МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

Факультет комп'ютерних наук
Кафедра Програмної інженерії

ЗВІТ
З ПЕРЕДАТЕСТАЦІЙНОЇ ПРАКТИКИ

Місце проходження практики:
Digital Cloud Technologies
у період з "*06*" *травня* по "*25*" *травня* 20*19* р.

Тема індивідуального завдання:
Веб-сервіс розрахунку фінансових витрат для ремонту квартир

Студент. гр.ПІ-15-4 _____ Шоков Д.Ф.
(підпис)

Керівник атестаційної роботи_____ Мельнікова Р.В.
(підпис)

Керівник практики _____ доц. Лещинська І.О.
(підпис)

Робота захищена з оцінкою _____ «__»_____2019

Комісія _____ Лещинська І.О.
(підпис)

_____ Трощило О.С.
(підпис)

_____ Черепанова Ю.Ю.
(підпис)

Харків, 2019

РЕФЕРАТ / ABSTRACT

Звіт з переддипломної практики, ___ стор., ___ рис., ___ табл., ___ джерел.

ASP .NET CORE, C#, VISUAL STUDIO 2019, АТЕСТАЦІЙНА РОБОТА, БАЗА ДАНИХ, РОЗРАХУНОК ФІНАНСІВ, ВЕБ-СЕРВІС, ВИТРАТИ НА РЕМОНТ КВАРТИР, ОСНОВНІ ЗАСОБИ.

Об'єкт розробки - Веб-сервіс розрахунку фінансових витрат для ремонту квартир.

Мета розробки – спростити менеджмент фінансів під час планування ремонту квартири.

Метод рішення - платформа .NET Core/Node Js, Microsoft Visual Studio 2019/Microsoft Visual Studio Code, мова C#/JS, Framework 4.5, ASP .NET Core WebApi/ReactJs+Redux.

В результаті розробки створений Web-сервіс, що дозволяє розраховувати подальшу кількість фінансових витрат згідно з планом ремонту квартири. Застосування розміщене на віддаленому сервері.

Explanatory note to the certification of the bachelor's degree, ___ pages, ___ rice, ___ tables, ___ sources.

ASP .NET CORE, C #, VISUAL STUDIO 2019, ATTESTATION WORK, DATA BASIS, FINANCIAL CALCULATION, WEB SERVICE, EXPENSES FOR REPAIR APARTMENTS, MAIN ACTIVITIES.

Object of development - Web service calculation of financial expenses for repair of apartments.

The purpose of the development is to simplify the management of finance when planning the flatness of the apartment.

The solution method is the .NET Core / Node Js, Microsoft Visual Studio 2019 / Microsoft Visual Studio Code, C # / JS, Framework 4.5, ASP .NET Core WebApi / ReactJs + Redux.

As a result of the development, the Web-service was created, which allows to count the further amount of financial expenses according to the plan of repair of the apartment. The application is hosted on a remote server.

ЗМІСТ

ВСТУП

Усе у нашому світі має якийсь термін дії, немає нічого вічного. І коли термін дії якоїсь речі доходить до кінця то люди починають займатися оновленням або ремонтом речі.

Однією з найважливіших речей людини є те місце, де вона живе, її домівля. Саме тому досить важно слідкувати за станом свого дому та речей які там знаходяться.

Ремонт квартир – дуже важливий, важкий, відповідальний та багатокоштовний етап у житті людини. Саме тому важливо чітко усвідомлювати що потрібно робити за будівельні роботи, які матеріали купувати, до якої фірми або компанії звертатися та, врешті-решт, скільки усе це може коштувати, бо саме фінансова складова має кінцевий та найголовніший вплив.

Частіше за все розрахунок коштів, які потрібно витратити на ремонт своєї квартири займає досить багато часу, бо включає у себе усі можливі вартості будівельних робіт, оплату будівельних матеріалів, та приблизний розрахунок усіх потрібних для цих робіт параметрів квартири, кімнати або декількох кімнат.

З усіма цими параметрами та показниками можна досить легко заплутатися, помилитися та розрахувати кошти неправильно, не кажучи вже про час, затрачений на розрахунки. Це може призвести до досить великих проблем, наприклад, якщо з’ясується, що за розрахунками коштів повинно було вистачити, але насправді вдалося зробити тільки половину робіт. І жити неможливо, і добудовувати немає коштів. Або навпаки – за розрахунками мало не вистачати коштів, але з’ясувалося, що це не так, а більш дешеві матеріали вже були закуплені, та менш якісні роботи проведені.

Отже, розрахунок затрат є досить важким процесом у якому легко помилитися, саме тому його потрібно автоматизувати та зробити зручним та швидким.

В даній роботі розглядається питання створення веб-сервісу для розрахунку фінансових витрат для ремонту квартир, який призваний спростити як менеджмент ведення ремонтних робіт, так і спростити, оптимізувати й зробити зручним розрахунок коштів, які потрібно витратити задля проведення ремонтних робіт.

ВІДОМОСТІ ПРО ПІДПРИЄМСТВО ПРАКТИКИ

Компанія Digital Cloud Technologies є експертом в розробці мобільних додатків для ОС Windows (UWP, Windows 10, Windows 10 Mobile).

Команда складається з експертів, визнаних компанією Microsoft (Microsoft Regional Director - MRD, Microsoft Most Valuable Professionals - MVP, Microsoft Certified Trainers - MCT, Microsoft Certified Professional Developers - MCPD, MicrosoftCertified IT Professionals-MCITP).

Компанія надає та підтримує такі сервіси:

а) РОЗРОБКА ПРОДУКТІВ:

1) Розробка додатків для універсальної платформи Windows;
2) Розробка під Windows 10, Windows 10 Mobile;
3) Веб розробка;
4) Розробка під Kinect;
5) Хмарні технології;

б) ПІДТРИМКА ПРОЕКТІВ:

1) Виконання проекту;
2) Аналітика;
3) Розробка інтернет речей;
4) Розробка систем автоматизації та моніторингу процесів в реальному часі;

Компанія була заснована у 2011 році Володимиром Лещинським таорієнтована на інноваційні технології лідерів ринку програмного забезпечення. Продукти DCT займають лідируючі позиції в Microsoft Windows Store.

