ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Функціональна модель системи бронювання житла на основі платформи Salesforce

Кваліфікаційна робота
Другий (магістерський) рівень

Автор:
Квасов Є.О.
студ.гр.КСМм-20-1

Керівник:
Федорченко В.М.
доц. каф. ЕОМ

1

МЕТА ТА ЗАВДАННЯ

Метою кваліфікаційної роботи є розробка функціонального, математичного, алгоритмічного та програмного забезпечення бізнес-процесу ведення обліку роботи з бронюваннями на основі CRM-платформи Salesforce.

Завданнями кваліфікаційної роботи є:

- дослідження основних бізнес-процесів роботи рієлторських агенцій та сайтів з бронювань нерухомотсті
- здійснення аналізу інтерфейсу та функціональності вже існуючих програмних продуктів-аналогів
- аналіз можливостей сучасних CRM-платформ
- розроблення функціональної моделі діяльності агентств з пошуку нерухомості (діаграма використань для адміністратора та діаграма використань для клієнта)
- проєктування UI-інтерфейсу програмного модулю
- розроблення програмного продукту, проєктних та технічних рішень

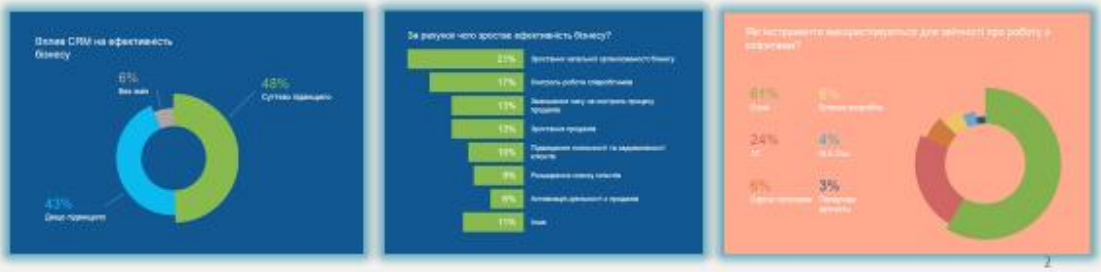
Актуальність теми дослідження визначається необхідністю підвищення рівня автоматизації бізнес-процесів для поліпшення їх ефективності.

2

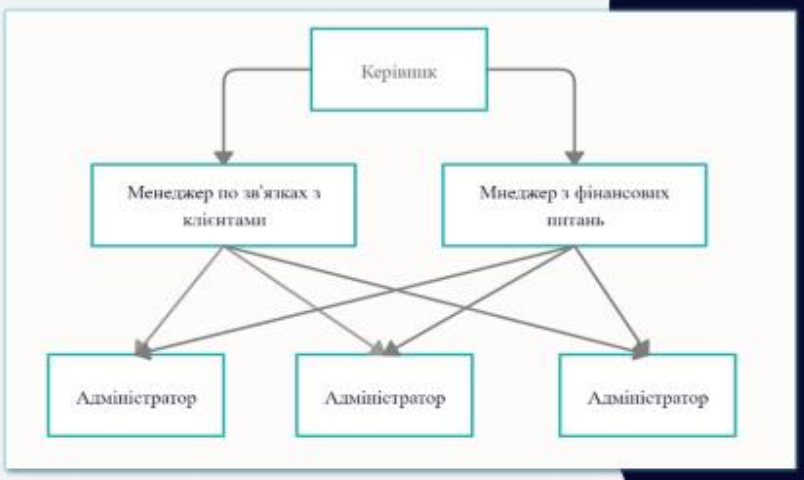
ДОКЛАДНІШЕ ПРО CRM

Концепція **CRM (Customer Relationship Management)** означає, що розрізнені інструменти ведення бізнесу об'єднуються в налагоджену систему. Замість табличок Excel, месенджерів, багатьох документів та біганини по кабінетах залишається один-єдиний сервіс.

У нього входять програми для збору даних про клієнтів, управління угодами, контролю за менеджерами, аналітики і прогнозування. Він спрощує рутину, прискорює прийняття правильних рішень і виключає помилки.

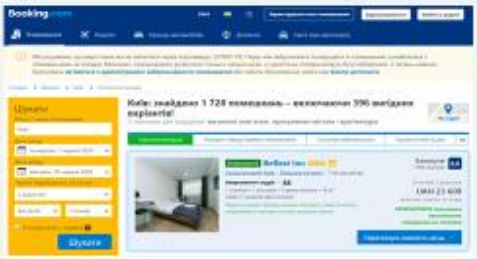


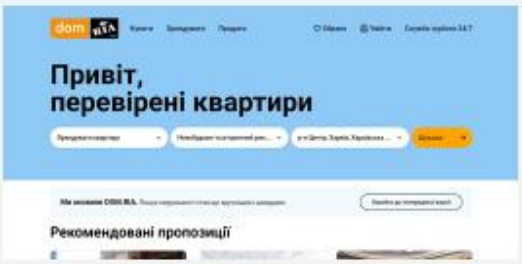
Результатом розроблення схеми організаційної структури управління підприємством є наступний опис основних структурних елементів:

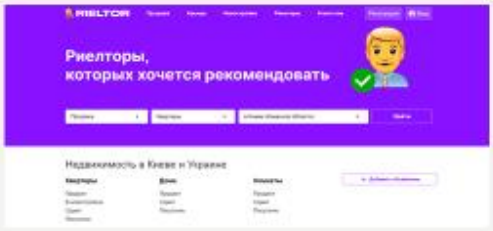


4

АНАЛОГИ СИСТЕМ ДЛЯ АГЕНТСТВА З ПОШУКУ НЕРУХОМОСТІ







5

ПОПУЛЯРНІ CRM-СИСТЕМИ

Zendesk

AgileCRM

HubSpot

ZOHO CRM

Freshsales

pipedrive

insightly

salesforce

6

SALESFORCE

За замовчуванням система Salesforce надає користувачам базовий набір різноманітного функціоналу. Вона повністю візьме на себе виконання рутинної роботи, яка зазвичай забирає багато часу у співробітників. Після впровадження менеджери зможуть більше часу приділяти своїм безпосереднім обов'язкам, а система буде займатися:

- веденням клієнтської бази;
- створенням історії взаємовідносин з клієнтами;
- швидким оформленням необхідних документів;
- організацією, плануванням роботи з клієнтами;
- розсиланням рекламних матеріалів з вигідними пропозиціями для покупців;
- створенням звітів про виконану роботу;

