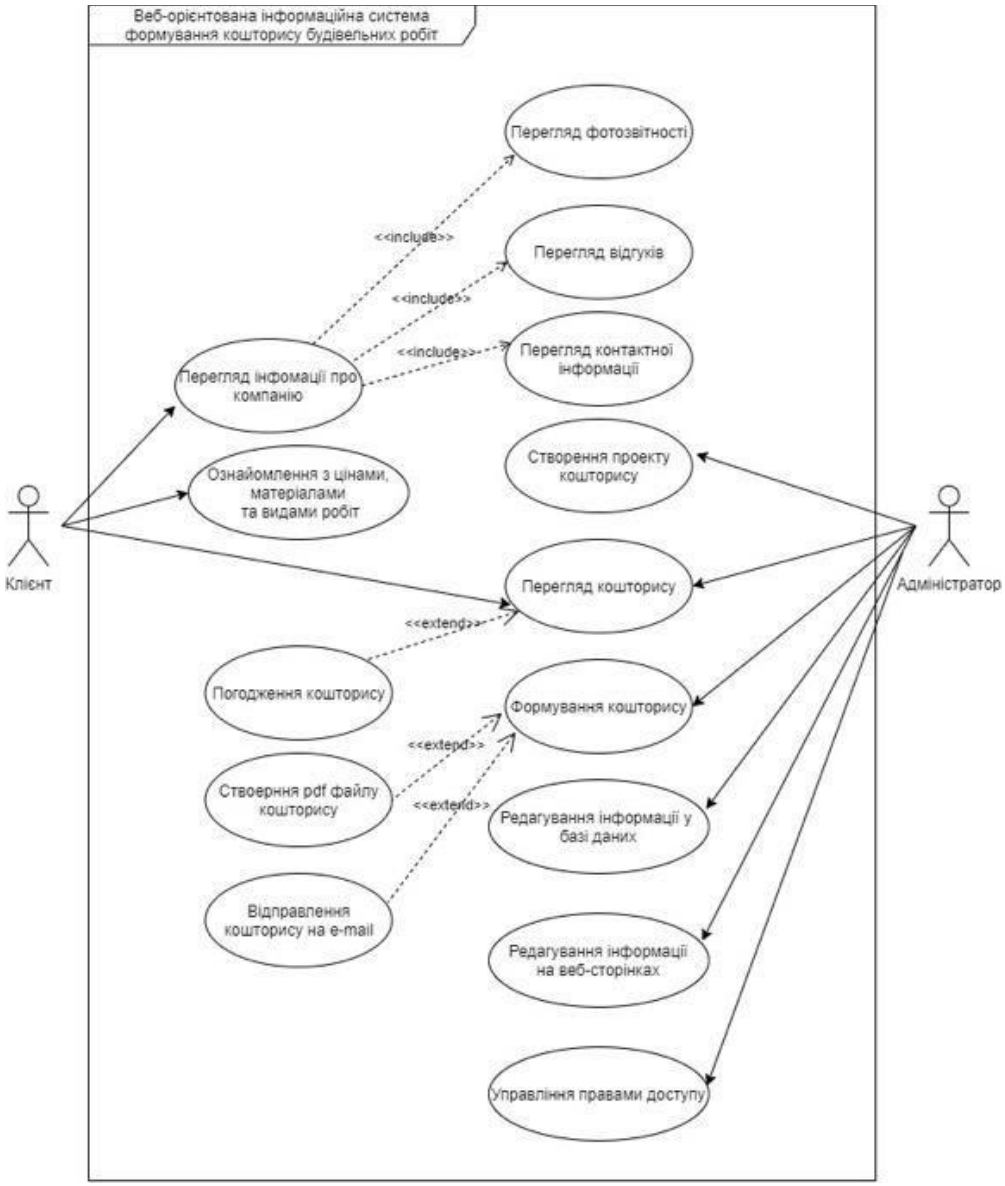


Рисунок 2.3 – Діаграма можливих варіантів використання функціоналу інформаційного застосунку

Можливі варіанти використання:

- Перегляд інформації компанії - надає можливість клієнту ознайомитися з діяльністю компанії.

- Перегляд фотозвітності дозволяє клієнтові побачити роботи, виконані компанією.

- Перегляд відгуків дозволяє клієнтам переглядати відгуки інших клієнтів щодо послуг компанії.

- Перегляд контактів компанії - дає можливість отримати контактну інформацію компанії.

- Ознайомлення з матеріалами та вартістю - дозволяє клієнту ознайомитися з цінами на матеріали (після авторизації).

- Створення проекту розрахунку - можливість створювати проекти кошторисів.

- Перегляд кошторису дозволяє переглядати кошторис робіт та матеріали.

- Погодження кошторису дає можливість погоджувати кошториси з клієнтом через веб-сайт.

- Розрахунок необхідних обсягів матеріалів - надає можливість створювати кошториси.

- Побудова pdf-файлу витрат на матеріали дозволяє створювати pdf файли калькуляції після збереження їх в базі даних.

- Відправка документації на e-mail дає можливість адміністраторові надсилати кошториси на e-mail клієнта з певного аккаунту одразу після їх формування.

- Зміну інформації в базі даних - дозволяє змінювати інформацію яка стосується опису видів робіт, цін на матеріали тощо.

- Зміну інформації за допомогою інтерфейсу застосунку - надає можливість редагувати інформацію на веб-застосунку.

- Зміна прав доступу - надає можливість реєстрації клієнта чи адміністратора; кожен обліковий запис має свою функціональну специфікацію; кожен пароль є зашифрованим.

2.4 Проектування структури бази даних

Обираємо реляційну базу даних для розробки сервісного застосунку. Адже вони спроектовані для швидкого маніпулювання великими обсягами записів. База даних представлена набором таблиць (сутностей). Кожна таблиця складається з рядків (кортежів) та колонок (атрибути). Рядки містять значення атрибутів та мають однакову структуру та розмір в межах всієї таблиці. Також в таблицях можуть застосовуватися обмеженні а між таблицями встановлено зв’язки – відносини.

Розглянемо більш детально таблиці бази даних, з яких вона складається табл. 2.1.

Таблиця 2.1 – Таблиці та їх функціональне призначення для системи

No з/п	**Назва**	**Функціональне призначення**
1	Constraction_projects	Зберігає інформацію про проект.
2	Constraction_materials	Зберігає дані по вартості матеріалів.
3	Constraction materials	відповідає за додавання запису в форму для розрахунку, (матеріал, вартість, кількість, сума).
8	Materials_for_estimate	Таблиця наповнення проекту матеріалами
9	Constraction units	Одиницы вимырювання
10	Constraction users	таблиця зареєстрованих користувачів.
11	Constraction comments	Інформація про товар.

Реляційна база даних надає можливість швидко порівняти інформацію, сприяючи формуванню нових таблиць з необхідною структурою та встановленням зв'язків з наявними таблицями. Це призводить до підвищення продуктивності бази даних [42]. Для кращого усвідомлення сутностей бази даних використовується ER-діаграма, що візуалізує ключові сутності, атрибути та їх взаємозв’язки. На рисунку 3.4 представлена структура бази даних проекту.

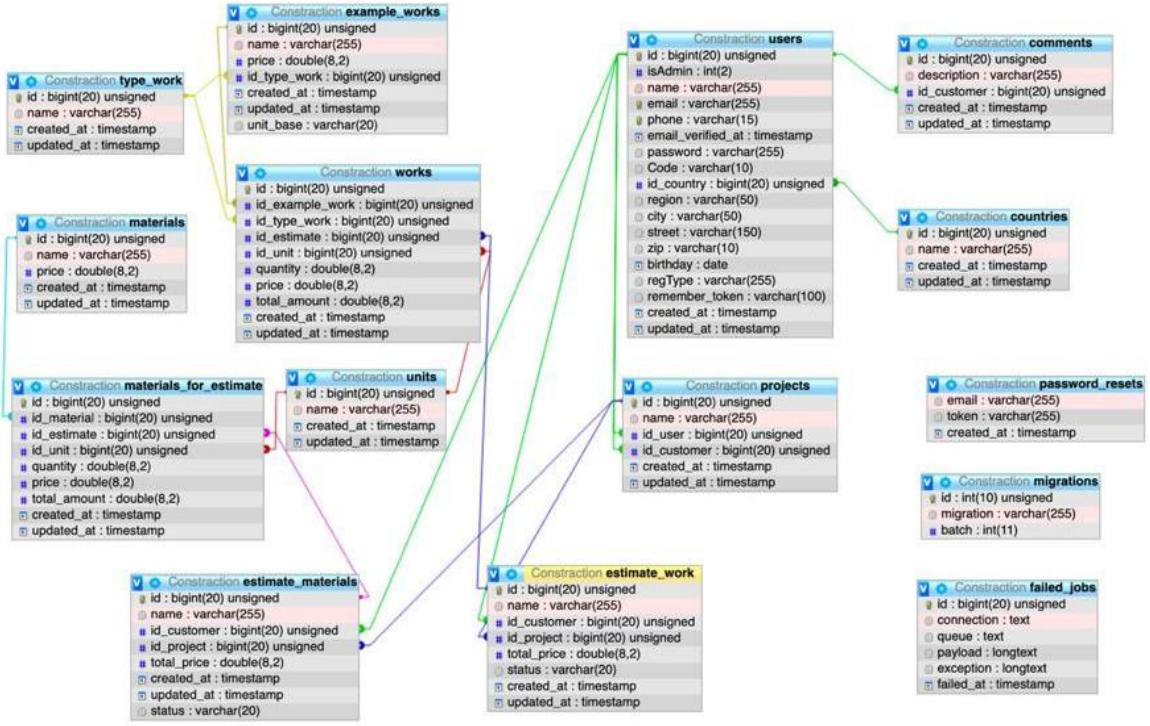


Рисунок 2.4 – Діаграма структури бази даних

2.5 Побудова архітектури сервісного застосунку

Архітектура веб-сайту грає важливу роль у зрозумінні системи користувачем, полегшує пошук інформації та забезпечує зручний перехід між сторінками. Прийнято рішення розробити веб-сайт за принципом трирівневої архітектури на базі моделі MVC (Model-View-Controller).

Така архітектура включає три складові компоненти:

1) клієнт (браузер, за допомогою якого користувач переглядає сторінки веб-інтерфейсу),
2) сервер (на якому розгорнуто веб-застосунок з використанням фреймворку Laravel);
3) сервер БД, що використовується системою (MySQL).

Цілісна реалізація веб-системи здійснена з використанням фреймворку Laravel для PHP.