1 АНАЛІЗ ПРОБЛЕМНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Ремонт квартири може долучати до себе досить об’ємний список пунктів як будівельних робіт, так і будівельних матеріалів. У свою чергу матеріали та роботи надають певні компанії, цінова політика яких також може відрізнятися, та навіть суттєво.

Також важливим чинником є якість будівельних робіт та матеріалів. Кожна компанія відрізняється якістю того чи іншого виду продукту або роботи, кожна має певні недоліки та сильні сторони, кожен матеріал має певну галузь використання.

Такими компаніями, наприклад, у нашому місті є компанії «Remont Stroy»,
«Servistroy», «Арт Альянс», «АртБудИнвест», «Харьков Строй» та ін., які відповідають за надання послуг будівельних робіт, та «Епіцентр», «БудМен», «Атлант», «СтройПартнер», «УкрБудМат» та ін., які відповідають за надання послуг щодо продажу будівельних матеріалів відповідно.

Усе це досить сильно ускладнює етап менеджменту процесу ремонту та розрахунку фінансових витрат.

Найчастіше такі процеси з точки зору клієнта відбуваються, як то кажуть «на папері», хоча в мережі Інтернет існують певні сервіси які спрощують ці задачі.

Більш того, працювати з сервісом не тільки легше, швидше, та й зручніше, ніж власноруч розраховувати усе це «на папері», це ще й також безпечніше відразу у двох сенсах:

- По-перше, це зберігаємість. Усі ми люди, саме тому можемо досить легко загубити розрахунки та планування свого ремонта, на які було потрачено досить немаленький час та зусилля, якщо воно знаходиться

у фізичному плані. На відміну від інтернет-сервісів, де усі свої розроблені етапи можливо зберегти, наприклад, у власному кабінеті, інформація про них буде зберігатися у базі даних, а може навіть і не одній, доступ до якої має тільки адміністрація сервісу, що насамперед передбачає гарантію надійності, набагато більшої, ніж у будь якому іншому плані.

- По-друге, це неможливість помилки при розрахунках. Знову ж таки, усі ми люди, та кожен може помилитися, розраховуючи кількість фінансів, які потрібно затратити на ремонт, можна зробити дуже великі погрішності, які внаслідок будуть обмежувати етапи ремонту, або, навіть і зовсім блокувати їх. Робота з грошима досить відповідальний момент, тому що розрахунок коштів на ремонт домівлі – це є той фундамент, звідки й сам ремонт безпосередньо починається. Інтернет веб-сервіси у цьому випадку це сервіси, які працюють, виконуючи певні формули, написані у коді, який виконується машинами, які не можуть оступитися, або «забути щось додати», вони є тим гарантом, що погрішності при розрахунках будуть мінімальні або взагалі відсутні. Саме тому вони є кращим вибором серед усіх можливих.

1.2 Виявлення та актуалізація рішень

У просторі мережі Інтернет можна знайти певні сервіси для розрахунку майбутніх фінансових витрат на ремонт домівлі. Усі вони мають певні сильні сторони та слабкі.

Прикладом таких сервісів можуть бути такі сервіси, як: «Smeta Online», «ArtPartner», «Meter», «Rabotniki.UA», «Timebuild», «Titanstroy» та ін.

Усі ці сервіси орієнтовані переважно на підрахунок кількості фінансових витрат на ремонт квартири.

Деякі з них мають дуже незрозумілий, поганий, не «User friendly» інтерфейс, що насамперед грає дуже важливу роль для користувача, бо це є точкою зв'язку між користувачем та, безпосередньо, сервісом. Поганий інтерфейс несе за собою питання про незрозумілість роботи з сервісом й у подальшому його використанні – неправильну роботу з ним як результат.

У багатьох з таких сервісів немає дуже простого, але надто важливого функціоналу, такого як збереження результату обробки розрахункових операцій.

Це є дуже великий мінус, який несе за собою ряд труднощів. Користувачеві банально потрібно буде кожен раз заповнювати одні й ті ж самі дані, щоб подивитися на результат розрахунку, або записувати кудись, щоб не забути, але це «вирішення» проблеми, може діяти тільки тоді, коли результат залишається тим самим. Але якщо потрібно буде, наприклад, додати нові дані про нову кімнату, то для того щоб підсумувати загальний результат, все одно потрібно буде вводити усі попередні дані кожен випадок.

Ще одним з головних недоліків усіх цих сервісів є дуже сильна прив'язка до будівельної компанії. Знаходячи такі сервіси у мережі Інтернет можна чітко побачити, що вони належать до якоїсь будівельної компанії, бо зазвичай, навіть знаходяться на їх ресурсах. Усі їх обчислення та розрахунки робіт та матеріалів основуються на ціновій політиці тільки тієї компанії, до якої цей сервіс і належить. Отже, одразу ж виникають питання та труднощі щодо розрахунку будівельних робіт та матеріалів які пропонують інші компанії та постачальники. Можливим «вирішенням» даної проблеми може бути використовування таких самих сервісів від інший компаній. Але це дуже сильно зменшує коло компаній, бо далеко не в усіх є такі сервіси. Іншою проблемою такого користування є те, що користувачеві потрібно буде з кожним використовуванням аналогічного сервісу вивчати та звикати до його інтерфейсу, що теж не є параметром зручності та економії часу.

Проблему обмеженості функціоналу сервісу можна віднести майже до усіх таких сервісів. Такі сервіси пропонують розрахунок фінансових витрат

для ремонту квартири, але майже нічого не пропонують додатково, наприклад, можливість мати не один проект – будівлю, для якої це можливо розрахувати, можливість менеджменту часу та дат робіт, дат закупівель та постачання будівельних матеріалів, та після підрахунку цілковитий список тих будівельних робіт та матеріалів, що входили для розрахунку та їх параметри.