7

ІНФОГРАФІКА CRM-СИСТЕМ

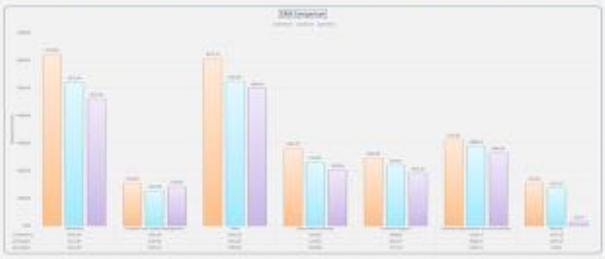
Salesforce

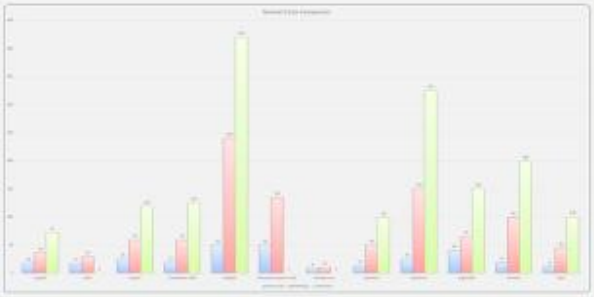
HubSpot

Zendesk

8

ОЦІНКА ФУНКЦІОНАЛУ CRM-СИСТЕМ

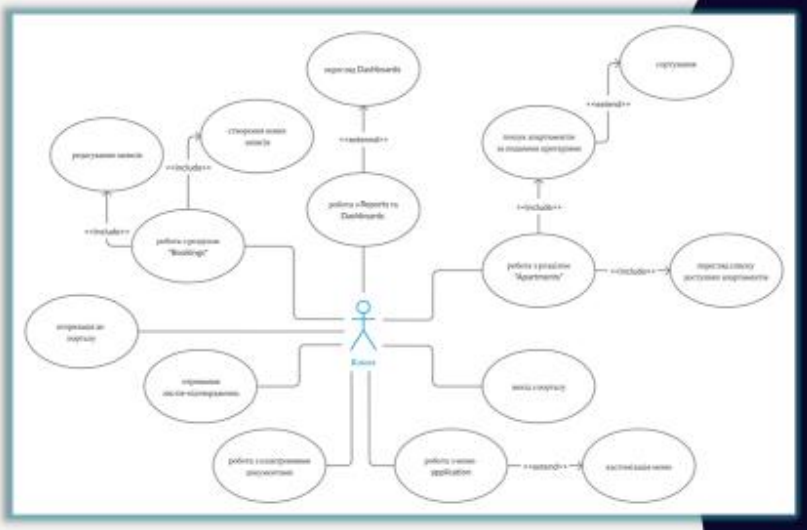




9

ОЦІНКА ФУНКЦІОНАЛУ CRM-СИСТЕМ

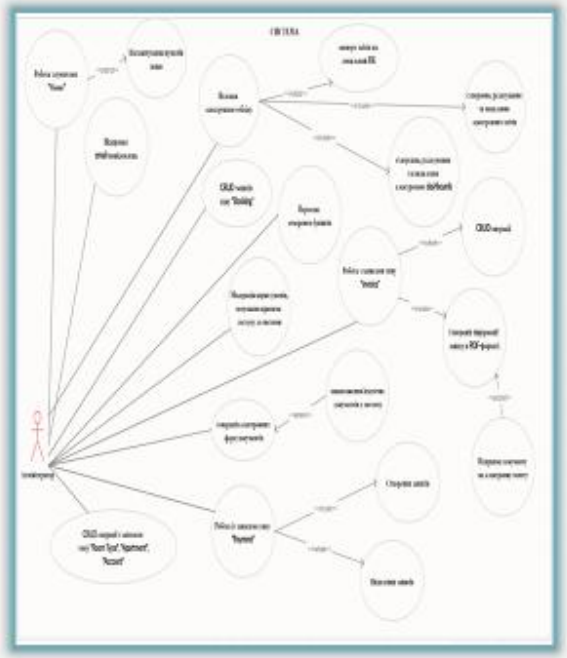
10



ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

ДЛЯ КЛІЄНТА

10

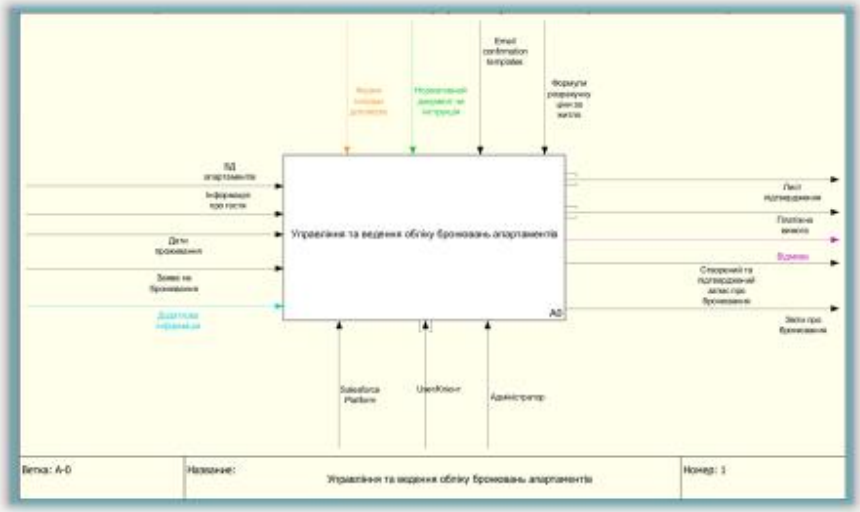


ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

ДЛЯ АДМІНІСТРАТОРА

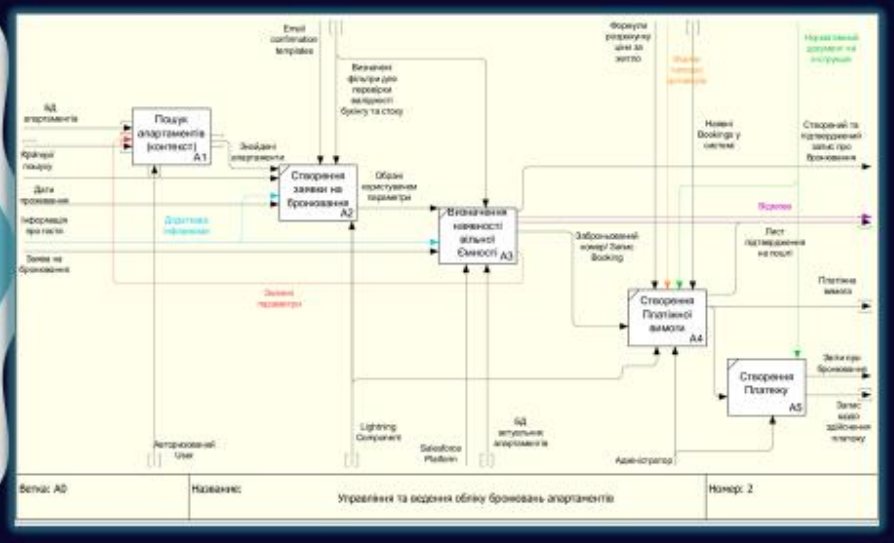
11

КОНТЕКСТНА ДІАГРАМА IDEFO



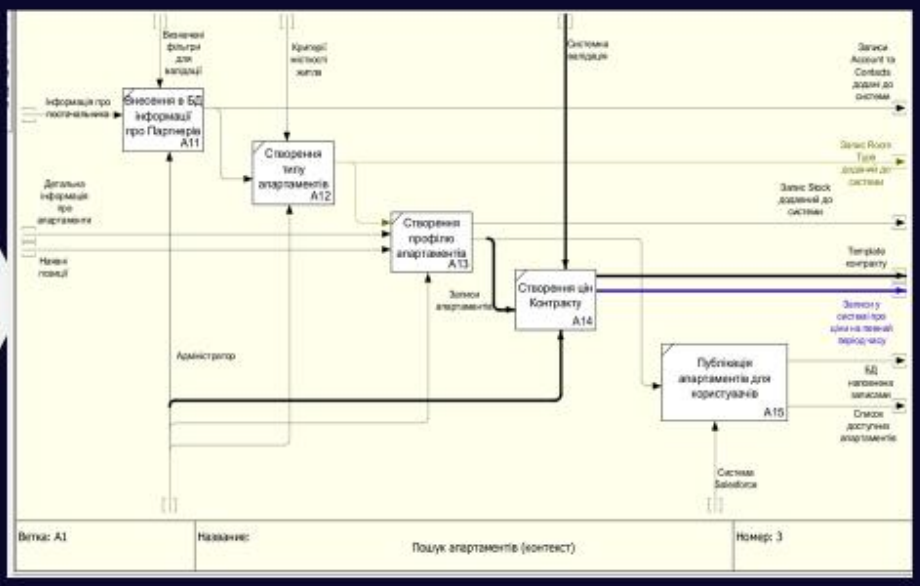
12