Шаблон MVC, який є базою фреймворка, застосовується для відокремлення даних (Модель) від інтерфейсу користувача (Виглядд), щоб

зміни у вигляді користувача мінімізували вплив на роботу з даними, і зміни у моделі даних могли не впливати на інтерфейс користувача.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Створення компонентів централізованої інформаційної системи

Для реалізації функціоналу та виконання задач проекту розроблено базу даних, на рис. 3.1 наведено структуру бази даних "Constractions", яка складається з наступних таблиць:

- Users;
- materials;
- password_resets,
- type_work;
- countries;
- example_works;
- comments;
- estimate_work;
- projects;
- units;
- estimate_materials;
- materials_for_estimate;
- migrations;
- works.

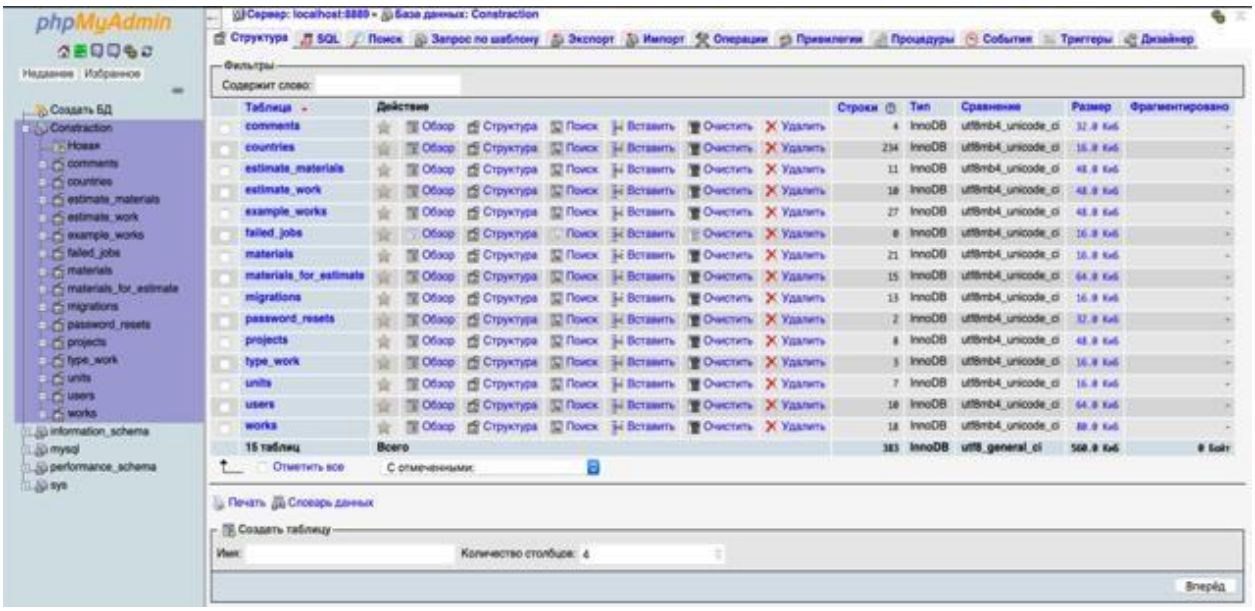


Рисунок 3.1 – Вікно phpMyAdmin зі створеними таблицями

Спершу біли створені моделі, що взаємодіють з базою даних (рис. 3.2), вони спрощують роботу з нею, та по суті є посиланнями на таблиці.

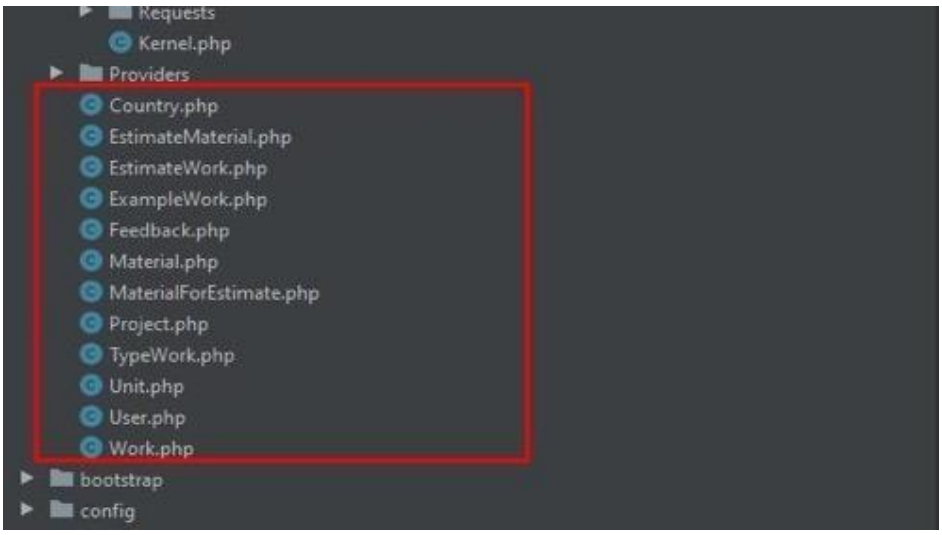


Рисунок 3.2 – Моделі централізованого веб-застосунку

В розробці було використовувалась технологія міграції. Вони необхідні для внутрішньої роботи Laravel. Міграція є механізмом контролю, який вирішує проблему неузгодженості баз даних. Треба лише виконати команду artisan (див. рис. 3.3).

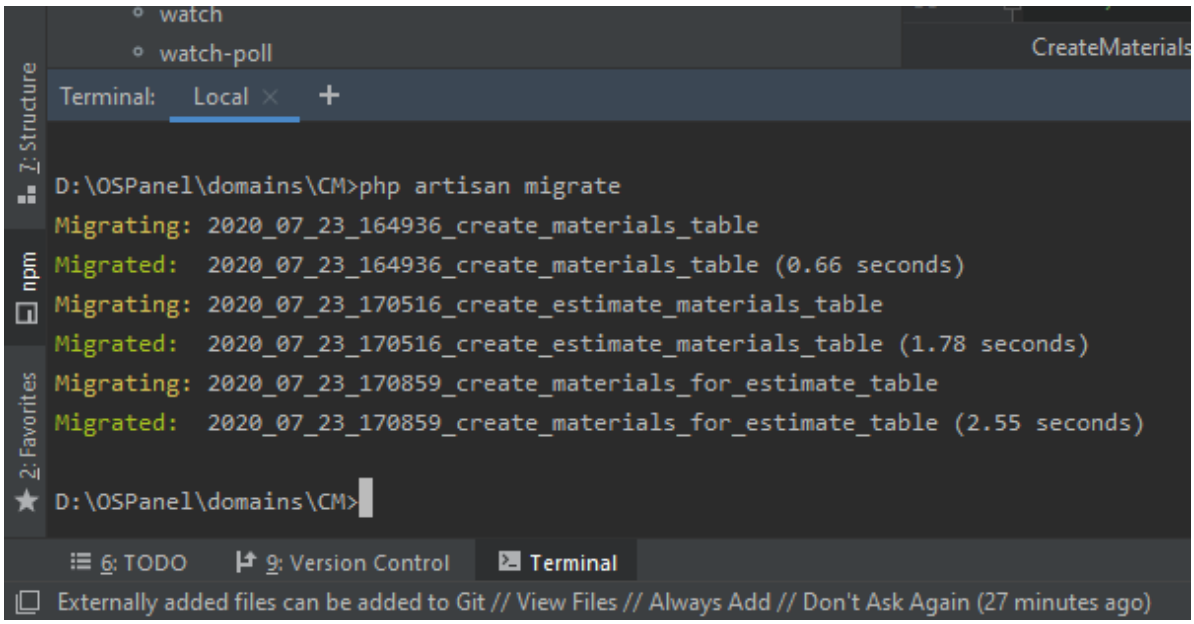


Рисунок 3.3 - Команда artisan для міграцій

Кожен процес міграції реалізується як клас і має два методи: up() - для запуску міграції, та down() - для відкату. На малюнку 3.4 зображено структуру файлів міграції.

Рисунок 3.4 – Структура міграції

Наступним важливим етапом є створення контролера (див. рис. 3.5). В контролерах описується звернення до бази даних з використанням моделі. Уся логічна частина взаємодії знаходиться саме в них (завантаження, видаження, оновлення, редагування та формування PDF звітності).

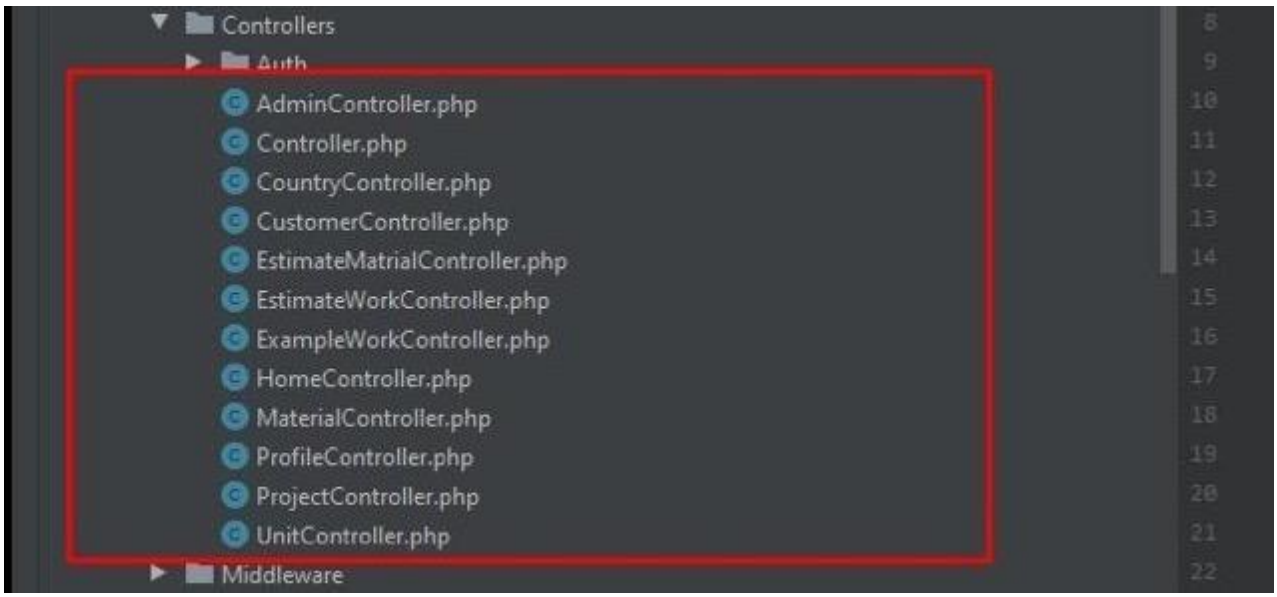


Рисунок 3.5 – Контролери

Наступний етап (після успішного встановлення з'єднання з базою даних) було відібрано всі необхідні дані, які потрібно передати на веб-сторінку. Для цього необхідно створити представлення (views) сторінок (див. рис. 3.6). Відображення сторінок здійснюється за допомогою комбінації html та php коду у веб-браузері.