Задача оптимізації та спрощення розробки плану ремонтних робіт та їх розрахунку була й є однією з найважливіших задач у цій галузі. Сучасні сервіси мають певні недоліки, питання про які повинні бути взяті до уваги, розглянуті, та вирішені на моментах постановки задачі, постановки вимог для ПП, проектування ПП та безпосередньо його імплементації.

1.3 Постановка задачі

Роблячи висновки на підставі результатів попередніх пунктів – «Аналіз предметної області»(1.1) та «Виявлення та актуалізація рішень»(1.2) дамо характеристику основної мети роботи.

Метою даної роботи є розробка інтернет веб-сервісу для розрахунку фінансових витрат для ремонту квартир, який надає функціонал для швидкого розрахунку вартості будівельних робіт та матеріалів, гарний користувацький інтерфейс, фунціонал для менеджменту часу та дат, та будівельних робіт з матеріалами, та виправляє основні недоліки попередніх аналогів сервісу, які буди наведені вище (див. п. 1.2 «Виявлення та актуалізація рішень» та п. «Аналіз предметної галузі»).

Перевагами пропонуємого сервісу є:

- Не орієнтованість не на ніяку компанію або підприємство;
- Каталог багатьох будівельних робіт та матеріалів різних компаній та постачальників;

- Більш оптимізований та швидкий розрахунок фінансових витрат для ремонту квартир;
- Менеджмент процесу ремонту квартир, що в свою чергу зробить кращим сферу менеджменту фінансів при ремонті квартир.

Створення веб-сервісу для розрахунку фінансових витрат для ремонту квартир досить нелегка задача яка містить у собі такі компоненти:

- Створення програмного рішення веб-додатку;
- Створення програмного рішення серверної частини;
- Створення відповідного сховища даних, у даному випадку бази даних;
- Ці компоненти взаємодіють за наступною схемою:

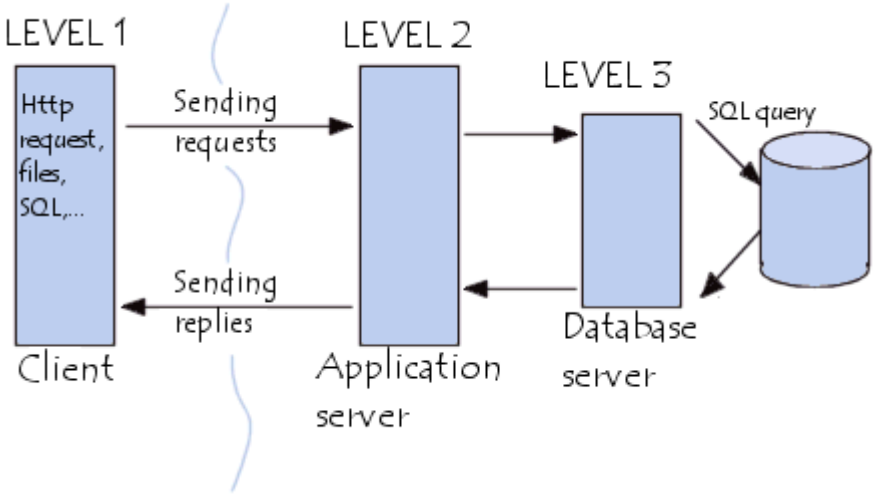

Рисунок 1.1 – Схема взаємодії компонентів

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Програмна система складається із трьох частин: клієнтський веб-додаток, сервер обробки клієнтських запитів та сховище даних у виді бази даних. Нижче будуть приведені функціональні, апаратні, програмні та операційні вимоги до описуваного програмного забезпечення.

2.1 Вимоги до функціональності клієнтського веб-додатку

Клієнтський веб-додаток має задовольняти наступним вимогам:

- Клієнтський веб-додаток повинен надавати доступ до регістрації нового користувача в системі через користувацький інтерфейс;
- Клієнтський веб-додаток повинен надавати доступ до авторизації вже існуючого користувача в системі через користувацький інтерфейс;
- Клієнтський веб-додаток повинен мати інтерактивний календар, на якому будуть відмічені заплановані події на різних етапах ремонтних робіт;
- Клієнтський веб-додаток повинен надавати доступ користувачеві до особистого кабінету;
- Клієнтський веб-додаток повинен надавати доступ до інтерфейсу управління проектами та їх схемами: перегляд, редагування, створення нових проектів;
- Клієнтський веб-додаток повинен мати можливість працювати з JSON-файлами та структурами;
- Клієнтський веб-додаток повинен надавати доступ до сторінки з усіма компаніями та відгуками про них;

- Клієнтський веб-додаток повинен надавати доступ до сторінки перегляду усіх даних про будівельні роботи та матеріали;
- Клієнтський веб-додаток повинен переходити на інші сторінки без фактичного оновлення сторінки браузеру;
- Клієнтський веб-додаток повинен дотримуватися політики REST;
- Клієнтський веб-додаток повинен надавати доступ користувачу до результатів розрахунків фінансових витрат на ремонтні роботи;
- Клієнтський веб-додаток повинен мати певну валідацію;

2.2 Вимоги до функціональності серверу обробки клієнтських запитів

Сервер для обробки клієнтських запитів має відповідати наступним вимогам:

а) Сервер повинен мати доступ до бази даних та певні інструменти для роботи з нею;

б) Сервер повинен вміти опрацьовувати запити з клієнта, а саме у форматі JSON;

в) Сервер повинен мати певну валідацію на запити, що приходять з клієнта;

г) Сервер повинен мати функціонал щодо реєстрації нового користувача у системі;

д) Сервер повинен мати функціонал щодо авторизації вже існуючого користувача у системі;

е) Сервер повинен мати CRUD функціонал щодо будівельних робіт, будівельних матеріалів, компаній, коментарів користувачів щодо компаній, проектів користувачів, схем проектів користувачів, контактів компаній;

ж) Сервер повинен мати багаторівневу архітектуру, у якій обов’язково повинні бути такі рівні, як:

1) Data Access Layer (DAL);
2) Service Layer або Business Logic Layer (BLL);
3) Representation Layer або WebApi – рівень представлення, де відбувається взаємодія з веб-клієнтом.