ДІАГРАМА ДЕКОМПОЗИЦІЇ IDEFO



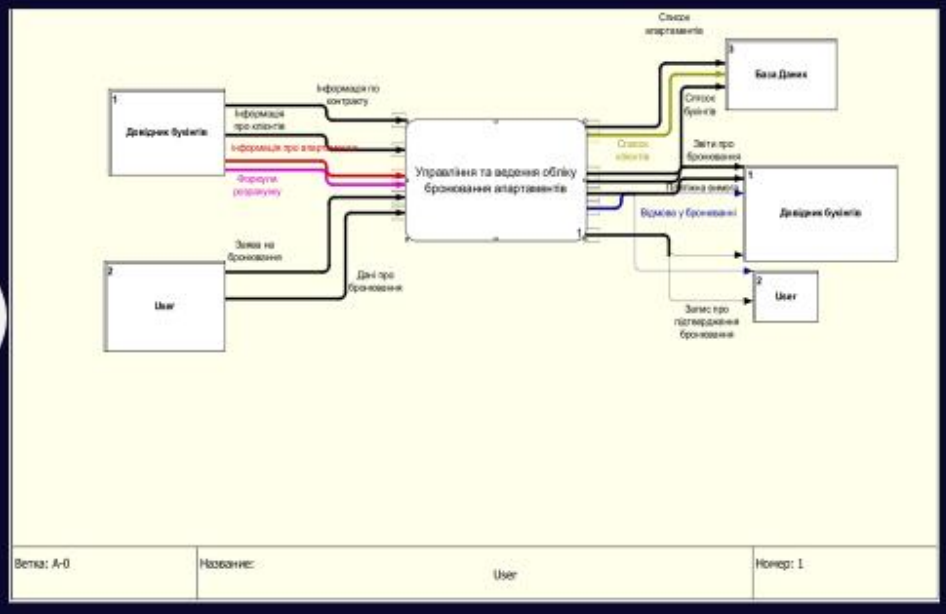
13

ДІАГРАМА ДЕКОМПОЗИЦІЇ IDEF0



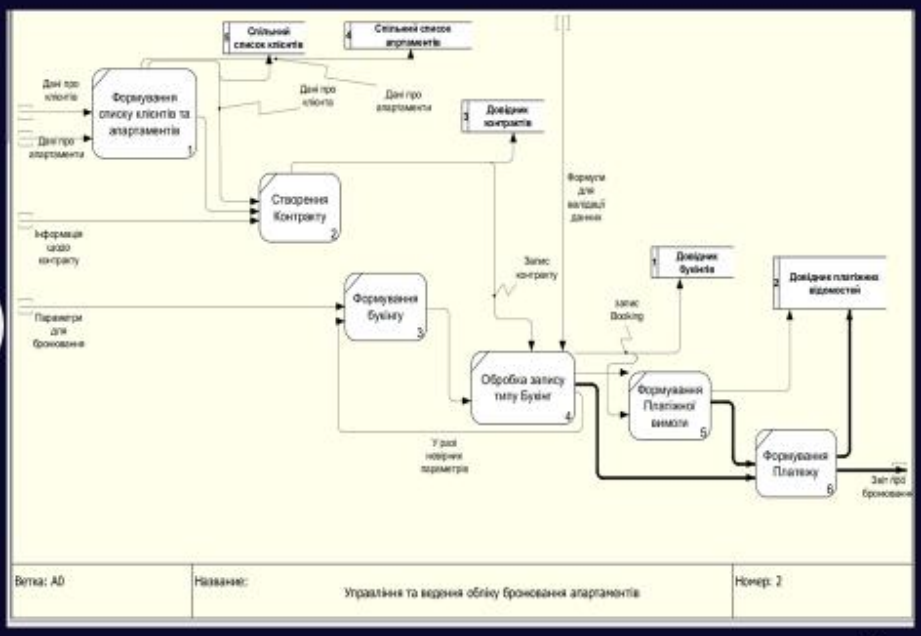
14

ДІАГРАМА ДЕКОМПОЗИЦІЇ DFD



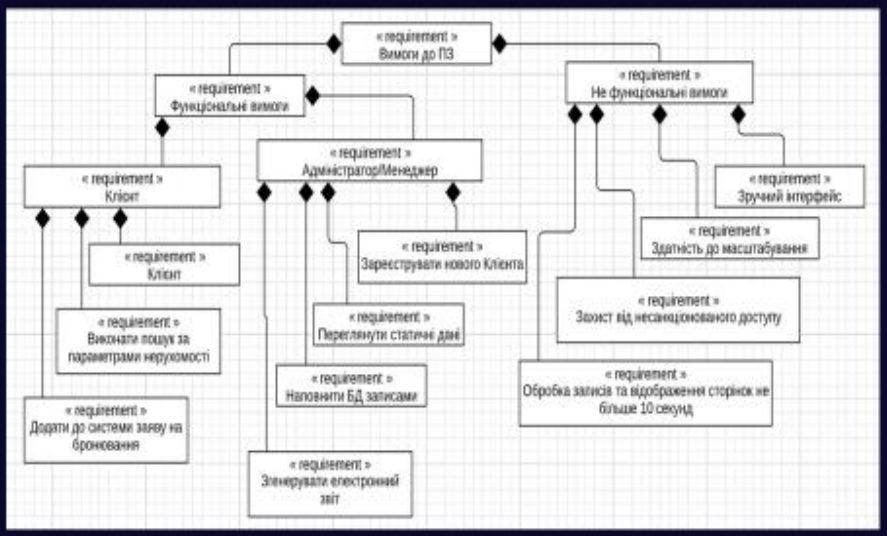
15

ДІАГРАМА ДЕКОМПОЗИЦІЇ DFD



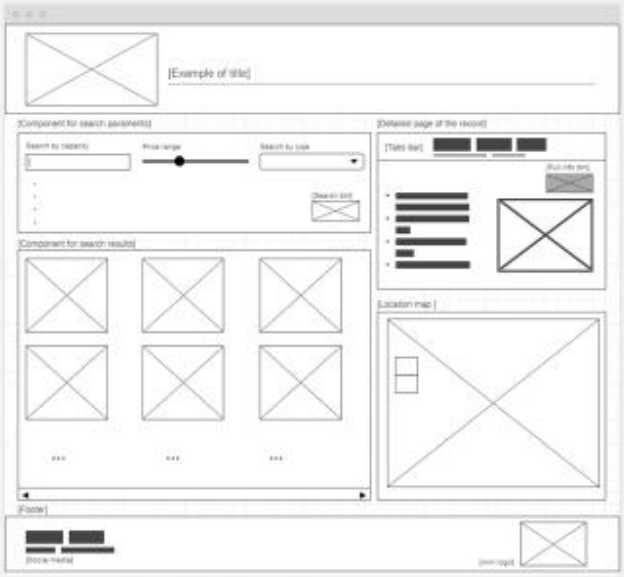
16

ДІАГРАМА ВИМОГ



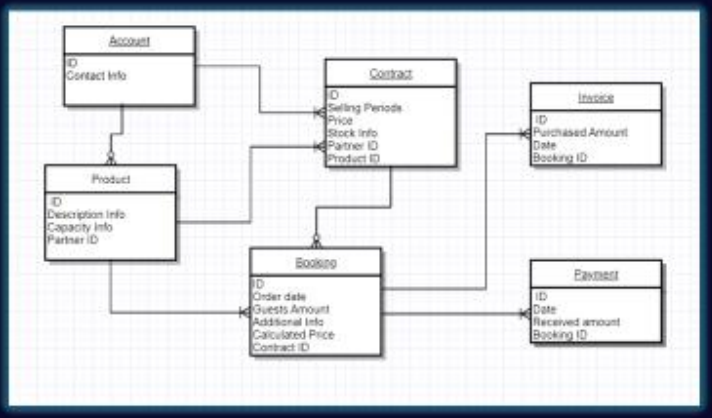
17

МАКЕТ КОМПОНЕНТУ ДЛЯ ПОШУКУ АПАРТАМЕНТІВ



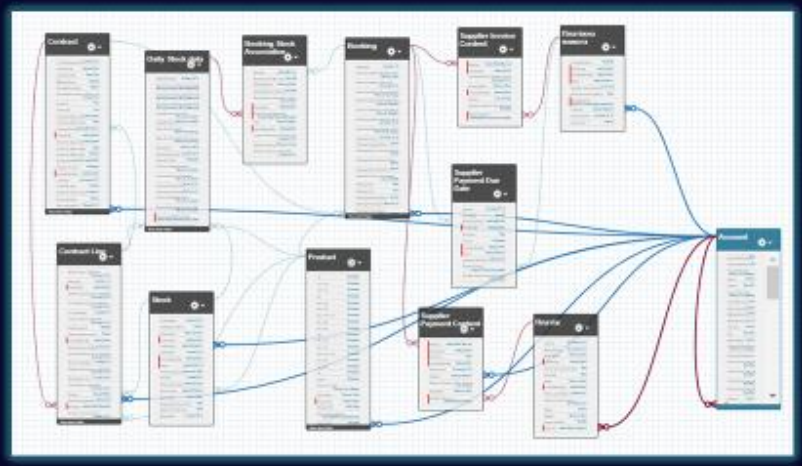
18

ЛОГІЧНА МОДЕЛЬ БАЗИ ДАНИХ



19

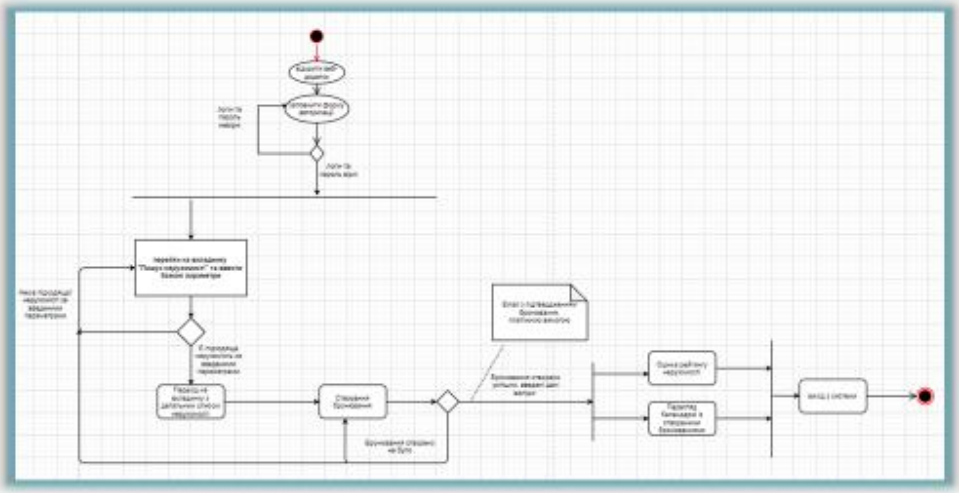
ФІЗИЧНА МОДЕЛЬ БАЗИ ДАНИХ



20

ДІАГРАМА ДІЯЛЬНОСТІ

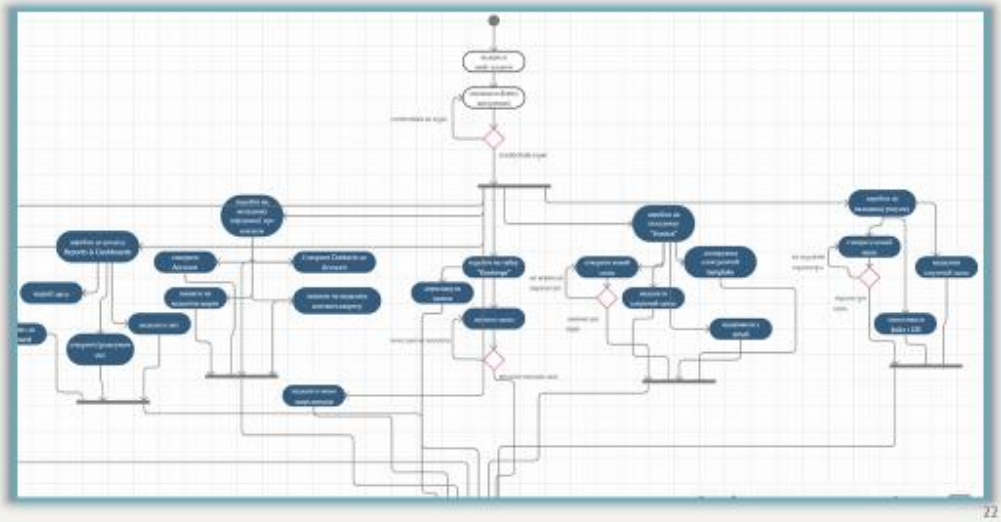
Для клієнта.