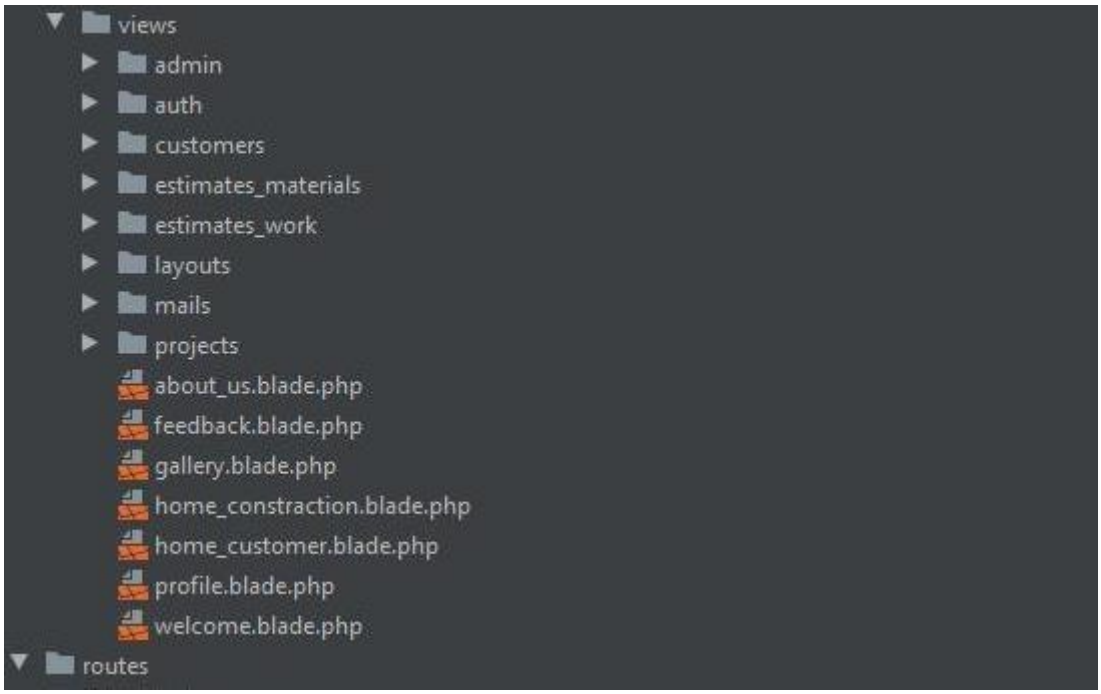


Рисунок 3.6 – Views

Також важливою складовою є маршрутизація. Маршрутизація - це система, що визначає шлях спрямування. Під час натискання кнопки

користувачем, , відбувається переміщення по сторінках. На малюнку 3.7 наведено, як налаштовуються маршрути.

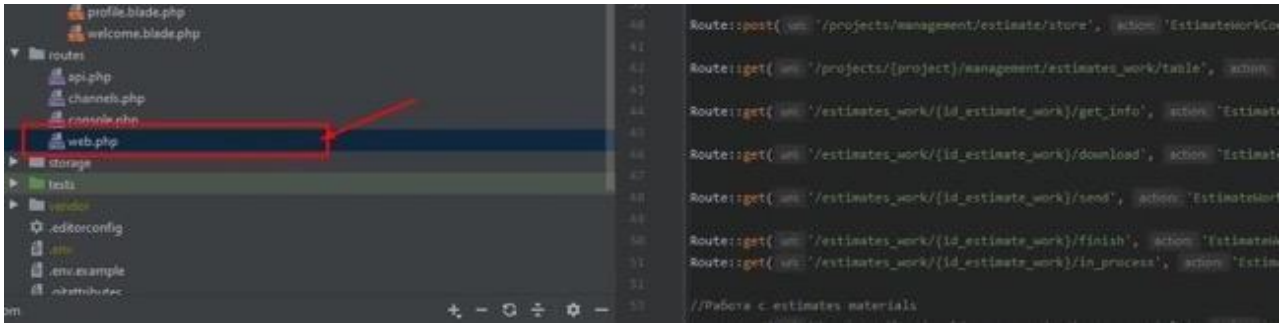


Рисунок 3.7 – Маршрутизація

Назва маршруту та його метод повинні бути прописані. Метод може бути: GET, POST. Роути вказують, до якого методу контролера вони відносяться. Для забезпечення однакової стилістики рисунків використовується програма Photoshop.

3.2. Тестування інформаційної системи

Для навігації до програми користувач вводить URL-адресу веб-сайту, що призводить до відкриття головної сторінки – Home (рис. 3.8). Ця сторінка призначена для презентації, у верхній частині розташоване меню.

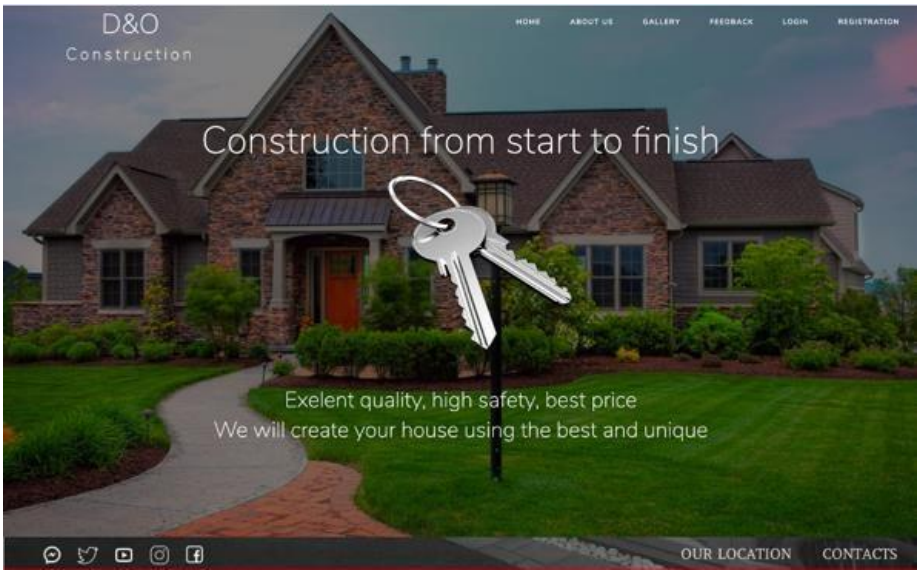


Рисунок 3.8 – Головна сторінка сайту

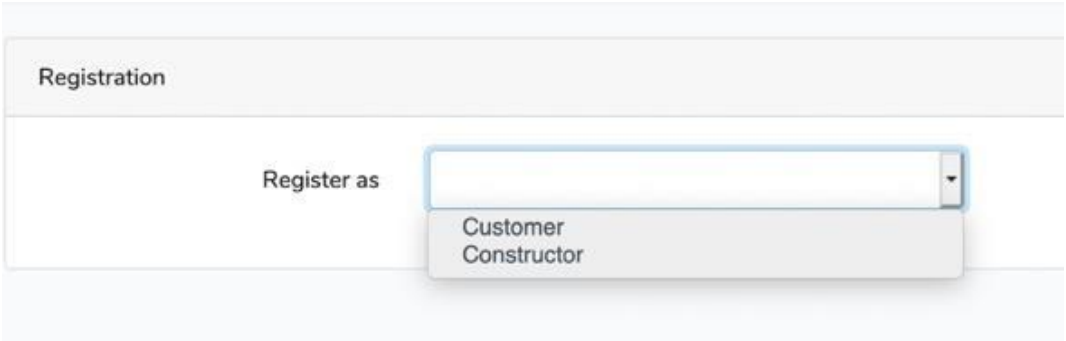


Рисунок 3.9 – Реєстрація по категоріям

Враховуючи той факт, що веб-сайт має авторизований доступ та розділення на функціональні частини - клієнтську та адміністративну, було реалізовано відповідні процедури реєстрації. При переході за посиланням "Registration", користувачу надається діалогове вікно вибору категорії,(customer/constructor) (рис. 3.9).

Для доступу до кабінету необхідно вказати логін та пароль (див. рис. 3.10). Щоб зберегти свої дані та увійти автоматично наступного разу, користувач може скористатися функцією "Запам'ятати мене" (Remember me). Якщо пароль був забутий, можна скористатися функцією "Забули пароль" (Fogot Your Password). Для цього необхідно вказати адресу електронної пошти, куди веб-застосунок відправить інструкцію по відновленню паролю.

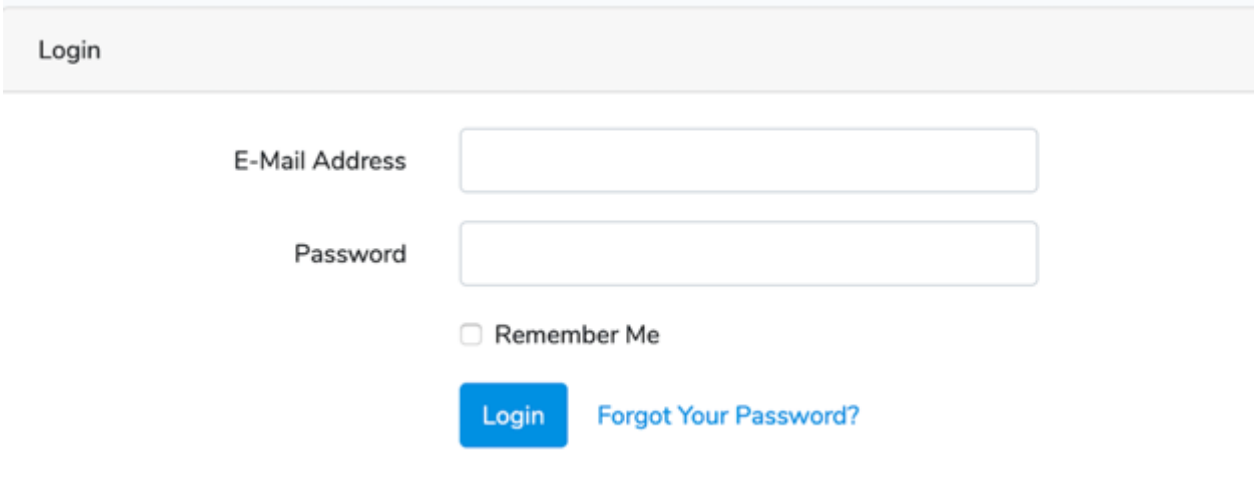


Рисунок 3.10 – Вхід до кабінету

При вході до кабінету адміністратора відобразиться робоча панель з таким меню (див. рис.. 3.11).