з) Сервер повинен дотримуватися політики REST;

2.3 Вимоги до функціональності бази даних

База даних повинна задовольняти наступні вимоги:

- База даних повинна базуватися на типі T-SQL та підтримувати запити у форматі SQL;
- База даних повинна бути створена за допомогою утиліти Entity Framework Core згідно з підходу Code First;

-

База даних повинна бути налаштована під роботу з утилітою Entity Framework Core;

2.4 Вимоги до апаратного забезпечення

Робоча станція, на якій має виконуватися описувана програмна система повинна задовольняти наступним мінімальним вимогам до апаратного забезпечення:

- Процесор – не менш ніж 2 ГГц;
- ОЗУ – не менш ніж 6 Гб;
- Місце на диску – не менш ніж 10 Гб;

2.5 Операційні вимоги

Робоча станція, на якій має виконуватися дана програмна система має працювати на одній із операційних систем з сімейства Windows, не нижче Windows 7, а також мати встановленим наступне програмне забезпечення:

- .NET Core 2.1 SDK
- ASP.NET Core 2.1 SDK
- Visual Studio 2017 та вище
- Visual Studio Code
- SQL Server Express
- Node JS SDK
- Npm модулі проекту веб-додатку

- Nu-get пакети серверної частини
- Компілятор мови C#
- Інтерпретатор мови JavaScript
- Фреймворк Entity Framework Core 2.1.*
- Microsoft Management Studio 2017
- Веб-браузер Google Chrome не пізніше версії 52, або Mozilla Firefox не пізніше версії 48.
- Фреймворк Redux

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML-проектування програмного забезпечення

Моделювання розроблюваної системи проводиться з використанням мови UML для побудови діаграм, що допоможуть відобразити функціональність та внутрішню структуру системи. UML – мова графічного опису для об’єктного моделювання в галузі розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю. При розробці системи було створено діаграму варіантів використання. Передбачуваний користувач описуваної програмної системи, тобто актор, повинен мати можливість реєструватися у системі як новий користувач, якщо він раніше ніколи цього не робив, авторизуватися у системі як вже існуючий користувач, якщо він вже колись реєструвався й існує у системі, створювати нові проекти та працювати з ними, додавати до них схеми кімнат або квартири в цілому, переглядати відгуки інших користувачів про будівельні компанії, переглядати списки будівельних робіт та матеріалів, а також мати можливість користуватися функцією розрахунку фінансових витрат вказаних ремонтних робіт.

На рисунку 3.1 відображена Use Case-діаграма для описуваної програмної системи, яка відображає можливі дії актора стосовно системи, реакції системи на дії актора, а також приховані дії, які виконує описувана програмна система за кадром, тобто ті дії, які користувач не має можливості побачити на власні очі.

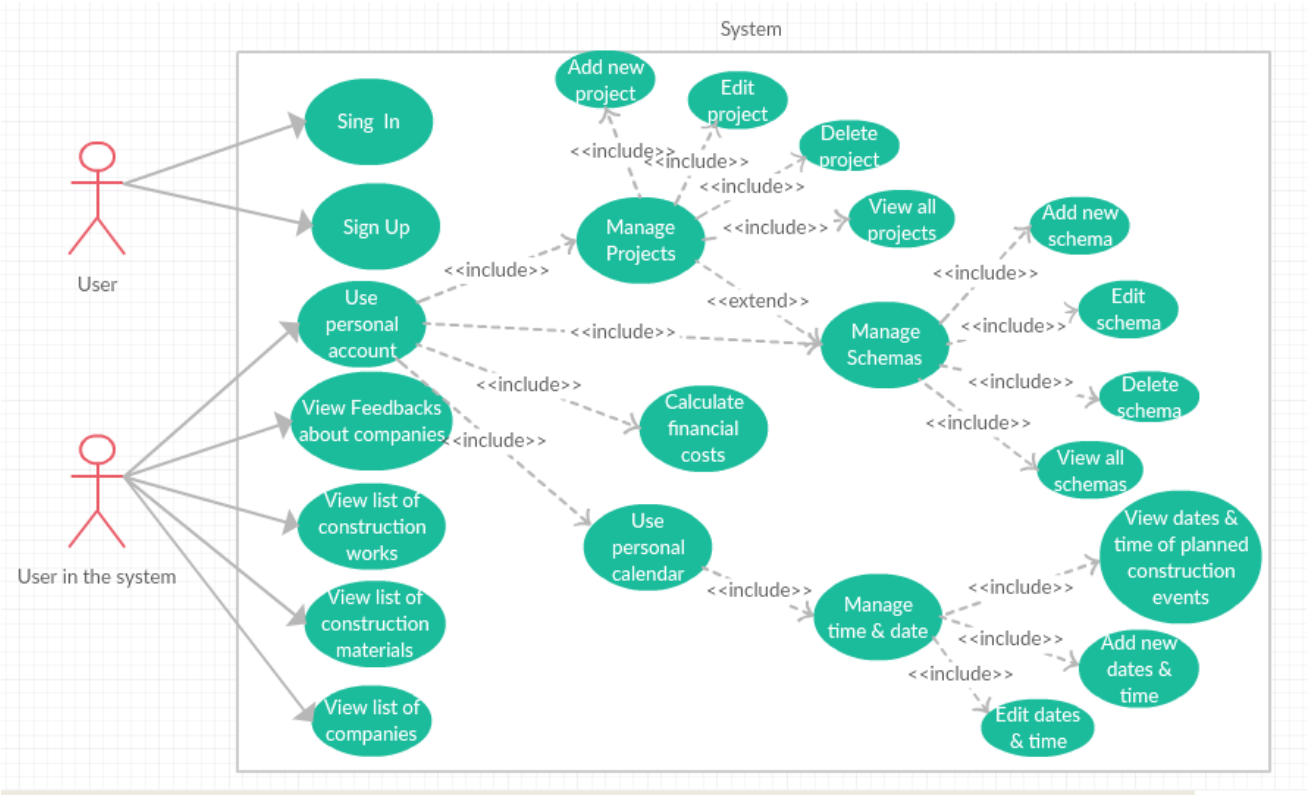

Рисунок 3.1.1 – Use Case-діаграма системи

Користувач, який не увійшов до системи має дуже обмежений функціонал, а саме можливість увійти до системи, якщо він вже був раніше зареєстрований у системі, та, безпосередньо, зареєструватися у системі, якщо цього не було ніколи зроблено раніше.