21

ДІАГРАМА ДІЯЛЬНОСТІ

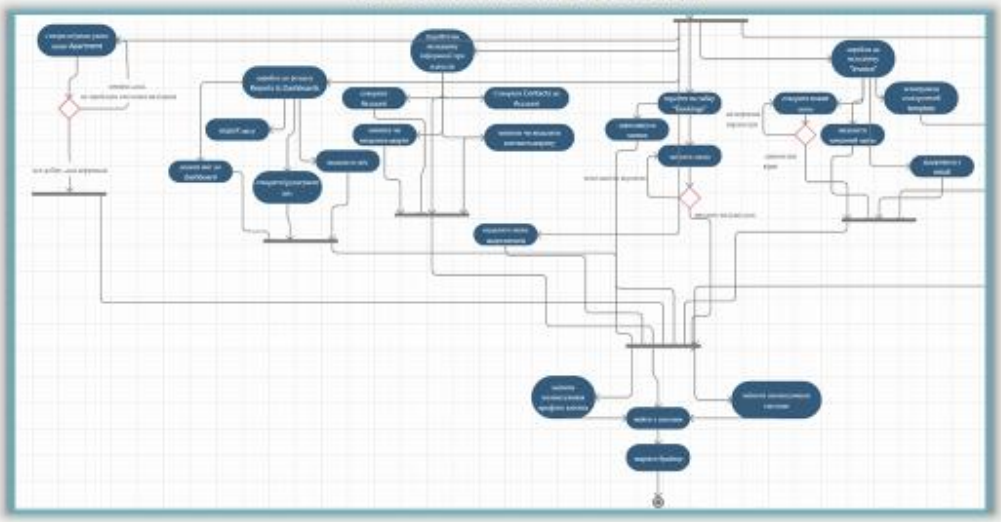
Для адміністратора, початок.



22

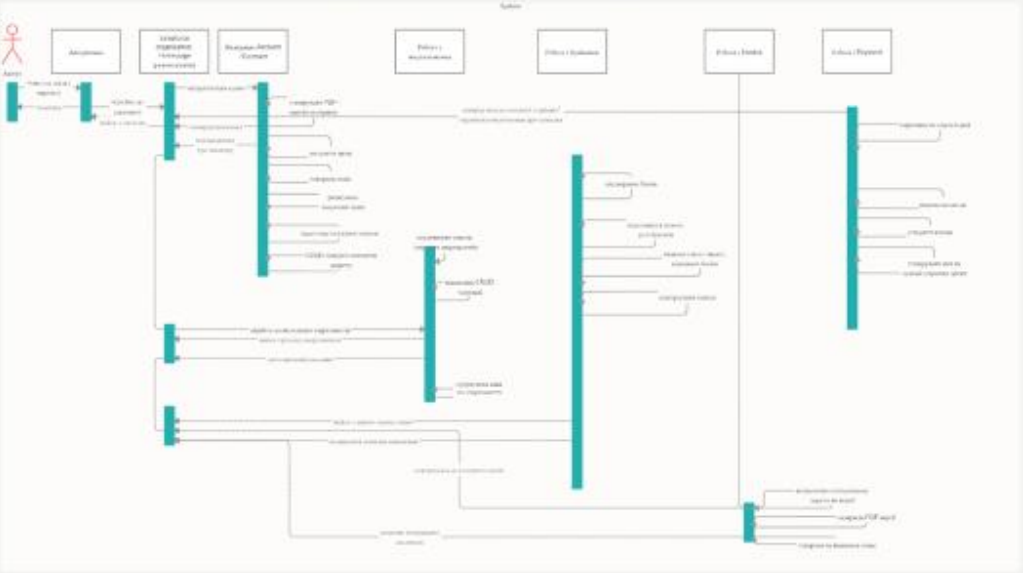
ДІАГРАМА ДІЯЛЬНОСТІ

Для адміністратора, продовження.



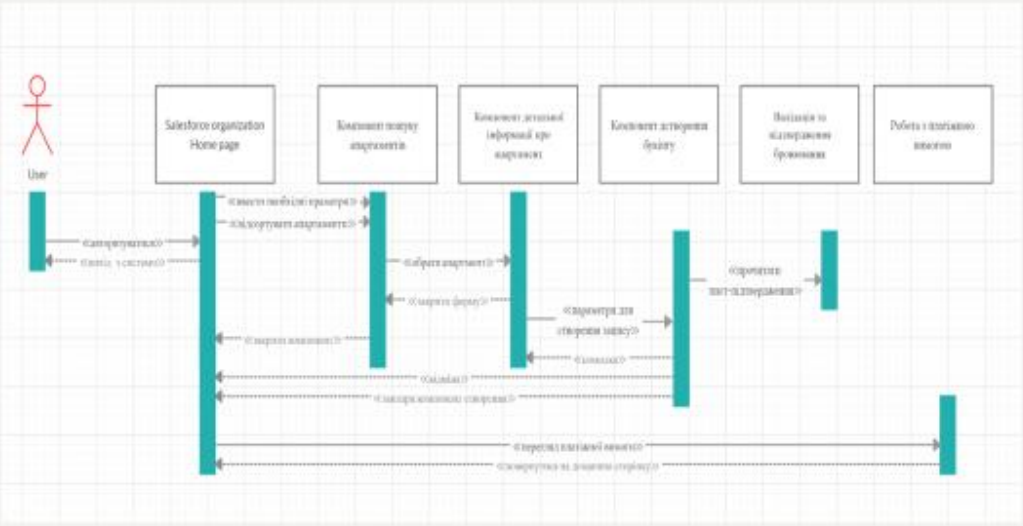
23

ДІАГРАМА ПОСЛІДОВНОСТІ ДІЙ ДЛЯ АДМІНІСТРАТОРА



24

ДІАГРАМА ПОСЛІДОВНОСТІ ДІЙ ДЛЯ КЛІЄНТА



25

ДОКУМЕНТИ

Вхідні документи

- Заява на бронювання
- Наявна у БД інформація щодо апартаментів
- Інформація про гостя

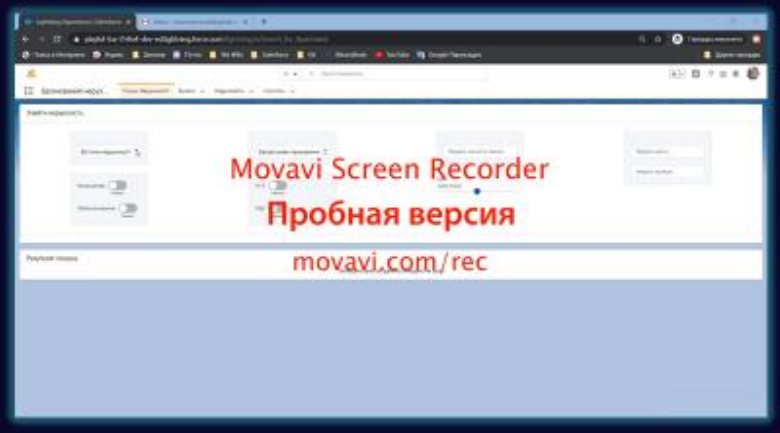
Вихідні документи

- Лист підтвердження
- Платіжна вимога
- Платіж
- Створений та підтверджений букінг
- Звіти про бронювання