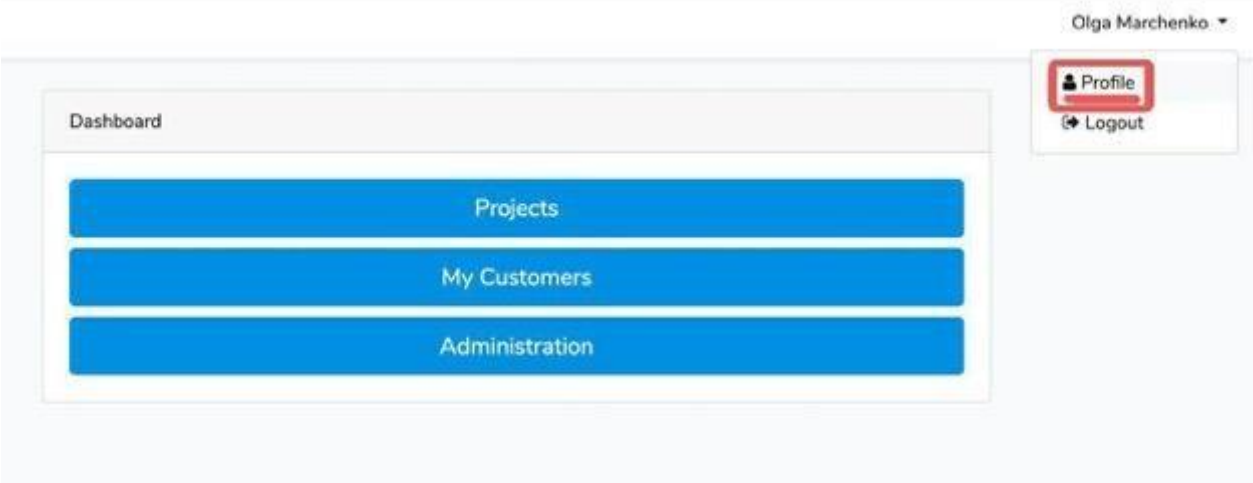


Рисунок 3.11 – Робоча панель аккаунту адміністратора

Після відкриття одного з проектів, система автоматично переходить на сторінку управління проектом (рис. 3.12). Тут адміністратор може створити кошториси на матеріали за допомогою функції "Оцінка матеріалів" та переглянути їх у розділі "Мої кошториси на матеріали".

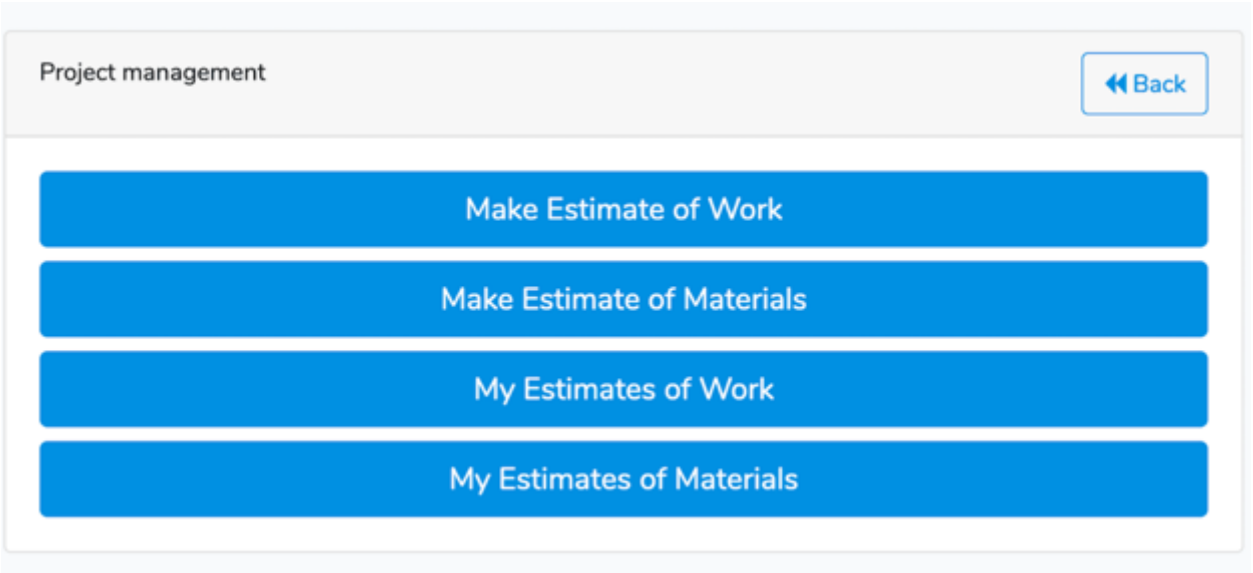


Рисунок 3.12 – Project management/управління проектом

Розглянемо меню "Оцінка матеріалів" (рис. 3.13). Форма заповнення

схожа на попередню. Усі поля обов'язкові до заповнення. Таблиця містить такі рубрики: Опис матеріалу, Одиниця виміру, Ціна, Кількість, Загальна сума. Ціна є фіксованою, проте у разі об'ємних замовлень може бути коригована індивідуально для кожного клієнта.

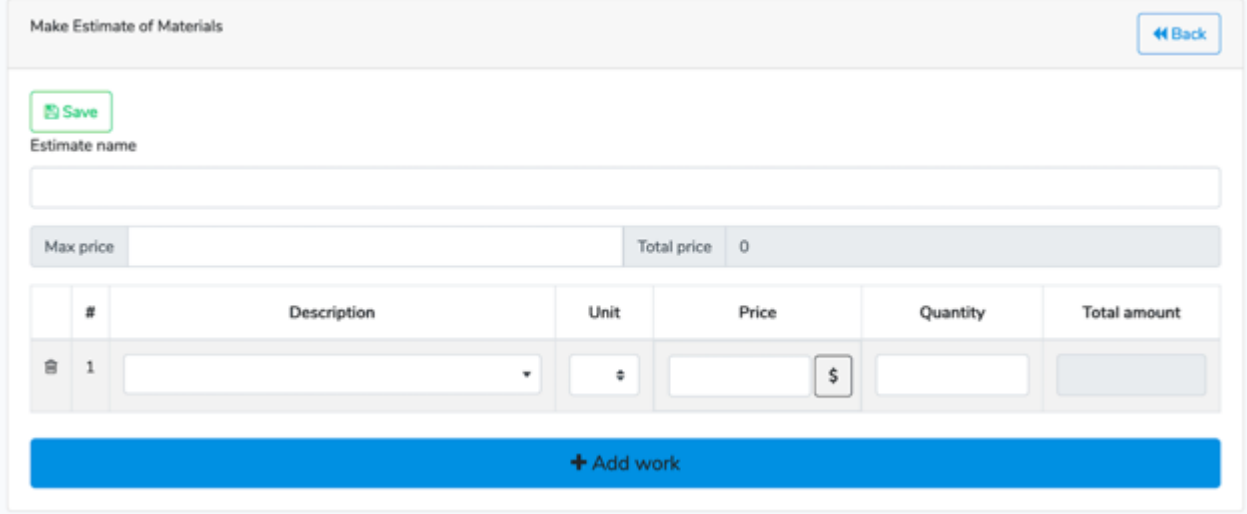


Рисунок 3.13 – Estimate of Materials

Якщо система показує на помилки або невідповідні значення, зберегти проект розрахунку неможливо (див. рис. 3.14).

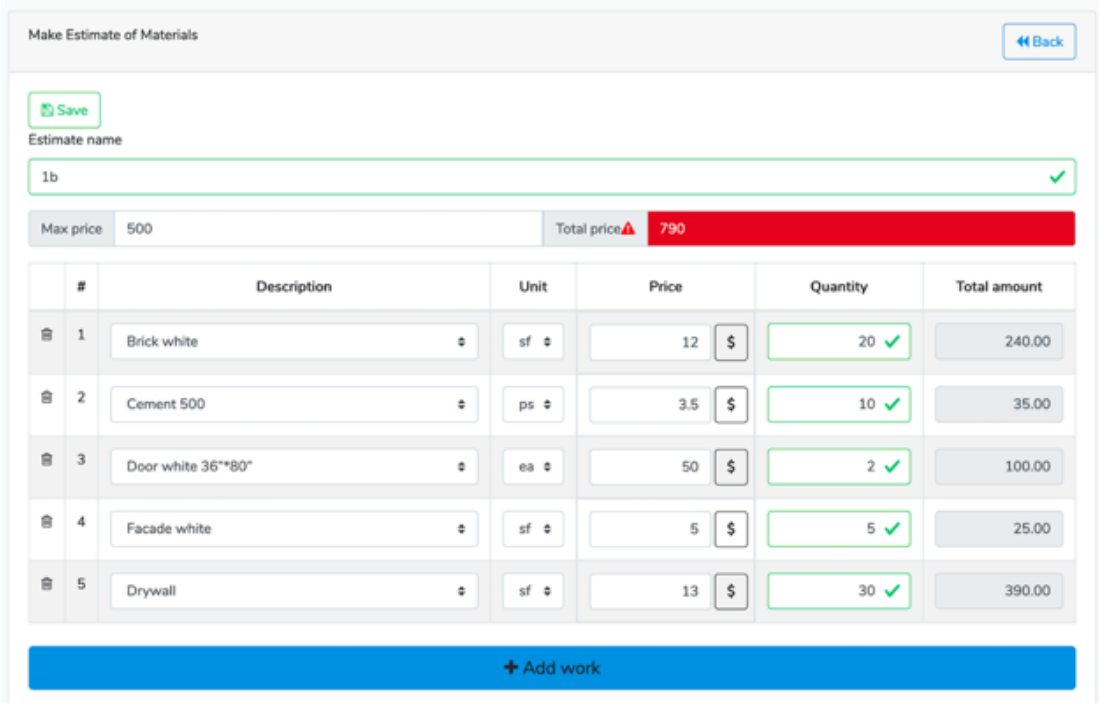
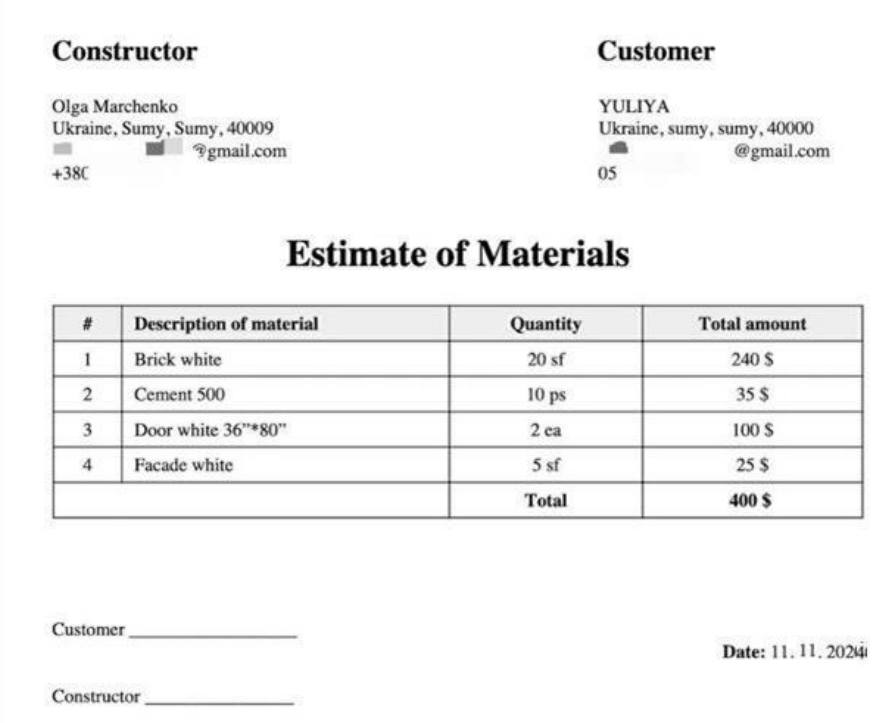


Рисунок 3.14 – Складання проекту розрахунку матеріалів, відображення помилок

Під час збереження інформації ми отримаємо файл у форматі PDF (рис. 3.15), що буде розташований у папці "Мій розрахунок матеріалів".