Далі користувач вважається як користувач у системі та у відмінності від неавторизованого користувача, він має набагато більш великий спектр дій. Таким чином він має доступ та може керувати особистим кабінетом, який насамперед долучає до себе такий функціонал як управління власними проектами, до чого відноситься створення нового проекту, редагування та видалення вже існуючого, та перегляд усіх існуючих проектів користувача.

Проекти насамперед мають певні схеми, які користувач також може редагувати, видаляти, додавати нові та переглядати усі існуючі.

Функціонал власного кабінету також дозволяє користувачу використовувати власний календар. У власному календарі користувач може вести управлінням дати та часом різноманітних подій етапу ремонтних робіт, будь то визначення та зберігання дати й часу ремонтно-будівельних робіт, або постачання будівельного матеріалу. У власному календарі користувач може додавати нову подію на календар та визначати їй дату й час, редагувати та видаляти вже існуючи подію та переглядати усі заплановані події, які в нього є.

Відходячи від особистого кабінету, користувач також може переглядати різноманітні будівельні матеріали та будівельні роботи різних компаній з їхньою ціновою політикою, додавати собі ці матеріали або роботи до конкретної схеми або декількох схем, проглядати будівельні компанії та інформацію про них, проглядати відгуки про різноманітні компанії від інших користувачів у системі та додавати свої відгуки.

3.2 Проектування архітектури програмного забезпечення

3.2.1 Проектування серверної сторони програмного забезпечення

Для розробки серверної сторони програмного забезпечення було вирішено використовувати мову програмування C# та платформу .NET Core та фреймворк ASP.NET Core.

C # - це витончена об'єктно-орієнтована мова зі строгою типізацією, що дозволяє розробникам створювати різні безпечні і надійні додатки, що працюють на платформі .NET Framework та .NET Core. C# можна використовувати для створення клієнтських додатків Windows, XML-веб-служб, розподілених компонентів, додатків клієнт-сервер, додатків баз даних ті ін. Visual C# надає розвинений редактор коду, зручні конструктори користувальницького інтерфейсу, інтегрований відладчик і багато інших

засобів, які спрощують розробку додатків на мові C# для платформи .NET Framework та .NET Core.

.NET Core - це універсальна платформа розробки з відкритим кодом, яку підтримує корпорація Майкрософт і співтовариство .NET на сайті GitHub. Вона є кроссплатформеною (підтримує Windows, macOS і Linux) і може використовуватися для створення додатків для пристроїв, хмари і Інтернету речей. . NET Core дозволяє створювати додатки і бібліотеки на мовах C#, Visual Basic і F #. .NET Core володіє наступними характеристиками:

- Кросплатформеність. Підтримка операційних систем Windows, macOS і Linux;
- Узгодженість між архітектурою. Однакове виконання коду в різних архітектурах, включаючи x64, x86 і ARM;
- Програми командного рядка. Зручні інструменти для локальної розробки і сценаріїв безперервної інтеграції;
- Гнучка розробка. Може включатися в додаток або встановлюватися паралельно на рівні користувача або комп'ютера. Можливість використання з контейнерами Docker;
- Сумісність. Платформа .NET Core сумісна з .NET Framework, Xamarin і Mono завдяки .NET Standard;
- Відкритий код. Платформа .NET Core має відкритий код і поширюється за ліцензіями MIT і Apache 2. .NET Core є проектом .NET Foundation;
- Підтримка від Майкрософт. Корпорація Майкрософт надає підтримку .NET Core.

ASP.NET Core є кроссплатформеним, високопродуктивним середовищем з відкритим вихідним кодом для створення сучасних хмарних додатків, підключених до Інтернету. ASP.NET Core дозволяє виконувати наступні завдання:

- Створювати веб-додатки та служби, додатки IoT і серверні частини для мобільних додатків;
- Використовувати вибрані засоби розробки в Windows, macOS і Linux;
- Виконувати розгортання в хмарі або локальному середовищі;
- Працювати в .NET Core або .NET Framework.

При проектуванні серверної частини програмного забезпечення було вирішено побудувати багаторівневу архітектуру. Таким чином, серверна частина буде долучати до себе:

-

Data Access Layer (DAL) – рівень доступу до даних, у нашому випадку це рівень доступу до нашої бази даних. Цей рівень містить визначення транспортних моделей для роботи з базою даних та серверною частиною, певні методи для роботи з базою та конфігураційні файли для бази даних;

- Service Layer або Business Logic Layer (BLL) – рівень бізнес логіки системи, де іде обробка даних у відповідності логіки системи. На цьому рівні знаходиться увесь основний функціонал для роботи з моделями як зі сторони бази даних, так й зі сторони веб-додатку, а також усі правила бізнес логіки системи в цілому. Це є основний рівень, що відповідає за обробку даних;
- Presentation Layer або WebApi – рівень представлення, де відбувається взаємодія з веб-додатком. На цьому рівні наша серверна частина програмного забезпечення спілкується з веб-додатком за допомогою запитів зі сторони веб-додатку, та кінцевим результатом обробки зі сторони серверу – відповіддю від нього. Усі запити та відповіді було вирішено збудувати за принципом REST. REST (Representational State Transfer) - архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгоджений набір обмежень, що враховуються при проектуванні розподіленої

гіпермедіа-системи. У певних випадках (інтернет-магазини, пошукові системи, інші системи, засновані на даних) це призводить до підвищення продуктивності і спрощення архітектури. У широкому сенсі компоненти в REST взаємодіють на зразок взаємодії клієнтів і серверів у Всесвітній павутині.