26

ІНТЕРФЕЙС

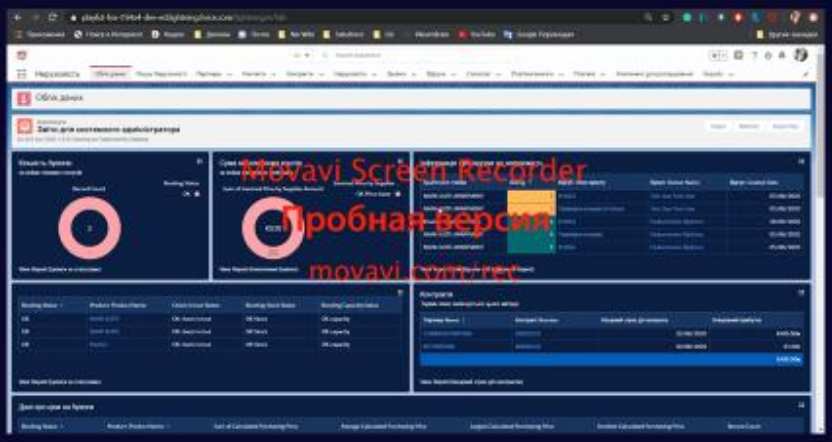
ДЛЯ КЛІЄНТА



27

ІНТЕРФЕЙС

ДЛЯ АДМІНІСТРАТОРА



28

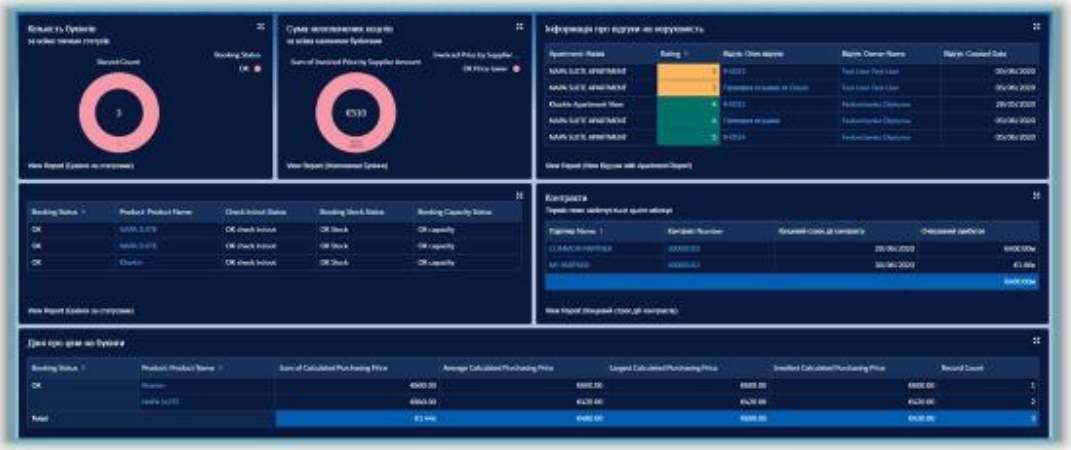
ВИХІДНІ ДОКУМЕНТИ

- Платіжна вимога
- Платіж

29

ВИХІДНИЙ ДОКУМЕНТ

- Звіти



30

ВИСНОВКИ

- У ході виконання даної кваліфікаційної роботи було розглянуто проблеми автоматизації діяльності агентства з пошуку нерухомості з використанням сучасних засобів обробки інформації. В роботі зроблена якісна постановка задачі, розглянуті методи проектування інформаційних систем. Розроблено функціональну модель та алгоритмічне забезпечення для поставленої задачі.
- У представленій кваліфікаційній роботі була обґрунтована актуальність розробки моделі та автоматизації процесу пошуку нерухомості, спроектовано модель користувальницького інтерфейсу. На підставі бізнес правил були розроблені функціональні та нефункціональні вимоги до програмної ситеми, діаграми варіантів використання, модель даних.
- Детально описані можливості програмної системи, обрано цільовий варіант архітектури, розроблена архітектура системи, було описано обрані технічні засоби.
- Модель розроблена для інформаційної та аналітичної підтримки діяльності агенства з пошуку нерухомості. Користувачами цієї системи мають бути клієнти, які шукають житло, та адміністратори системи чи менеджери рієлторских агенцій. Розроблене програмне забезпечення розгорнуте на платформі світового лідера з надання CRM послуг, платформі для розробки Salesforce. Тож описане у даній роботі ПЗ користується системою захисту та безпеки компанії з світовим досвідом та визнанням однойменної компанії Salesforce.

31

ДОДАТОК Б

Словник термінів

Таблиця Б.1 – Словник термінів

Термін	Опис терміну
1	2
1. Основні поняття і категорії предметної області та проєкту	
Credentials	логін та пароль користувача для авторизації у системі;
Посилання, гіперпосилання, лінк	вказівка порталу здійснити перехід на головну сторінку будь-якого розділу, або на одну з внутрішніх;
Звіт про дохід агенції	електронний документ чи таблиця, що містить інформацію щодо оцінки темпів росту чи падіння прибутків;
Контракт	запис у системі, який створюється адміністратором і містить у собі інформацію про ціни на апартаменти на певний період їх здачі;

Продовження таблиці Б.1

1	2
1. Основні поняття і категорії предметної області та проєкту	
Сток/Stock	об’єкт БД, що містить у собі інформацію про наявність вільного місця у певних апартаментах;
Табка/Tab	спеціальні вкладки що дозволяють відображати об’єктні дані чи інший веб-вміст у Salesforce. Відображається як елемент на панелі навігації програми;
Setup	головне меню налаштування Salesforce організації;
Контент	інформаційний зміст сайту (тексти, графічна, звукова інформація та ін.);
Відправка документів	можливість автоматично відправити наявний у системі документ на пошту клієнту;
Апартаменти	запис об’єкту БД, що представляє собою інформацію щодо житла, його опис та характеристику;
Локалізація	переклад призначеного для користувача інтерфейсу на потрібну мову;
Популярні апартаменти	незаміщені житлові пропозиції, відсортовані за датами та кількістю вільного місця;

Продовження таблиці Б.1

1	2
1. Основні поняття і категорії предметної області та проєкту	
Дашборд/Dashboard	це наочне представлення інформації у вигляді декількох електронних звітів про поточний прибуток підприємства, зосереджене на електронній дошці чи календарі;
Підтверджене бронювання	букінг, який було схвалено системою та адміністратором;
Бронювання/Booking	форма заповнена клієнтом компанії, котра містить у собі інформацію щодо бронювння нерухомості, а саме: кількість гостей, кількість кімнат, дати заїзду і виїзду, коментарі і т.д.;
Вподобані записи	апартаменти, на яких була поставлена позначка «Мені подобається»;
Detail page	системна сторінка окремого запису сутності у Salesforce;
Блокування користувача	заборона адміністратором системи взаємодіяти іншому користувачеві з системою;

Продовження таблиці Б.1

1	2
1. Основні поняття і категорії предметної області та проєкту	
Інтерфейс	засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача, тобто елементи сторінок порталу та їх розташування;
Іконка	Значок – елемент графічного інтерфейсу, невелике зображення, що репрезентує застосунок, файл, вікно, пристрій тощо;
Портал, організація	комплекс апаратно-програмних засобів, що дозволяє користувачам комп'ютерної мережі Інтернет обмінюватися повідомленнями за темами;
Ідентифікатор користувача (логін)	це ім'я користувача, яке буде відображатися тільки на сайті. Логін є псевдонімом та не має обов'язково повторювати справжнє ім'я людини;
Ідентифікаційний номер/Id	унікальний набір літер та цифр що закріпляється системою за кожним записом у системі, для подальшого використання адміністратором;

Продовження таблиці Б.1

1	2
1. Основні поняття і категорії предметної області та проєкту	
Авторизація	керування рівнями та засобами доступу до різних об’єктів (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від ідентифікатора і пароля користувача;
Service Type	об’єкт БД призначений для зберігання записів про наявну нерухомість у Поставника, її місткість;
Visualforce page	це фреймворк для розробки інтерфейсу односторінкових програм для мобільних та настільних пристроїв;
Lightning Component	це фреймворк для розробки інтерфейсу односторінкових програм для мобільних та настільних пристроїв;
Process Builder/ Конструктор процесів	допомагає автоматизувати бізнес-процеси, такі як оновлення записів, оповіщення електронною поштою та будь-які завдання, які призначить розробник і все це без єдиного рядка коду;

Продовження таблиці Б.1

1	2
1. Основні поняття і категорії предметної області та проєкту	
Головна сторінка/Home page	початкова сторінка порталу, яка, зазвичай, надає відомості про тематику веб-сайту та матеріали, дозволяє переглянути зміст та меню порталу;
Затверджені заяви на бронювання	заяви, які створені користувачем, та вже затверджені системою на адміністратором;
Поставник	безпосередній власник нерухомості, інформація про якого зберігається у системному об’єкті Account;
Web-service	технологія, що реалізує автоматичне завантаження набору даних між різними організаціями у Salesforce;
Сутність/Об’єкт БД	визначена структура об’єкту для подальшого створення, редагування та видалення записів даного типу у системі Salesforce;
Record Type	типи записів, які дозволяють пропонувати різним користувачам різні бізнес-процеси, значення списку вибору та макети сторінок;

Продовження таблиці Б.1

1	2
2. Користувачі системи	
Адміністратор порталу	користувач, який керує всім наповненням порталу і його контентом;
Юзер	клієнт/користувач портал, інформація про якого зберігається у системному об’єкті User;
Користувачі/Клієнти	всі користувачі порталу, які мають доступ до функціоналу системи;
3. Вхідні та вихідні документи	
Електронний документ	електронній документ, завантажений до порталу, що зберігається на ПК чи у хмарі;
Платіжна вимога/ Invoice	рахунок який має сплатити клієнт за бронювання;
Платіж/Payment	запис у системі, що є підтвердженням оплати бронювання;
Шаблон/Template	«PDF» версія запису, яка генерується за заздалегідь визначеною формою;
Вхідні повідомлення	підтвердження бронювань, які були отримані користувачем по електронній пошті;