Constructor

Olga Marchenko
Ukraine, Sumy, Sumy, 40009
[redacted]gmail.com
+380

Customer

YULIYA
Ukraine, sumy, sumy, 40000
[redacted]@gmail.com
05

Estimate of Materials

#	Description of material	Quantity	Total amount
1	Brick white	20 sf	240 $
2	Cement 500	10 ps	35 $
3	Door white 36"*80"	2 ea	100 $
4	Facade white	5 sf	25 $
		Total	**400 $**

Customer _____

Date: 11. 11. 2024

Constructor _____

Рисунок 3.15 – Розрахунок витрат на матеріали

Розглянемо адміністративну частину кабінету (див. рис. 3.16). Меню адміністратора призначене для редагування інформації в базі даних внесення змін до бази даних і складається з категорій, перелік яких відповідає конкретна функціональна кнопка сторінки веб-застосунку, що зображена на рис. 3.16.

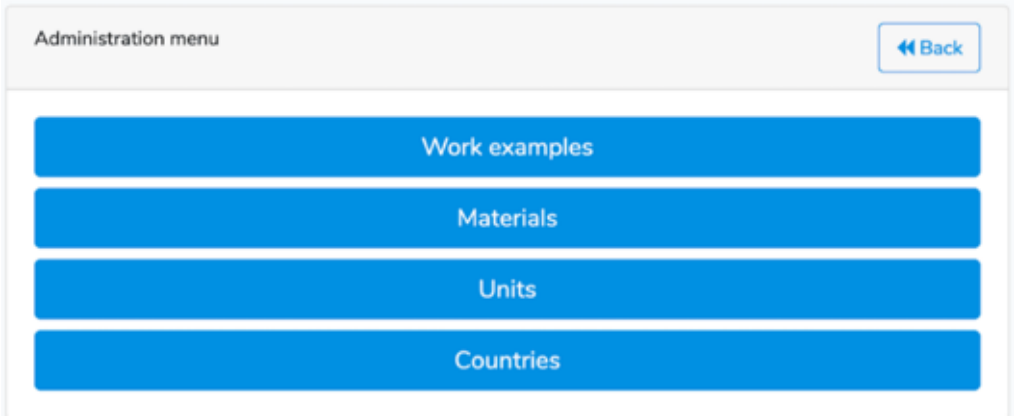


Рисунок 3.16 – Адміністративне меню

Сторінка "Materials" містить інформацію про доступні матеріали (див.

рис. 3.17). Адміністратор може переглянути базу даних щодо матеріалів, ознайомитися з їх ціновою політикою, скориставшись можливістю пошуку та фільтром, додавати нову інформацію про матеріали за допомогою кнопки "Додати", видаляти та вносити зміни до наявних записів (див. рис. 3.18).

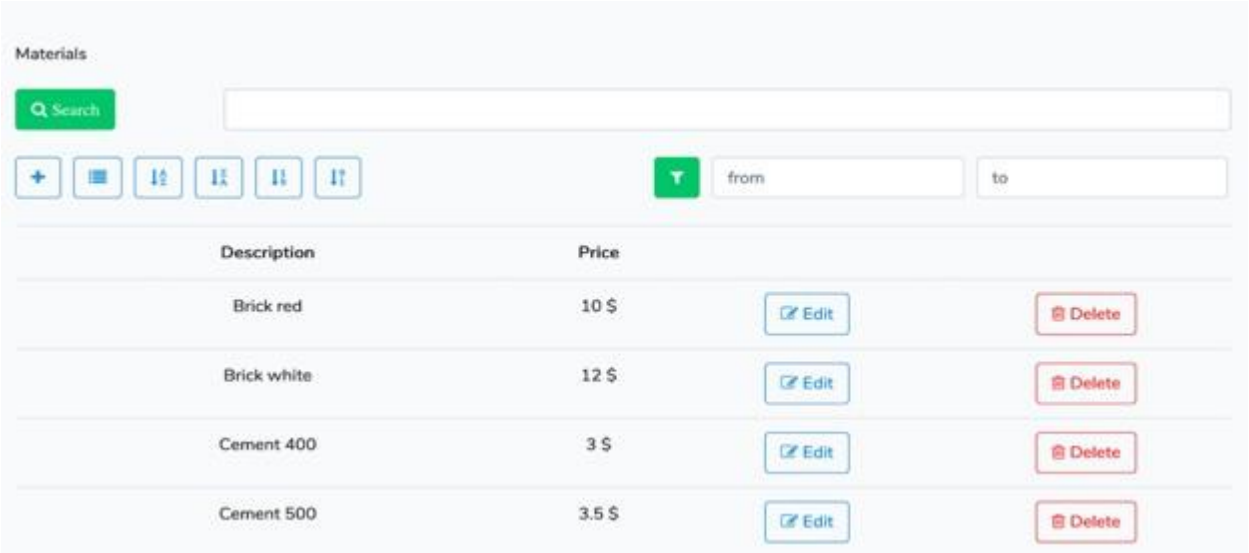


Рисунок 3.17 – Materials

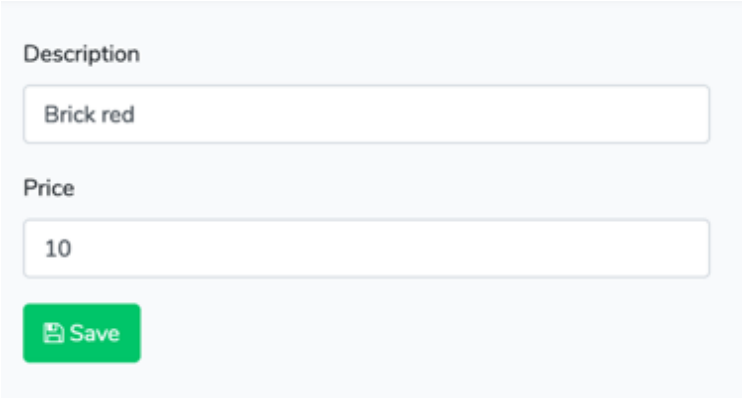


Рисунок 3.18 – Корегування матеріалів

Подивимося на кабінет клієнта. Робоче меню складається з: Мої проекти, Зворотній зв'язок та Ціна матеріалів (див. рис. 3.19).

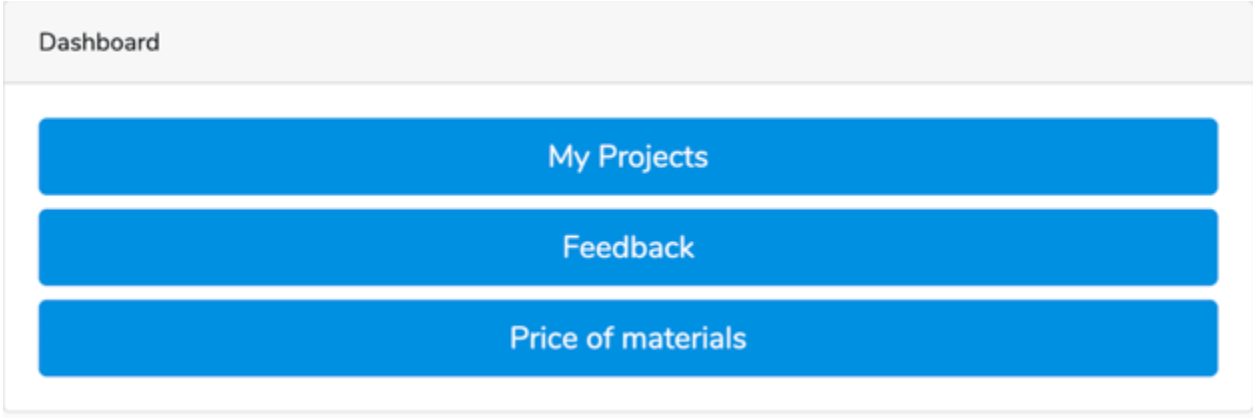


Рисунок 3.19 – Робоча панель аккаунту клієнта

Сторінка "Мої проекти" містить усі проекти клієнта (рис. 3.20). Кожному проекту призначені кошториси на матеріали (рис. 3.21) з відображенням дати створення та статусу. Для ініціації запуску проекту клієнт вказує його статус "в процесі". При натисканні на розрахунок веб-застосунок пропонує його перегляд у форматі PDF.