Отже, стандартна схема роботи серверної частини у простому виді буде виглядати наступним чином:

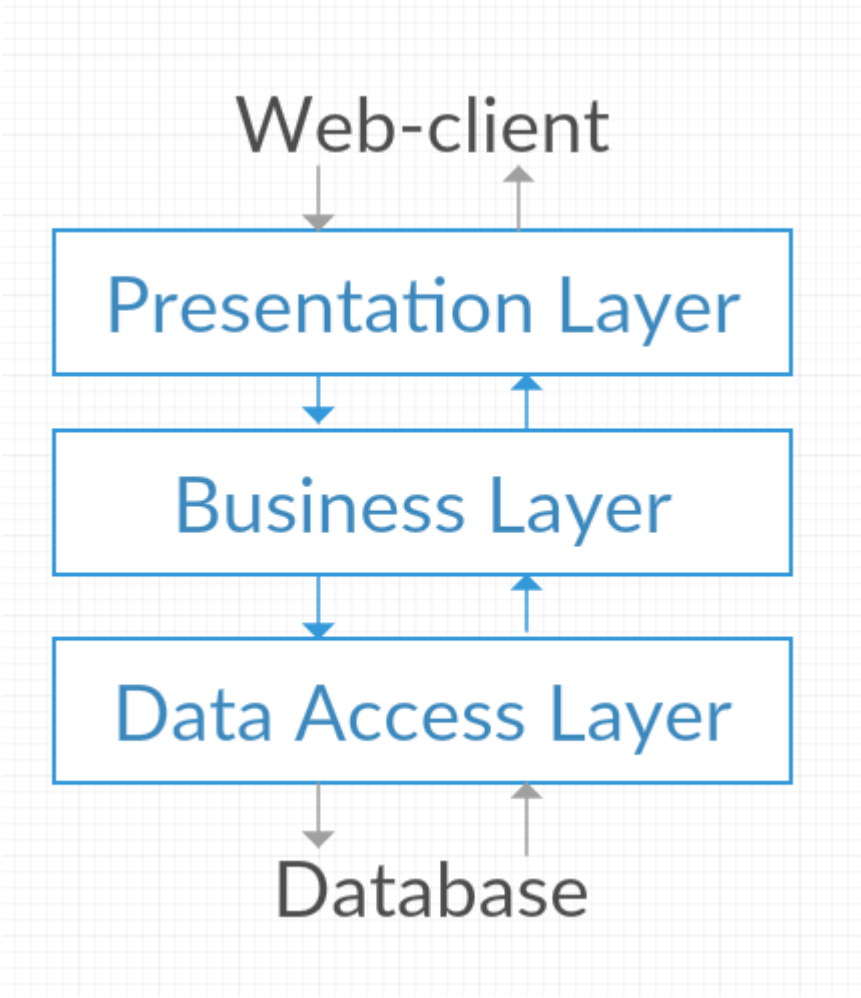

Рисунок 3.2 – Схема багаторівневої архітектури

Описати цю схему можна наступним чином:

- Спочатку веб-клієнт або веб-додаток відсилає запит до серверу. Запит потрапляє до рівня представлення. Цей рівень містить у собі набір різноманітних методів та інструментів, які можуть та вміють

приймати запити від веб-додатку, певним чином їх обробляти так, що передати на рівень нижче;

- Після обробки запиту вони посилають створені з запиту дані до рівня бізнес логіки, де й відбувається основна обробка даних згідно з бізнес правилами, встановленими при розробці та проектуванні системи. Набір функцій та методів обробляє потраплені до них дані та в результаті транзакції, він передає вже оброблені та змінені дані під роботу відповідно до рівня доступу даних;
- Після потрапляння до рівня доступу даних, дані які прийшли на рівень вище будуть використовуватися для взаємодії з джерелом та сховищем даних, у нашому випадку це база даних. В результаті успішної операції транспорту та обробки даних до бази даних ми отримує відповідь що все гаразд й передаємо у рівень, який з нею зв’язан, тобто рівень доступу до даних, далі рузльтат передається на рівень вище, й так ланцюг йде у зворотньому порядку допоки у рівні представлення не сформується відповідь, яка й надішлеться до веб-додатку.

3.2.2 Проектування веб-додатку програмного забезпечення

Для розробки веб-додатку програмного забезпечення було вирішено використовувати мову програмування JavaScript та платформу NodeJs та фреймворк React Js в парі з Redux.

JavaScript – мультипарадигмена мова програмування. Підтримує об'єктно-орієнтований, імперативний і функціональний стилі. Є реалізацією мови ECMAScript (стандарт ECMA-262).

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування

знаходить в браузерах як мова сценаріїв для додавання інтерактивності веб-сторінок.

Основні архітектурні риси:

- Динамічна типізація;
- Слабка типізація;
- Автоматичне керування пам'яттю;
- Прототипне програмування, функції як об'єкту першого класу.

Node або Node.js – програмна платформа, заснована на движку V8, що перетворює JavaScript з вузькоспеціалізованої мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel і espruino). В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням / висновком.

React (старі назви: React.js, ReactJS) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використаний у поєднанні з іншими JavaScript бібліотеками або в

великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Redux - це інструмент управління як станом даних, так і станом інтерфейсу в JavaScript-додатках. Він підходить для односторінкових додатків, в яких управління станом може з часом стає складним. Redux не пов'язаний з якимось певним фреймворком, і хоча розроблявся для React, може використовуватися з Angular або jQuery.