Продовження таблиці Б.1

1	2
3. Вхідні та вихідні документи	
Відправлені повідомлення	підтвердження бронювань, у вигляді email, які були відправлені користувачем по електронній пошті , у чаті, або на форумі;
Форма бронювання/форма для заповнення	форма створення запису букінгу апартаментів, електронна форма для введення необхідних параметрів замовлення;
Звіт про дохід агенції	електронний документ чи таблиця, що містить інформацію щодо оцінки темпів росту чи падіння прибутків;
Контракт	запис у системі, який створюється адміністратором і містить у собі інформацію про ціни на апартаменти на певний період їх здачі;

ДОДАТОК В
Опис БД

В.1 Словник даних

Таблиця В.1 – Словник даних

Ім’я сутності	Описання	Псевдоніми	Коментарі
1	2	3	4
Партнер/ Контакт	Загальне позначення для характеристики орендодавців, які надаюсь нерухомість під оренду та їх потенційних представників. А також контактів, що відображають клієнтів підприємства, які будуть створювати заяви на бронювання нерухомості.	Account/ Contact	Об’єкти, що беруть участь у формуванні та представленні статичних даних щодо наявних партнерів підприємства. Кожний об’єкт Account може мати декілька посилань на пов’язаний із ним запис сутності Product2 та декілька посилань на пов’язані із ним записи сутностей: Contract, Contract_Lines, Product2, Supplier_Invoice, Supplier_Payment, Contact

Продовження таблиці В.1

1	2	3	4
Букінг	Загальне позначення для характеристики запитів на бронювання нерухомості	Booking	Може містити безліч посилань на об’єкти Contract_Lines, Supplier_Invoice,Supplier_Payment (зв’язок “один до багатьох”) та лише одне посилання на запис типу Account, Contract
Нерухомість/Ємність	Загальне позначення для безпосередньої характеристики представленої до оренди нерухомості. Об’єкт Ємність використовується для контролю вільного місця у апартаментах під бронювання	Product2/Stock	Може містити безліч посилань на об’єкти Booking__c, Contract_Lines__c (зв’язок “один до багатьох”). З об’єктом Account існує зв’язок «один до одного»
Контракт/ Ціноутворення	Загальне позначення для характеристики цін за нерухомість на певний період здачі її в оренду	Contract/ Contract_Lines	Contract може мати лише одне посилання на об’єкт Account та безліч посилань на записи Contract_Lines. Ціноутворення містить лише одне посилання на об’єкти Product2 та Stock та безліч посилань на записи букінгів

Закінчення таблиці В.1

1	2	3	4
Платіжна вимога/ Платіж	Парні характеристики, що використовуються для обліку інформації щодо платежів за бронювання нерухомості	Supplier_Invoice/ Supplier_Payment	Дані сутності можуть містити безліч посилань на об’єкт Booking__c, та лише одне посилання на об’єкт Account та Product2, адже за одну операцію можна здійснити оплату нерухомості одного конкретного Поставника
Daily Stock Data	Ця сутність використовується для описання інформації про кожен день ціноутворення і контролює кількість зайнятого гостями місця у нерухомості, а також використовується для підрахунку вартості Букінга	Daily Stock Data	Дана сутність може містити безліч посилань на записи букінгів (зв’язок «один до багатьох»), та лише одне посилання на такі об’єкти як Ємність, Ціноутворення та Нерухомість

В.2 Сутність Booking

Таблиця В.2 - Сутність Booking.

Межі / допустимі значення	Структура (формат)	Умова	Значення за замовчуванням
1	2	3	4
Ім’я атрибута або агрегату: Ім’я орендатора , Pax_First_Name			
[A-Za-z]{3,15}	Text(50)	Обов’язкове поле	-
Ім’я атрибута або агрегату: Ім’я орендатора , Pax_First_Name			
[A-Za-z]{3,15}	Text(50)	Обов’язкове поле	-
Ім’я атрибута або агрегату: Основні умови проживання , Board_Basis			
Room Only Bed & Breakfast Half Board Full Board All Inclusive Ultra All Inclusive	Picklist	Обов’язкове поле Має відповідати аналогічному полю на Contract Line	-
Ім’я атрибута або агрегату: Орендодавець, Account			
-	Text(50), Lookup(Партнер)	Автоматично заповнюється системою на основі обраної нерухомості	-

Продовження таблиці В.2

1	2	3	4
Ім’я атрибута або агрегату: Мобільний телефон , Pax_Phone_Number			
^[0-9]{3}-[0-9]{3}-[0-9]{4}$	Phone	Обов’язкове поле	-
Ім’я атрибута або агрегату: Кількість малюків. Number_of_Babies			
Не повинно бути менше нуля	Number(1, 0)	-	0
Ім’я атрибута або агрегату: Кількість дітей, Number_of_Childs			
Не повинно бути менше нуля	Number(1, 0)	-	0
Ім’я атрибута або агрегату: Кількість дорослих , Number_of_Adults			
Не повинно бути менше чи дорівнювати нулю	Number(1, 0)	Обов’язкове поле	1
Ім’я атрибута або агрегату: Контракт , Contract			
-	Lookup (Контракт)	Автоматично заповнюється системою на основі обраної нерухомості та дат перебування	-
Ім’я атрибута або агрегату: Додаткова інформація, Additional_information			
-	Rich Text Area(32768)	-	-

Продовження таблиці В.2

1	2	3	4
Ім’я атрибута або агрегату: Дата Check out , Booking_Check_out_Date			
Має бути більша за поточну дату на більше за дату check-in	Date	Обов’язкове поле	-
Ім’я атрибута або агрегату: Дата Check in , Booking_Check_in_Date			
Має бути більша за поточну дату	Date	Обов’язкове поле	-
Ім’я атрибута або агрегату: Бажана кількість кімнат , Number_of_Rooms			
-	Number(1, 0)	Обов’язкове поле	-
Ім’я атрибута або агрегату: Supplier Invoice Status , Supplier_Invoice_Status			
Received Received Less Conflict	Picklist	Автоматично присвоюється системою залежно від суми за букінг у платіжній вимозі	-
Ім’я атрибута або агрегату: Supplier Paid Amount, Supplier_Paid_Amount			
-	Roll-Up Summary (SUM Supplier Payment Content)	Автоматично присвоюється системою залежно від оплаченої суми за букінг	-

Продовження таблиці В.2

1	2	3	4
Ім’я атрибута або агрегату: Invoiced Price by Supplier Amount			
-	Roll-Up Summary (SUM Supplier Invoice Content)	Сума всіх платіжних вимог за конкретний букінг	-
Ім’я атрибута або агрегату: Email, Pax_Email__c			
-	Email	Обов’язкове поле	-
Ім’я атрибута або агрегату: Check in/out Status , Check_in_out_Status			
OK check in/out KO check in/out	Picklist	Автоматично розраховується системою на основі дат заїзду і виїзду	-
Ім’я атрибута або агрегату: Calculated Purchasing Price			
-	Currency(16, 2)	Система автоматично розраховує вартість букінгу залежно від кількості гостей та терміну проживання	-
Ім’я атрибута або агрегату: Booking Capacity Status			
OK capacity KO capacity Not Controlled	Picklist	Автоматично калькулюється системою на основі дат кількості гостей та ємності обраної нерухомості	-

Продовження таблиці В.2

1	2	3	4
Ім’я атрибута або агрегату: Booking Stock Status			
-	Picklist	Автоматично калькулюється системою на основі дат заїзду і виїзду, а також бажаної кількості кімнат	-
Ім’я атрибута або агрегату: Booking Supplier Payment Status			
Conflict Paid Consolidated Upcoming Paid Less	Picklist	Автоматично присвоюється системою залежно від оплаченої суми за букінг	Upcoming

В.3 Обмеження атрибутів сутностей Stock та Contract Line

Таблиця В.3 - Обмеження атрибутів сутностей Stock та Contract Line

Межі / допустимі значення	Структура (формат)	Умова	Значення за замовчуванням
1	2	3	4
Ім’я атрибута або агрегату: Stock Period From Date			
-	Date	Обов’язкове поле	Початок дії ціноутворення
Ім’я атрибута або агрегату: Stock Period To Date			
-	Date	Обов’язкове поле	Кінець дії ціноутворення

Продовження таблиці В.3

Ім’я атрибута або агрегату: Service Type			
-	Lookup (Нерухомість)	Обов’язкове поле	Автоматично заповнюється під час створення Ціноутворення
Ім’я атрибута або агрегату: Initial Stock			
Має бути більше нуля	Number(3, 0)	Обов’язкове поле	Автоматично заповнюється під час створення Ціноутворення
Ім’я атрибута або агрегату: Ємність , Stock__c			
-	Lookup(Stock)	Створюється автоматично та приєднується то того ж Поставника, що й обраний Контракт	-
Ім’я атрибута або агрегату: Власник нерухомості , Supplier			
-	Text(50), Lookup(Партнер)	Заповнюється автоматично при створення запису Ціноутворення	Постачальник вказаний у відповідному контракті
Ім’я атрибута або агрегату: Початок дії , Travel_Period_From			
-	Date	Має бути в рамках обраного Контракту Обов’язкове поле	Початок дії Контракту