Рисунок 3.20 – Сторінка зі списком проектів користувача

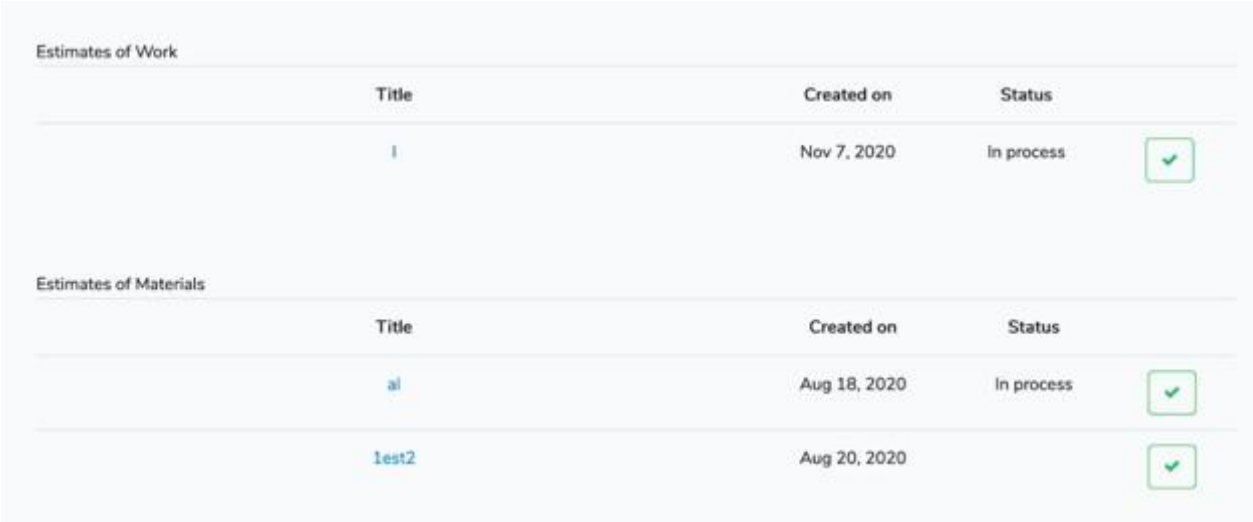


Рисунок 3.21 – Список розрахунків, що створені в одному проекті

Feedback – діалогова сторінка, де клієнт може залишити свої відгуки та коментарі (рис. 3.22)

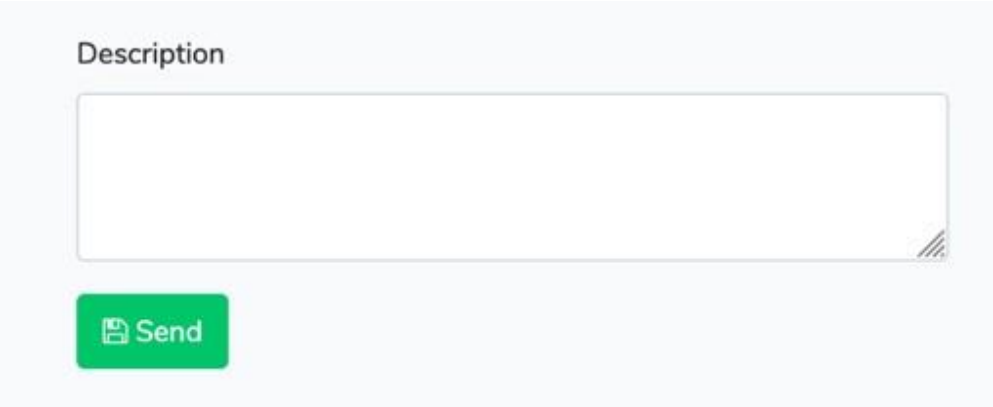


Рисунок 3.23 – Вікно з діалоговим вікном для відгуків та коментарів

Price of materials веб-сторінка з цінами на матеріали. Наявні функції пошуку і фільтрації (рис. 3.24).

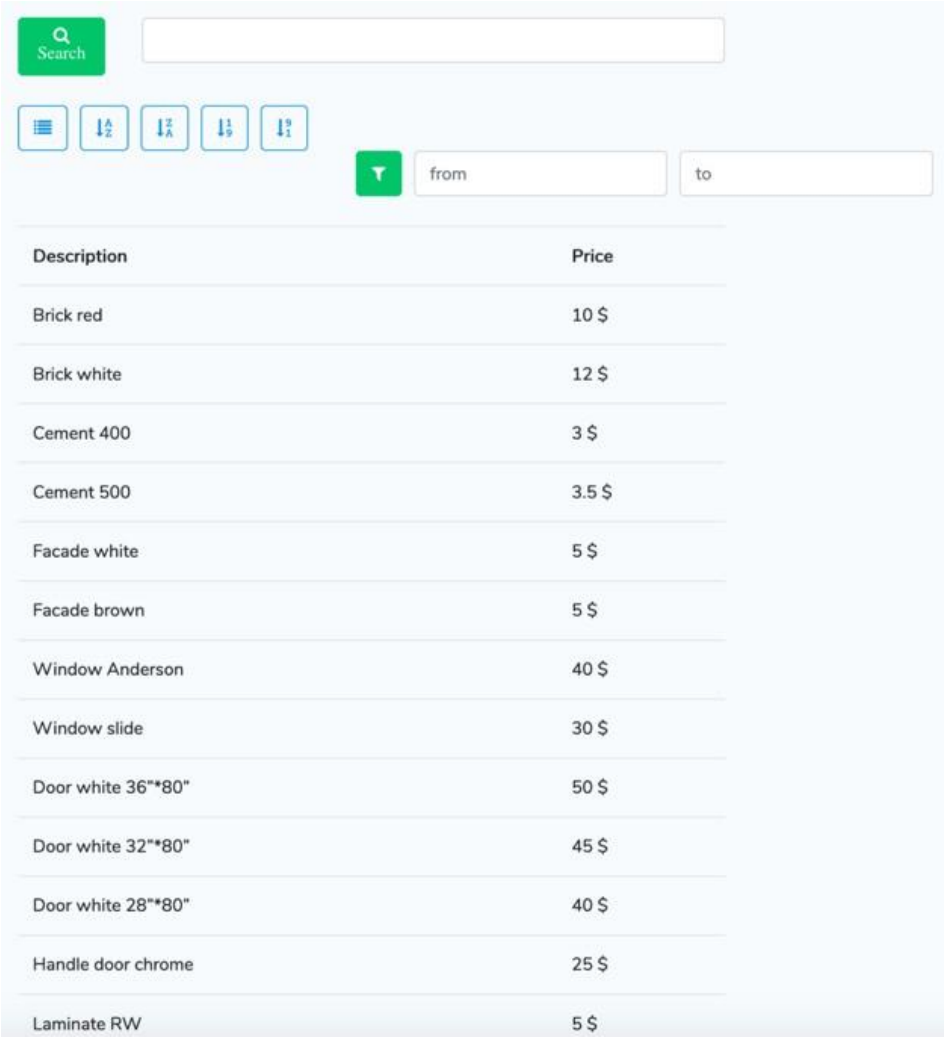


Рисунок 3.24 – Сторінка цін на матеріали

ВИСНОВКИ

Впродовж виконання кваліфікаційної роботи детально розглянуто та виконано аналіз програмних рішень, що застосовуються у будівельній сфері. Були визначені основні вимоги до розробки, сформульовано мету та поставлені завдання. Крім того, було проведено дослідження предметної області, оглянуто існуючі аналоги програмного забезпечення для складання кошторисів будівельних робіт, визначено функціональні вимоги, сплановано роботу та створено структуру проекту.

Розроблено функціональне моделювання структури, побудовано діаграми IDEF0, та її розгорнуту декомпозицію в також діаграму можливих варіантів використання централізованого сервісного веб-застосунку. Деякий обсяг часу був приділений огляду та аналізу наявних технологій у галузі розробки інформаційних систем що орієнтовані на web-технології

Для реалізації проекту було обрано мову програмування PHP, з розробкою проектів в інтегрованому середовищі PhpStorm, з використанням фреймворк Laravel для побудови функціоналу серверу. Для побудови та стилістики веб-сторінок - HTML з таблицею стилів CSS для визначення дизайну, мову JavaScript для анімації об'єктів на сторінках браузера а також СУБД MySQL для зберігання даних

Також спроектовано архітектуру розробки на базі технології MVC, створено базу даних та розроблено функціонал серверу Механізм міграції використовувався під час розробки з метою скорочення часу розробки.

В результаті кваліфікаційної роботи було розроблено готовий застосунок, за допомогою якого користувачі можуть здійснювати розрахунки необхідних обсягів будівельно-оздоблювальних матеріалів і їх вартостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Информационные технологии в строительстве, [Електронний ресурс] – режимдоступу: https://spravochnick.ru/informacionnye_tehnologii/informacionnye_tehnologii_ v_stroitelstve/
2. IT у сфері будівництва, [Електронний ресурс] – режим доступу: http://yurholding.com/news/117-t-u-sfer-budvnictva.html
3. Информационные системы строительных организаций: моделирование и оценка потребительского качества, [Електронний ресурс] – режим доступу: https://rsue.ru/avtoref/Kudinov.pdf
4. Информационные технологии в строительстве, Шумский А.Ю., [Електронний ресурс] – режим доступу: https://rep.bntu.by/bitstream/handle/data/36353/Informacionnye_tekhnologii_ v_ stroitelstve.pdf?sequence=1
5. Информационные технологии в архитектуре и строительстве, Г.В. Прохорский, [Електронний ресурс] – режим доступу: https://cdn1.ozone.ru/multimedia/1023960376.pdf
6. Інтелектуальна архітектурна сапр archicad, [Електронний ресурс] – режим доступу:
7. Vue.js. Getting Started, [Електронний ресурс] – режим доступу:https://012.vuejs.org/guide/
8. Vue.js: особенности, применение и отличия от других Javascript фреймворков, [Електронний ресурс] – режим доступу: https://stfalcon.com/ru/blog/post/vue-js-guide-to-tech
9. React или Angular или Vue.js — что выбрать?, [Електронний ресурс] – режим доступу: https://habr.com/ru/post/476312/
10. 6 Reasons Why Vue.js Is Important For Web Development, [Електронний ресурс] – режим доступу: https://medium.com/@shashank.nautiyal_20242/6- reasons-why-vue-js-is-important-for-web-development-b05ff3f8dee7

11. 10 Popular PHP frameworks in 2020, [Електронний ресурс] – режим доступу: https://raygun.com/blog/top-php-frameworks/
12. Why Laravel is the Best PHP Framework In 2020?, [Електронний ресурс] – режим доступу: https://www.esparkinfo.com/why-laravel-is-the-best-php-framework.html
13. 17 преимуществ использования Laravel в IT-индустрии, [Електронний ресурс] – режим доступу: https://wezom.com.ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii
14. 12 Reasons Why Laravel Is The Best PHP Framework?, [Електронний ресурс] – режим доступу: https://www.pontikis.net/blog/12-reasons-laravel-is-the-best- php-framework