==========

При проектуванні веб-додатку було вирішено створити SPA(Single Page Application), використовуючи вище наведені фреймворки. Веб-додаток буде мати можливість працювати з файлами JSON формату, крім того, це буде основний його формат. Усі запити та відповіді будуть використовувати цей формат. Усі звертання до серверу повинні також як і на сервері бути узгоджені з методологією REST. Також буде використовуватися локальне сховище даних для зручності роботи веб-додатку – Redux. Веб-додаток повинен буде мати можливість роботи с асинхронністю. Основне завдання веб-додатку – надати користувачеві зручний інтерфейс для роботи з сервером. Саме тому він не тільки має «User-friendly», та й коректно відсилати запити та реагувати на отримані відповіді. Також у веб-додатку буде впроваджена клієнтська валідація. Основною перевагою такого веб-додатку мають бути швидкість роботи, швидкість спілкування з сервером та оперативність реагування на відповіді від нього та відсутність ре-рендорингу сторінки браузеру.

Схема роботи React+Redux:

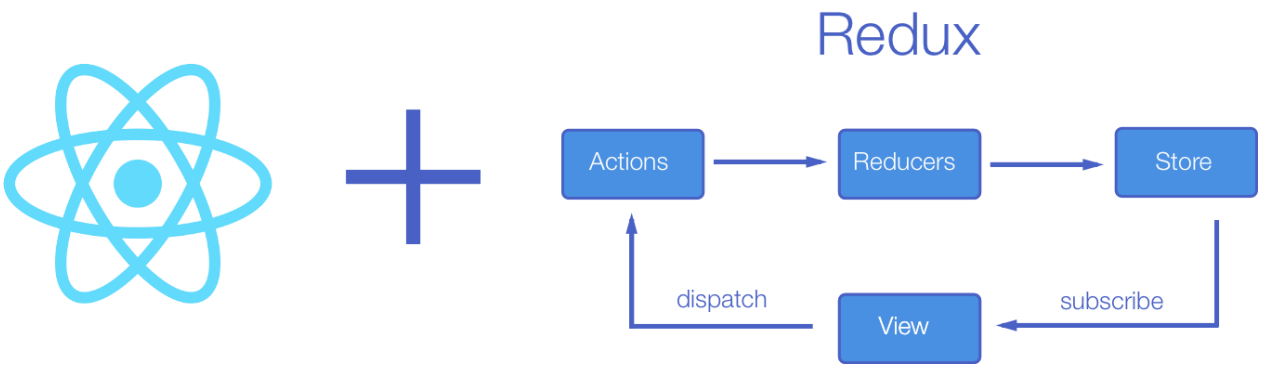

Рисунок 3.3 – Схема роботи React+Redux

Як можемо бачити, використання Redux передбачає наявність екшонів (actions) і ред'юсорів (reducers), а також констант (constants), які використовуються для зв'язку екшонів з ред'юсерами за допомогою передачі type (типу екшону).

Також екшони можуть бути виловлені певним мідлваром або сагою для подальшої їх обробки. Частіш за усе це використовується для відправки даних з екшону на сервер, попередньо обробивши їх.

У даному випадку буде використовуватися redux-saga.

Redux-saga - це бібліотека, яка покликана спростити і покращити виконання сайд-ефектів (тобто таких дій, як асинхронні операції, типу завантаження даних і "брудних" дій, типу доступу до браузерних кешу) в React / Redux додатках.

Можна уявити це так, що сага - це як окремий потік в веб-додатку, який відповідає за сайд-ефекти. Redux-saga - це Redux мідлвар, що означає, що цей потік може запускатися, зупинятися і скасовуватися з основного додатка за допомогою звичайних Redux екшонів, воно має доступ до повного станом Redux додатки і також може діспатчити Redux екшени.

Бібліотека використовує концепцію ES6, під назвою генератори, для того, щоб зробити ці асинхронні потоки легкими для читання, написання та тестування. Тим самим, ці асинхронні потоки виглядають, як стандартний синхронний JavaScript код. (На зразок async / await, але генератори мають кілька відмінних можливостей)

3.2.3 Проектування бази даних програмного забезпечення

Для розробки бази даних програмного забезпечення було вирішено використовувати тип баз даних Transact-SQL (T-SQL), а точніше MS SQL та безпосередньо SQL Server Express.

База повинна генеруватися на основі коду, моделей та конфігурацій, написаних для неї на серверній частині програмного забезпечення. Такий підхід називається Code First. Також вона повинна бути правильно сконфігурованна під роботи з Entity Framework Core.

SQL – декларативна мова програмування,яка застосовується для створення, модифікації та управління даними в реляційній базі даних, керованій відповідною системою управління базами даних.

Є перш за все інформаційно-логічною мовою, призначеною для опису, зміни і вилучення даних, що зберігаються в реляційних базах даних. SQL вважається мовою програмування, в загальному випадку (без ряду сучасних розширень) не є тьюринг-повним, але разом з тим стандарт мови специфікацією SQL/PSM передбачає можливість його процедурних розширень.

Transact-SQL (T-SQL) - процедурне розширення мови SQL, створене компанією Microsoft (для Microsoft SQL Server) і Sybase (для Sybase ASE).

SQL був розширений такими додатковими можливостями як:

- Керуючі оператори;
- Локальні і глобальні змінні;
- Різні додаткові функції для обробки рядків, дат, математики і т. п.;
- Підтримка аутентифікації Microsoft Windows.

Мова Transact-SQL є ключем до використання MS SQL Server. Всі додатки, які взаємодіють з екземпляром MS SQL Server, незалежно від їх реалізації і призначеного для користувача інтерфейсу, відправляють серверу інструкції Transact-SQL.

Microsoft SQL Server Express - це версія системи керування базами даних Microsoft SQL Server, яка є безкоштовною для завантажень, поширення і використання. Вона містить базу даних, спеціально призначену для вбудованих і маломасштабних додатків. Продукт веде своє коріння до продукту Microsoft Database Engine (MSDE), що поставляється з SQL Server 2000. Фірмова символіка Express використовується з моменту випуску SQL Server 2005.

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану, легковажну і розширюючуюся технологію від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping - відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати з базами даних, але на більш високому рівні абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища.