Продовження таблиці В.3

1	2	3	4
Ім’я атрибута або агрегату: Основні умови проживання, Board_Basis			
Room Only Bed & Breakfast Half Board Full Board All Inclusive Ultra All Inclusive	Picklist	Створюваний у подальшому букінг має мати аналогічне значення у відповідному полі Обов’язкове поле	-
Ім’я атрибута або агрегату: Нерухомість , Room_Type			
-	Lookup (Нерухомість)	Має належати то того ж Поставника, що й обраний Контракт Обов’язкове поле	-
Ім’я атрибута або агрегату: Назва			
-	Auto Number	-	CL-{0001}
Ім’я атрибута або агрегату: Мінімальний термін проживання , Length_of_Stay_From			
Повинно бути більше 0, та не повторюватися з іншими Contract Line у рамках одного контракту	Number(2, 0)	Використовується для відстеження тривалості відповідних букінгів Обов’язкове поле	-

Продовження таблиці В.3

1	2	3	4
Ім’я атрибута або агрегату: Максимальний термін проживання , Length_of_Stay_To			
Повинно бути більше 0 і мінімального строку проживання	Number(2, 0)	Використовується для відстеження тривалості букінгів. Не має межувати з іншими Contract Line у рамках одного контракту Обов’язкове поле	-
Ім’я атрибута або агрегату: Кінець дії , Travel_Period_To			
-	Date	Має бути в рамках обраного Контракту Обов’язкове поле	Кінець дії Контракту
Ім’я атрибута або агрегату: Кількість вільного місця , Stock_period_quantity			
-	Number(18, 0)	Кількість запропонованого вільного місця на період дії Ціноутворення Обов’язкове поле	-
Ім’я атрибута або агрегату: Контракт , Contract			
-	Master-Detail (Контракт)	Заповнюється автоматично при створення запису Ціноутворення Обов’язкове поле	-