ДОДАТОК А

Вихідний текст програми

Файл EstimateMatrialController.php

```
<?php

namespace

App\Http\Controllers;

use

App\EstimateMaterial;
use App\Material;
use
App\MaterialForEstimat
e; use App\Project;
use
App\TypeWork;
use App\Unit;
use
App\User;
use
App\Work;
use Barryvdh\DomPDF\Facade as
PDF; use
Illuminate\Http\Request;

use
App\Http\Requests\ProjectRe
quest; use
Illuminate\Support\Facades\
DB; use
Illuminate\Support\Facades\
Mail;

class EstimateMatrialController extends Controller
{

  /**

   * Display a listing of the resource.
   *

   * @return \Illuminate\Http\Response
   */

  public function index(Project $project)
```

```
  {
    $estimates_material =
    EstimateMaterial::where('id_project',$project->id)->get();
    return
                            view('estimates_materials.estimatesM
    aterialIndex',
compact('estimates_material')
      ,[ 'id_project' =>
      $project->id
    ]);
  }


  /**
   * Store a newly created resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @return
    \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirec
    tor
   */
  public function store(Request $request)
  {

    $estimate_material = new EstimateMaterial([ 'name' =>
      $request->get('Name'),
      'id_customer' => $request-
      >get('selectedOptionCustomer'),
      'id_project' => $request-
      >get('idProject'),
      'total_price' => $request-
      >get('total_price'), 'status' =>
      'New',
    ]);
    $estimate_material->save();

    $last_estimate =

   EstimateMaterial::latest()->first();

   foreach ($request->materials as $key

   => $value) {
     $material = new MaterialForEstimate([
       'id_material' => $value['id_material'],
       'id_estimate' => $last_estimate->id, 'id_unit'
```

```
        => $value['id_unit'], 'quantity' =>
        $value['quantity'], 'price' => $value['price'],
        'total_amount' => $value['total_amount'],
      ]);


      $material->save();
    }
     return redirect('/projects')->with('success', 'Stock has
     been added');

  }


  /**
   * Display the specified resource.
   *
   * @param $id
   * @return
     \Illuminate\Contracts\View\Factory|\Illuminate\View\View
   */
  public function show($id)
  {
    return
      view('estimates_materials.show',
      compact('id'), [
        'customers' =>    User::select('id','Name')-
        >where('regType',0)->get()-
>toArray(),
        'units' => Unit::select('id','Name')->get()->toArray(),
        'materials_all' =>
        Material::select('id','Name','price')->get()->toArray(),
      ]);
  }


  /**
   * Show the form for editing the specified resource.
   *
   * @param  int  $id
   * @return
     \Illuminate\Contracts\View\Factory|\Illuminate\View\View
   */
  public function edit($id)
  {
```

```
    return
      view('estimates_materials.edit',
      compact('id'), [

>first(),
'id_project'     => EstimateMaterial::select('id_project')-
>where('id',$id)-

'customers' =>    User::select('id','Name')->where('regType',0)-
>get()-
>toArray(),
        'units' => Unit::select('id','Name')->get()->toArray(),
        'materials_all' =>
        Material::select('id','Name','price')->get()->toArray(),
      ]);


  }


  /**
   * Update the specified resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function update(Request $request, EstimateMaterial
  $estimates_material)
  {


    $estimates_material->name = $request->Name;
    $estimates_material->id_customer = $request-
    >selectedOptionCustomer;
    $estimates_material->id_project = $request->idProject;
    $estimates_material->total_price = $request->total_price;
    $estimates_material->save();
    $array_materials         =
                             MaterialForEstimate::select('id')-
>where('id_estimate',$estimates_material->id)->get()->toArray();
    $stack = array();

    foreach ($request->materials as $key =>
      $value) { if($value['id'] == 0){
        $material = new MaterialForEstimate([
          'id_material' => $value['id_material'],
          'id_estimate' => $value['id_estimate'], 'id_unit'
```

```
        => $value['id_unit'], 'quantity' =>
        $value['quantity'],
        'price' => $value['price'],
        'total_amount' => $value['total_amount'],
      ]);
    }

    else{
      $material = MaterialForEstimate::where('id',
      $value['id'])->first();
      $material->id_material = $value['id_material'];
      $material->id_estimate = $value['id_estimate'];
      $material->quantity = $value['quantity'];
      $material->price = $value['price'];
      $material->total_amount = $value['total_amount'];
    }
    $material->save();


    array_push($stack, $value['id']);
  }


  foreach ($array_materials as $keya =>
                $valuea) {
  $result = array_search($valuea['id'],
                $stack);

     if ($result === false){
       $material = MaterialForEstimate::where('id',
       $valuea['id'])->first();
       $material->delete();
     }

   }
}


/**
 * Remove the specified resource from storage.
 *
 * @param Request $request
 * @param EstimateMaterial $estimates_material
 * @return \Illuminate\Http\Response
 * @throws \Exception
 */
public function destroy(Request $request, EstimateMaterial
```

```
  $estimates_material)
  {

    $estimates_material-
    >delete(); return
    redirect('/projects')
    ;

  }


  /**
   * Получение данных EstimateMaterial
   *
   * @param $id_estimate_material
   * @return array
   */
  public function getEstimateMaterialInfo($id_estimate_material)
  {

    $estimate_materials   =
                      EstimateMaterial::where('id',$id_estimate_
    material)-
>first();

    $materials  =
    MaterialForEstimate::where('id_estimate',$id_estimate_materia
    l)-
>get()->toArray();

    return ['estimate_materials' => $estimate_materials,
    'materials' => $materials];

  }


  public function download($id_estimate_materials)
  {

    if($id_estimate_materials == 0){
      $last_estimate = EstimateMaterial::latest()->first();
      $id_estimate_materials = $last_estimate->id;
    }

    $estimate_materials  =
                      EstimateMaterial::where('id',$id_estimate_m
    aterials)-
>first();
```

```
  $materials = MaterialForEstimate::select(
    'materials_for_estimate.quantity as
    quantity',
    'materials_for_estimate.total_amount as
    total_amount', 'materials.Name AS
    material_name',
    'units.Name AS unit_name'
  )

    ->leftJoin('materials', 'materials.id', '=',
    'materials_for_estimate.id_material')
    ->leftJoin('units', 'units.id', '=',
    'materials_for_estimate.id_unit')
    -
    >where('materials_for_estimate.id_estimate',$id_estimate_ma
    terials)
    ->get()
    ->toArray();

  $customer = User::select( 'countries.name as
    country', 'users.name',
    'users.email', 'users.phone', 'users.region',
    'users.city', 'users.street', 'users.zip'
  )

    ->leftJoin('countries', 'countries.id', '=',
    'users.id_country')
    ->where('users.id',$estimate_materials->id_customer)
    ->first();


  $constructor = User::select(
    'countries.name as country', 'users.name',
    'users.email', 'users.phone', 'users.region',
    'users.city', 'users.street', 'users.zip'
  )

    ->leftJoin('countries', 'countries.id', '=',
    'users.id_country')
    ->where('users.id',\Auth::id())
    ->first();


  $pdf =
    PDF::loadView('estimates_materials
    .pdf', [ 'estimate_materials' =>
    $estimate_materials, 'customer' =>
    $customer,
    'constructor' => $constructor, 'materials' =>
    $materials]);


  return $pdf->download('estimates_materials.pdf');
}
```

```
  public function sendEmail($id_estimate_materials){

    $estimate_materials  =
                    EstimateMaterial::where('id',$id_estimate_m
    aterials)-
>first();

    $materials = MaterialForEstimate::select(
      'materials_for_estimate.quantity as
      quantity',
      'materials_for_estimate.total_amount as
      total_amount', 'materials.Name AS
      material_name',
      'units.Name AS unit_name'
    )

      ->leftJoin('materials', 'materials.id', '=',
      'materials_for_estimate.id_material')
      ->leftJoin('units', 'units.id', '=',
      'materials_for_estimate.id_unit')
      -
      >where('materials_for_estimate.id_estimate',$id_estimate_ma
      terials)
      ->get()
      ->toArray();


    $customer = User::select( 'countries.name as country',
      'users.name',
      'users.email', 'users.phone', 'users.region',
      'users.city', 'users.street', 'users.zip'
    )

      ->leftJoin('countries', 'countries.id', '=',
      'users.id_country')
      ->where('users.id',$estimate_materials->id_customer)
      ->first();


    $constructor = User::select( 'countries.name as
      country', 'users.name',
      'users.email', 'users.phone', 'users.region',
      'users.city', 'users.street', 'users.zip'
    )

      ->leftJoin('countries', 'countries.id', '=',
      'users.id_country')
      ->where('users.id',\Auth::id())
      ->first();
    $pdf =
```

```
      PDF::loadView('estimates_materials
      .pdf', [ 'estimate_materials' =>
      $estimate_materials, 'customer' =>
      $customer,
      'constructor' =>
      $constructor,
      'materials' =>
      $materials]);

    $data["email"]=$customer->email;
    $data["client_name"]=$customer->name;

    try{
      Mail::send('mails.mail_estimate_material',
                                                          $
data, function($message)use($data,$pdf) {
        $message->to($data["email"], $data["client_name"])
          ->subject('Estimates')
          ->attachData($pdf->output(),
          "estimates_materials.pdf");
      });
    }catch(JWTException $exception){
      $this->serverstatuscode = "0";
      $this->serverstatusdes = $exception->getMessage();
    }

    return redirect('/home');
  }


  public function setFinishStatus($id_estimate_materials)
{


    $estimate_materials  =
                      EstimateMaterial::where('id',$id_estimate_m
    aterials)-
>first();

    $estimate_materials->status = 'Done';
    $estimate_materials->save();

    $user =\Auth::user();
    $projects =
      Project::select(
      'projects.id',
      'projects.name',
      'projects.create
      d_at',
      'users.name as customer_name'
```

```
    )
      ->leftJoin('users', 'users.id', '=',
      'projects.id_customer')
      ->where('id_user',$user->id)->get();
    return view('projects.index', compact('projects'));
  }

  public function setProcessStatus($id_estimate_materials){

    $estimate_materials  =
                    EstimateMaterial::where('id',$id_estimate_m
    aterials)-
>first();

    $estimate_materials->status = 'In process';
    $estimate_materials->save();

    $user =\Auth::user();

    return
      view('customers.my_proj
      ects', [
        'user_info' => User::where('id',$user->id)-
        >get()->first(), 'projects' =>
        Project::where('id_customer',$user->id)->get(),
      ]);

  }


}
```