Схема бази даних у даній роботі виглядає так:

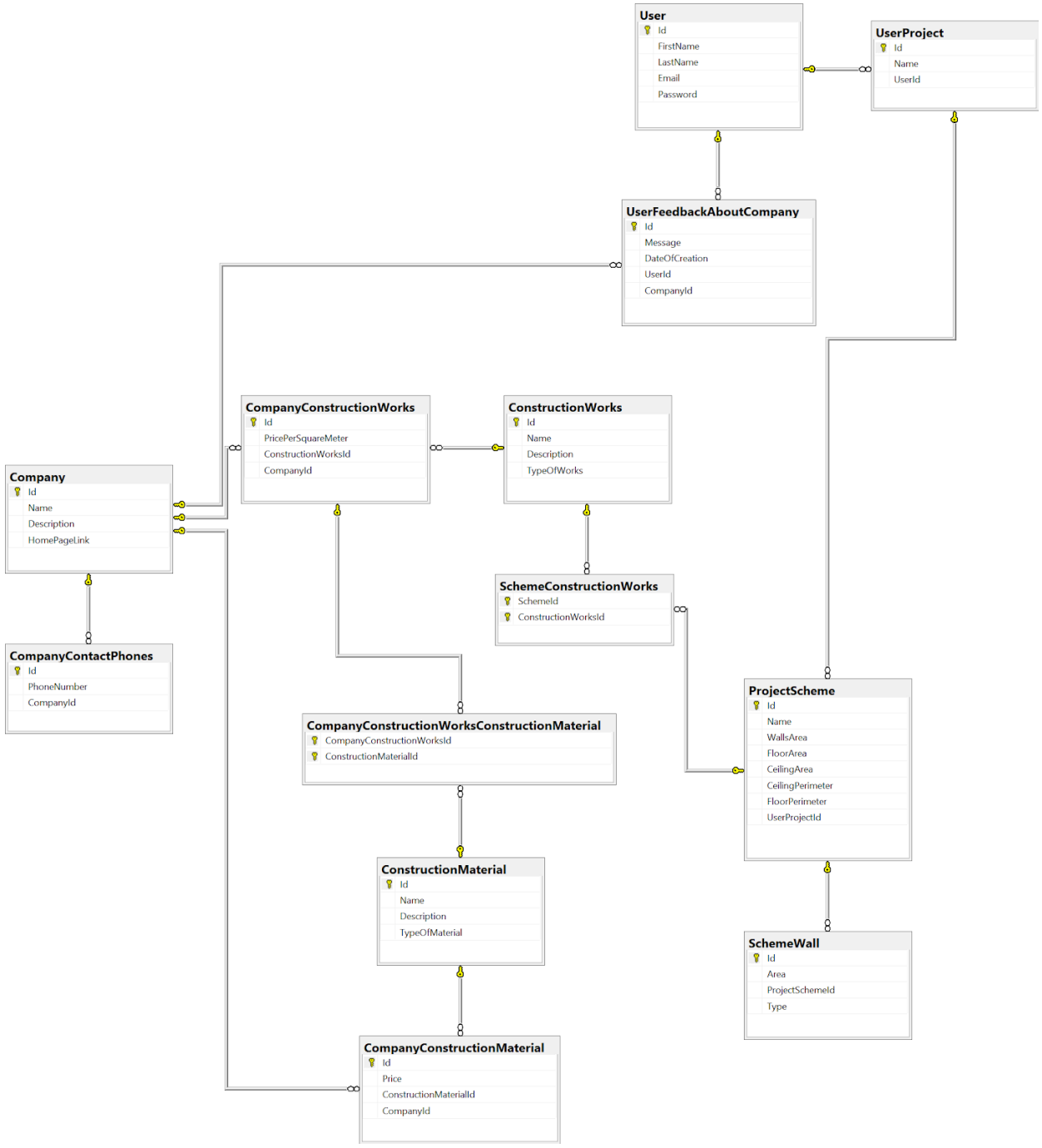

Рисунок 3.4 - Схема бази даних

ВИСНОВКИ

У ході виконання атестаційної роботи було розглянуто тенденції у сфері ремонтних будівельних робіт, а саме програмних продуктів, які дозволяють розраховувати фінансові витрати для ремонту квартир.

На основі проведеного аналізу та виявлених проблем було спроектовано та розроблено систему, що дозволяє користувачу автоматично підраховувати фінансові витрати для ремонту своєї або своїх квартир, а також користуватися зручними елементами менеджменту етапу ремонту.

Дана система складається з веб-серверу, веб-сайту та власної бази даних. Кожна із складових представлена у вигляді незалежного компоненту. Це робить систему більш гнучкою та розширюваною, дозволяючи вносити зміни у окрему частину, не торкаючись інших.

Слід відзначити, що реалізований продукт може бути розширеним та удосконаленим. Так, наприклад, можна додати конструктор, де користувач за допомогою UI буде конструювати свою квартиру у форматі 3D моделі. В цілях підвищення зручності користування системою та покриття більш широкої аудиторії доцільним буде створити мобільний додаток для різних платформ (Android, Windows Phone, iOS).

Крім того, важливим є той факт, що окремі складові системи можуть використовуватися для створення нових проектів у інших сферах життя.

ПЕРЕЛІК ПОСИЛАННЬ

1. Діго С.М. Бази даних: проектування та використання [Текст]: навч. / С.М.Діго. - М .: «Фінанси і статистика», 2005 - 518 с
2. https://metanit.com/sharp/mvc/
3. Head First C#, Jennifer Greene, Andrew Stellman
4. C# 4.0: повне керівництво, Герберт Шилдт.
5. Хендерсон, К. Microsoft SQL Server: структура та реалізація. Професійне керівництво [Текст]: Пер. з англ. - М.: Видавничий дім «Вільямс», 2006. - 1056с.
6. Троелсен, Е. C # і платформа .NET. Бібліотека програміста Е. Троелсен - СПб .: Питер, 2013. - 1310с.
7. JavaScript. Детальний керівництво, 6-е видання. - Пер. з англ. - СПб: Символ-Плюс, 2012. - 1080 с.
8. Дейт, К. Дж. Введення у системи баз Даних [Текст] / К. Дж. Дейт.- 7-е вид. - М.: Вид. дім «Вільямс», 2001. - 846 с.