Файл EstimateWorkController.php

```
<?php

namespace

App\Http\Controllers;

use App\EstimateWork;
use App\ExampleWork;
use
App\Project;
use
App\TypeWork
; use
```

```
App\Unit;
use
App\User;
use
App\Work;
use Barryvdh\DomPDF\Facade
as PDF; use
Illuminate\Support\Facades\U
RL; use
Illuminate\Http\Request;


use Illuminate\Support\Facades\Mail;

class EstimateWorkController extends Controller
{

  /**

   * Display a listing of the resource.
   *

   * @return \Illuminate\Http\Response
   */

  public function index(Project $project)
  {

    $estimates_work =
    EstimateWork::where('id_project',$project->id)->get();
    return view('projects.estimatesWorkIndex',
    compact('estimates_work'),[
      'id_project' => $project->id
    ]);

  }


  /**

   * Store a newly created resource in storage.
   *

   * @param  \Illuminate\Http\Request  $request
   * @return
    \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirec
    tor
   */

  public function store(Request $request)
  {


    $estimate_work = new
```

```
    EstimateWork([ 'name' =>
    $request->get('Name'),
    'id_customer' => $request-
    >get('selectedOptionCustomer'),
    'id_project' => $request-
    >get('idProject'),
    'total_price' => $request-
    >get('total_price'), 'status' =>
    'New',
  ]);
  $estimate_work->save();

  $last_estimate = EstimateWork::latest()->first(); foreach

 ($request->works as $key => $value) {
   $work = new Work([
     'id_example_work' =>
     $value['id_example_work'],
     'id_type_work' =>
     $value['id_type_work'], 'id_estimate'
     => $last_estimate->id,
      'id_unit' => $value['id_unit'], 'quantity' =>
      $value['quantity'], 'price' => $value['price'],
      'total_amount' => $value['total_amount'],
   ]);


   $work->save();
 }
  return redirect('/projects');

}


/**
* Display the specified resource.
*
* @param EstimateWork $estimateWork
* @return
  \Illuminate\Contracts\View\Factory|\Illuminate\View\View
*/
public function show($id)
{
  return view('estimates_work.show',
    compact('id'), [
      'customers' =>    User::select('id','Name')-
```

```
        >where('regType',0)->get()-
>toArray(),
        'typeWorks' => TypeWork::select('id','Name')->get()-
        >toArray(),
        'units' => Unit::select('id','Name')->get()->toArray(),
        'examples_all'

        =>
ExampleWork::select('example_works.id','example_works.Name','exam
ple_works.pri ce','units.id as id_unit_base')
          -
          >leftjoin('units','units.name','=','example_works.unit_
          base')
          ->get()->toArray(),

        'examples_exterior'

        =>
ExampleWork::select('example_works.id','example_works.Name','exa
mple_works.pri ce','units.id as id_unit_base')
          -
          >leftjoin('units','units.name','=','example_works.unit_
          base')
          ->where('id_type_work',1)
          ->get()->toArray(),

        'examples_interior'

        =>
ExampleWork::select('example_works.id','example_works.Name','exam
ple_works.pri ce','units.id as id_unit_base')
          -
          >leftjoin('units','units.name','=','example_works.unit_
          base')
          ->where('id_type_work',2)
          ->get()->toArray(),

        'examples_furniture'

        =>
ExampleWork::select('example_works.id','example_works.Name','exam
ple_works.pri ce','units.id as id_unit_base')
          -
          >leftjoin('units','units.name','=','example_works.unit_
          base')
          ->where('id_type_work',3)
          ->get()->toArray()
```

```
        ]);
    }


    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return
       \Illuminate\Contracts\View\Factory|\Illuminate\View\View
     */
    public function edit($id)
    {


      return view('estimates_work.edit',
        compact('id'), [
          'id_project' => EstimateWork::select('id_project')-
          >where('id',$id)->first(), 'customers'    =>
                      User::select('id','Name')-
          >where('regType',0)->get()-
>toArray(),
          'typeWorks' => TypeWork::select('id','Name')-
          >get()->toArray(), 'units' =>
          Unit::select('id','Name')->get()->toArray(),
          'examples_all'

          =>
ExampleWork::select('example_works.id','example_works.Name','exam
ple_works.pri ce','units.id as id_unit_base')
            -
            >leftjoin('units','units.name','=','example_works.unit_
            base')
            ->get()->toArray(),

          'examples_exterior'

          =>
ExampleWork::select('example_works.id','example_works.Name','exam
ple_works.pri ce','units.id as id_unit_base')
            -
            >leftjoin('units','units.name','=','example_works.unit_
            base')
            ->where('id_type_work',1)
            ->get()->toArray(),

          'examples_interior'
```

```
        =>
ExampleWork::select('example_works.id','example_works.Name','exam
ple_works.pri ce','units.id as id_unit_base')
          -
          >leftjoin('units','units.name','=','example_works.unit_
          base')
          ->where('id_type_work',2)
          ->get()->toArray(),

        'examples_furniture'

        =>
ExampleWork::select('example_works.id','example_works.Name','exa
mple_works.pri ce','units.id as id_unit_base')
          -
          >leftjoin('units','units.name','=','example_works.unit_
          base')
          ->where('id_type_work',3)
          ->get()->toArray()
      ]);


  }


  /**
   * Update the specified resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function update(Request $request, EstimateWork
  $estimates_work)
  {


    $estimates_work->name = $request->Name;
    $estimates_work->id_customer = $request-
    >selectedOptionCustomer;


    $estimates_work->id_project = $request->idProject;
    $estimates_work->total_price = $request->total_price;
    $estimates_work->save();

    $array_works=  Work::select('id')-
    >where('id_estimate',$estimates_work->id)-
```

```
>get()->toArray();
    $stack = array();

    foreach ($request->works as $key =>
      $value) { if($value['id'] == 0){
        $work = new Work([
          'id_example_work' =>
          $value['id_example_work'],
          'id_type_work' =>
          $value['id_type_work'], 'id_estimate'
          => $value['id_estimate'],
          'id_unit' =>
          $value['id_unit'],
          'quantity' =>
          $value['quantity'],
          'price' =>
          $value['price'],
          'total_amount' => $value['total_amount'],
        ]);
      }

      else{
        $work = Work::where('id', $value['id'])->first();
        $work->id_example_work = $value['id_example_work'];
        $work->id_estimate = $value['id_estimate'];
        $work->id_type_work = $value['id_type_work'];
        $work->quantity = $value['quantity'];
        $work->price = $value['price'];
        $work->total_amount = $value['total_amount'];
      }
      $work->save();

      array_push($stack, $value['id']);
    }

    foreach ($array_works as $keya =>
                            $valuea) {
$result = array_search($valuea['id'],
                              $stack);

      if ($result === false){
        $material = Work::where('id', $valuea['id'])->first();
        $material->delete();
      }
    }

  }
```

```
/**
 * Remove the specified resource from storage.
 *
 * @param Request $request
 * @param EstimateWork $estimates_work
 * @return \Illuminate\Http\Response
 * @throws \Exception
 */
public function destroy(Request $request, EstimateWork
$estimates_work)
{

  $estimates_work->delete(); return
  redirect('/projects');

}


/**
 * Получение данных EstimateWork
 *
 * @param $id_estimate_work
 * @return array
 */
public function getEstimateWorkInfo($id_estimate_work)
{

  $estimate_work =
  EstimateWork::where('id',$id_estimate_work)->first();

  $works = Work::where('id_estimate',$id_estimate_work)-

  >get()->toArray(); return ['estimate_work' =>

  $estimate_work, 'works' => $works];

}


public function download($id_estimate_work)
{

  if($id_estimate_work == 0){
    $last_estimate = EstimateWork::latest()->first();
    $id_estimate_work = $last_estimate->id;
```

```
    }
    $estimate_work =
    EstimateWork::where('id',$id_estimate_work)->first();

    $works = Work::select( 'works.quantity as
      quantity', 'works.total_amount as
      total_amount', 'example_works.Name AS
      work_name', 'units.Name AS unit_name'
    )
      ->leftJoin('example_works',      'example_works.id',
                                       '=',
'works.id_example_work')
      ->leftJoin('units', 'units.id', '=', 'works.id_unit')
      ->where('works.id_estimate',$id_estimate_work)
      ->get()
      ->toArray();

    $customer =
      User::select(
      'countries.name as
      country',
      'users.name',
      'users.email', 'users.phone', 'users.region',
      'users.city', 'users.street', 'users.zip'
    )
      ->leftJoin('countries', 'countries.id', '=',
      'users.id_country')
      ->where('users.id',$estimate_work->id_customer)
      ->first();


    $constructor = User::select(
      'countries.name as country',
      'users.name',
      'users.email', 'users.phone', 'users.region',
      'users.city', 'users.street', 'users.zip'
    )
      ->leftJoin('countries', 'countries.id', '=',
      'users.id_country')
      ->where('users.id',\Auth::id())
      ->first();

    $pdf =
      PDF::loadView('estimates_work.pd
      f', [ 'estimate_work' =>
      $estimate_work, 'customer' =>
      $customer,
```

```
        'constructor' => $constructor,


        'works' => $works]);


      return $pdf->download('estimates_work.pdf');
    }


    public function sendEmail($id_estimate_work){

      $estimate_work =
      EstimateWork::where('id',$id_estimate_work)->first();

      $works =
        Work::select(
        'works.quantity as
        quantity',
        'works.total_amount as total_amount',
        'example_works.Name AS work_name', 'units.Name
        AS unit_name'
      )
        ->leftJoin('example_works',      'example_works.id',
                                         '=',
'works.id_example_work')
        ->leftJoin('units', 'units.id', '=', 'works.id_unit')
        ->where('works.id_estimate',$id_estimate_work)
        ->get()
        ->toArray();

      $customer = User::select( 'countries.name as country',
        'users.name',
        'users.email', 'users.phone', 'users.region',
        'users.city', 'users.street', 'users.zip'
      )

        ->leftJoin('countries', 'countries.id', '=',
        'users.id_country')
        ->where('users.id',$estimate_work->id_customer)
        ->first();


      $constructor = User::select( 'countries.name as country',
        'users.name',
        'users.email', 'users.phone', 'users.region',
        'users.city', 'users.street',
        'users.zip'
      )

        ->leftJoin('countries', 'countries.id', '=',
        'users.id_country')
```

```
      ->where('users.id',\Auth::id())
      ->first();

    $pdf = PDF::loadView('estimates_work.pdf', [
      'estimate_work' => $estimate_work, 'customer' =>
      $customer,
      'constructor' =>
      $constructor, 'works'
      => $works]);

    $data["email"]=$customer->email;
    $data["client_name"]=$customer->name;

    try{
      Mail::send('mails.mail_estimate_work',
                                                          $
data, function($message)use($data,$pdf) {
        $message->to($data["email"], $data["client_name"])
          ->subject('Estimates')
          ->attachData($pdf->output(), "estimates_work.pdf");
    })
    ;

       }catch(JWTException
           $exception){
    $this->serverstatuscode
             = "0";
       $this->serverstatusdes = $exception-
                  >getMessage();
    }

    return redirect('/home');
  }


  public function setFinishStatus($id_estimate_work){

    $estimate_work =
    EstimateWork::where('id',$id_estimate_work)->first();

    $estimate_work->status = 'Done';
    $estimate_work->save();

    $user =\Auth::user();
    $projects = Project::select( 'projects.id', 'projects.name',
      'projects.created_at',
      'users.name as customer_name'
    )
      ->leftJoin('users', 'users.id', '=',
      'projects.id_customer')
```

```
        ->where('id_user',$user->id)->get();
      return view('projects.index', compact('projects'));
    }


    public function setProcessStatus($id_estimate_work){

      $estimate_work =
      EstimateWork::where('id',$id_estimate_work)->first();

      $estimate_work->status = 'In process';
      $estimate_work->save();

      $user =\Auth::user();

      return
        view('customers.my_proj
        ects', [

        ]);

    }
'user_info' => User::where('id',$user->id)->get()-
>first(), 'projects' =>
Project::where('id_customer',$user->id)->get(),

 }
```