ДОДАТОК Г

Лістинг програми для компоненту пошуку

```
Component.cmp
<aura:component
implements="force:appHostable,flexipage:availableForAllPageTypes
,flexipage:availableForRecordHome,force:hasRecordId"
access="global" controller="FindApartmentsController">
    <aura:registerEvent name="formsubmit" type="c:FormSubmit"/>
    <aura:attribute name="options" type="String[]" />
    <aura:attribute name="optionsBB" type="String[]" />
    <aura:attribute name="numOfRooms" type="Integer" />
    <aura:attribute name="fromDate" type="Date" />
    <aura:attribute name="toDate" type="Date" />
    <aura:attribute name="street" type="String" />
    <aura:attribute name="city" type="String" />
    <aura:attribute name="val" type="Integer"/>
    <aura:attribute name="condToggle" type="Boolean"
default="false"/>
    <aura:attribute name="washingMachineToggle" type="Boolean"
default="false"/>
    <aura:attribute name="wifiToggle" type="Boolean"
default="false"/>
    <aura:attribute name="elevToggle" type="Boolean"
default="false"/>
    <aura:attribute name="renderNew" type="Boolean"
default="true" />
    <aura:attribute name="selectedValue" type="String"
default="" />
    <aura:attribute name="selectedBBValue" type="String"
default="" />
    <aura:handler name="init" value="{!this}"
action="{!c.doinit}" />
    <div class="c-container">
        <lightning:layout horizontalAlign="space">
            <lightning:layoutItem padding="around-small">
                <lightning:layoutItem padding="around-small">
                <div class="custom-box">
                    <lightning:select aura:id="boatStatus"
name="boatStatus" label="" value="{!v.selectedValue}" >
                        <option value=""> Всі типи нерухомості
</option>
                        <aura:iteration items="{!v.options}"
var="item">
                            <option value="{!item}"
>{!item}</option>
                        </aura:iteration>
                    </lightning:select>
                </div>
```

```
                </lightning:layoutItem>
                <lightning:layoutItem padding="around-small">
                    <div class="custom-box">
                        <lightning:input type="toggle"
name="toggleButton" aura:id="tglbtn" label="Кондиціонер"

messageToggleActive="Є!" messageToggleInactive="Немає!"

onchange="{!c.getToggleButtonValueCond}"/>
                        <br/>
                        <lightning:input type="toggle"
label="Пральна машина" name="inputToggle1"
                                         aura:id="tglbtn1"
messageToggleActive="Є!" messageToggleInactive="Немає!"

onchange="{!c.getToggleButtonValueWash}" />
                    </div>
                </lightning:layoutItem>
            </lightning:layoutItem>
            <lightning:layoutItem padding="around-small">
                <lightning:layoutItem padding="around-small">
                    <div class="custom-box">
                        <lightning:select aura:id="boatBBStatus"
name="boatStatus" label="" value="{!v.selectedBBValue}" >
                            <option value="">  Базові умови
проживання </option>
                            <aura:iteration
items="{!v.optionsBB}" var="item">
                                <option value="{!item}"
>{!item}</option>
                            </aura:iteration>
                        </lightning:select>
                    </div>
                </lightning:layoutItem>
                <lightning:layoutItem padding="around-small">
                    <div class="custom-box">
                        <lightning:input type="toggle"
label="Wi-fi" name="inputToggle2"
                                         aura:id="tglbtn2"
messageToggleActive="Є!" messageToggleInactive="Немає!"

onchange="{!c.getToggleButtonValueWifi}"
                                         />
                        <br/>
                        <lightning:input type="toggle"
label="Ліфт" name="inputToggle3"
                                         aura:id="tglbtn3"
messageToggleActive="Є!" messageToggleInactive="Немає!"

onchange="{!c.getToggleButtonValueElev}"
                                         />
                    </div>
                </lightning:layoutItem>
```

```
            </lightning:layoutItem>

            <lightning:layoutItem padding="around-small">
                <lightning:layoutItem padding="around-small">

<div class="custom-box">
  <lightning:input type="number" name="inputNumOfRooms"
placeholder = "Введіть кількість кімнат..." value =
"{!v.numOfRooms}" max="10" min = "1"/>
                    </div>
                </lightning:layoutItem>
                <lightning:layoutItem padding="around-small">
                    <lightning:slider label="Ціна"
value="{!v.val}" min = "1000" max="5000" />
                </lightning:layoutItem>
            </lightning:layoutItem>
            <lightning:layoutItem padding="around-small">
                <lightning:layoutItem padding="around-small">
                    <div class="custom-box">
                        <lightning:input name="inputcity"
value="{!v.city}" placeholder="Введіть місто..."/>
                        <lightning:input name="inputstreet"
value="{!v.street}" placeholder="введіть вулицю..."/>
                    </div>
                </lightning:layoutItem>
            </lightning:layoutItem>

            <aura:if isTrue="{!or(or( or(
or(not(empty(v.selectedValue)), not(empty(v.selectedBBValue))),
or(not(empty(v.numOfRooms)), not(empty(v.city))) ), or(
or(not(empty(v.street)), v.condToggle),
or(v.washingMachineToggle, v.wifiToggle) ) ),
or(v.elevToggle,not(empty(v.val)) ))}" >
                <lightning:layoutItem padding="around-small">
                    <div class="custom-box">
                        <lightning:button variant="brand"
label="Шукати" aura:id="SearchButton"
onclick="{!c.onFormSubmit}"
                                          />
                    </div>
                </lightning:layoutItem>
            </aura:if>
        </lightning:layout>
    </div>
</aura:component>

Controller.js
({
    doinit : function(component, event, helper) {
        var action =
component.get("c.getPickListValuesIntoList");
        var opts = [];
```

```
  action.setCallback(this, function(response) {
            if (response.getState() == "SUCCESS") {
                var allValues = response.getReturnValue();
                component.set("v.options", allValues);
            }
        });
        $A.enqueueAction(action);
        var action =
component.get("c.getPickListValuesIntoListBB");
        var opts = [];
        action.setCallback(this, function(response) {
            if (response.getState() == "SUCCESS") {
                var allValues = response.getReturnValue();
                component.set("v.optionsBB", allValues);
            }
        });
        $A.enqueueAction(action);
    },

    getToggleButtonValueCond : function(component,event,helper){
        var checkCmp =
component.find("tglbtn").get("v.checked");
        component.set("v.condToggle", checkCmp);
    },

    getToggleButtonValueWash : function(component,event,helper){
        var checkCmp =
component.find("tglbtn1").get("v.checked");
        component.set("v.washingMachineToggle", checkCmp);
    },

    getToggleButtonValueWifi : function(component,event,helper){
        var checkCmp =
component.find("tglbtn2").get("v.checked");
        component.set("v.wifiToggle", checkCmp);
    },

    getToggleButtonValueElev : function(component,event,helper){
        var checkCmp =
component.find("tglbtn3").get("v.checked");
        component.set("v.elevToggle", checkCmp);
    },

    onFormSubmit : function(component, event, helper){
        var boatTypeId = component.get("v.selectedValue");
        var boatTypeIdBB = component.get("v.selectedBBValue");

        var numberOfRooms = component.get("v.numOfRooms");
        var cityValue = component.get("v.city");
        var streetValue = component.get("v.street");
        var priceValue = component.get("v.val");
        var condToggle = component.get("v.condToggle");
        var washingMachineToggle =
```

```
component.get("v.washingMachineToggle");
        var wifiToggle = component.get("v.wifiToggle");
        var elevToggle = component.get("v.elevToggle");

        var formSubmit = component.getEvent("formsubmit");
        formSubmit.setParams({
            "formData": boatTypeId,
            "numOfRooms": numberOfRooms,
            "city": cityValue,
            "street": streetValue,
            "val": priceValue,
            "condToggle": condToggle,
            "washingMachineToggle": washingMachineToggle,
            "wifiToggle": wifiToggle,
            "elevToggle": elevToggle,
            "selectedBBValue": boatTypeIdBB
        });
        formSubmit.fire();
    },

    checkValidity : function(component, event, helper) {
        var inputCmp = component.find("inputFrom");
        var value = component.get("v.fromDate");
        let today = new Date().toISOString().slice(0, 10)
        if (value >= today || value === null) {
            inputCmp.setCustomValidity(""); // if there was a
custom error before, reset it
        } else {
            inputCmp.setCustomValidity("Будь-ласка введіть дату
більше чи рівну сьогоднішній");
        }
        inputCmp.reportValidity(); // Tells lightning:input to
show the error right away without needing interaction*/
    },
        checkValidityTo : function(component, event, helper) {
        var inputCmp = component.find("inputTo");
        var value = component.get("v.toDate");

       var valueFrom = component.get("v.fromDate");
        let today = new Date().toISOString().slice(0, 10)
        if (value > today || value === null) {
            inputCmp.setCustomValidity(""); // if there was a
custom error before, reset it

} else {
            inputCmp.setCustomValidity("Будь-ласка введіть дату
більше чи рівну сьогоднішній");
        }
        inputCmp.reportValidity(); // Tells lightning:input to
show the error right away without needing interaction*/
    },
})
```

```
FindApartmentsController.cls
public class FindApartmentsController {
    public class WrapperBookingContractLines {
        public Contract_Line__c wrContractLine {get; set;}
        public Booking__c wrBooking {get;set;}
        public Integer daysOfBooking {get;set;}
        Integer num {get;set;}
        public WrapperBookingContractLines(){
            wrContractLine = new Contract_Line__c();
            wrBooking = new Booking__c();
            num = 0;
            daysOfBooking = 0;
        }
    }

    public class WrapperForCapacity{
        Integer Number_of_Childs { get; set; }
        Integer Number_of_Babies { get; set; }
        Integer Number_of_Adults { get; set; }
        Product2 room { get; set; }

        public WrapperForCapacity(){
            room = new Product2();
            Number_of_Childs = 0;
            Number_of_Babies = 0;
            Number_of_Adults = 0;
        }    }

    @AuraEnabled
    public static List<String> getPickListValuesIntoList(){
        List<String> pickListValuesList = new List<String>();
        Schema.DescribeFieldResult fieldResult =
Product2.Apartment_Type__c.getDescribe();

         List<Schema.PicklistEntry> ple =
fieldResult.getPicklistValues();
        for( Schema.PicklistEntry pickListVal : ple){
            pickListValuesList.add(pickListVal.getLabel());
        }
        return pickListValuesList;
    }

    @AuraEnabled
    public static String searchForContractDates(Date fromDate,
Date toDate, Integer numberOfRooms, string ServiceTypeId, string
boardBasisValues){
        List<String> pickListValuesList = new List<String>();

        Map <id, Contract_Line__c> contractLines = new Map<id,
Contract_Line__c>([Select id, Travel_Period_To__c,
Travel_Period_From__c, Board_Basis__c, Room_Type__c, Name,
Length_of_Stay_From__c, Length_of_Stay_to__c, Contract__c
                                                    from
```

```
Contract_Line__c where Board_Basis__c =: boardBasisValues and
Room_Type__c =: ServiceTypeId and (Travel_Period_From__c <=:
fromDate and Travel_Period_To__c >=: toDate)
                                                ORDER BY
Travel_Period_From__c]);
        List <Daily_Stock_data__c> Daily_Stock = [select
Date__c, Common_Available_Stock__c, Stock_Quantity__c,
Stock__c,Product__c, Contract__r.Board_Basis__c,
Booked_Stock_Qty__c,

Length_of_Stay_From__c, Length_of_Stay_To__c,
Contract__r.Contract__c, Contract__r.Travel_Period_To__c,

Contract__r.Travel_Period_From__c
                                                 from
Daily_Stock_data__c where (date__c in: datesBetween(fromDate,
toDate)) and Common_Available_Stock__c >=: numberOfRooms and
Contract__c in: contractLines.keyset() ];
        Set<Date> datesToCheck = new Set<Date>();
        datesToCheck = datesBetween(fromDate, toDate);
        Map<Id, Integer> mapToCountDays = new Map<Id,
Integer>();
        for(Daily_Stock_data__c item : Daily_Stock){

if(mapToCountDays.containsKey(item.Contract__r.Contract__c)){
                Integer counterForPrevious =
mapToCountDays.get(item.Contract__r.Contract__c);
                mapToCountDays.put(item.Contract__r.Contract__c,
counterForPrevious+1);

         }else{
                Integer counterForPrevious = 0;
                mapToCountDays.put(item.Contract__r.Contract__c,
counterForPrevious+1);
            }
        }
        Map<Id,Boolean> haveAnAppropriatecontract = new
Map<Id,Boolean>();
        for(Id itemOfMap : mapToCountDays.keySet()){
            if(mapToCountDays.get(itemOfMap) ==
datesToCheck.size()){
                haveAnAppropriatecontract.put(itemOfMap, true);
            }else{
                haveAnAppropriatecontract.put(itemOfMap, false);
            }
        }
        String finalcheck = 'false';
        for(Id itemToCheck :
haveAnAppropriatecontract.keyset()){
            if(haveAnAppropriatecontract.get(itemToCheck) ==
true){
                finalcheck = 'true-' + itemToCheck;
            }
```

```
        }
        return finalcheck;
    }

    @AuraEnabled
    public static List<String> getPickListValuesIntoListBB(){
        List<String> pickListValuesList = new List<String>();
        Schema.DescribeFieldResult fieldResult =
Product2.Board_Basis__c.getDescribe();
        List<Schema.PicklistEntry> ple =
fieldResult.getPicklistValues();
        for( Schema.PicklistEntry pickListVal : ple){
            pickListValuesList.add(pickListVal.getLabel());
        }
        return pickListValuesList;
    }

    @AuraEnabled
    public static List<Product2> getApps(String boatTypeId,
String BBTypeId, Integer amountOfRooms, String street, String
city, Integer val, Boolean condToggle, Boolean
washingMachineToggle, Boolean wifiToggle, Boolean elevToggle){
        List<Product2> apartments = new List<Product2>();
        String whereClause = 'Select Id,Name,PictureID__c from
Product2 WHERE ';
        String whereClause1 = '';
        String whereClause2 = '';

        String whereClause3 = '';
        String whereClause4 = '';
        String whereClause5 = '';
        String whereClause6 = '';
        String whereClause7 = '';
        String whereClause8 = '';
        String whereClause9 = '';
        String whereClause10 = '';
        String whereClause11 = '';

        List<String> myListForQuery = new List<String>();
        if(!(String.isEmpty(boatTypeId))){
            whereClause1 = ' Type_of_apartment__c =: boatTypeId'
;
            myListForQuery.add(whereClause1);
        }
        if(!(String.isEmpty(BBTypeId))){
            whereClause2 = ' Board_Basis__c =: BBTypeId';
            myListForQuery.add(whereClause2);
        }
        if(amountOfRooms > 0){
            whereClause3 = ' Rooms_Amount__c =: amountOfRooms' ;
            myListForQuery.add(whereClause3);
        }
```

```
        if(!(String.isEmpty(street))){
            whereClause4 = ' Street__c LIKE \'%' + street + '%\'
';
            myListForQuery.add(whereClause4);
        }

        if(!(String.isEmpty(city))){
            whereClause5 = ' City__c LIKE \'%' + city + '%\' ';
            myListForQuery.add(whereClause5);
        }

        if(condToggle){
            whereClause7 = ' conditioner__c =: condToggle ';
            myListForQuery.add(whereClause7);
        }
        Integer priceBig = 0;
        Integer priceLess = 0;
        if(val > 0){

            priceBig = val + 50;
            priceLess = val - 50;
            whereClause8 = ' (PriceBase__c >=: priceLess OR
PriceBase__c <=: priceBig ) ';
            myListForQuery.add(whereClause8);
        }
        if(washingMachineToggle){
            whereClause9 = ' WashingMachine__c =:
washingMachineToggle ';
            myListForQuery.add(whereClause9);
        }
        if(wifiToggle){
            whereClause10 = ' WiFi__c =: wifiToggle ';
            myListForQuery.add(whereClause10);
        }
        if(elevToggle){
            whereClause11 = ' Pets__c =: elevToggle ';
            myListForQuery.add(whereClause11);
        }
        for(integer item = 0 ; item < myListForQuery.size();
item++){
            if(item == 0){
               whereClause += myListForQuery[item];
            }else{
                whereClause += ' AND ' + myListForQuery[item];
            }
        }
        apartments = Database.query(whereClause);
        return apartments;
    }

    @AuraEnabled
    public static String createBooking(String boardBasis, Double
bookingAmount, Integer numberOfAdults, Integer numberOfBabies,
```

```
Integer numberOfChilds, Integer numberOfRooms, String
paxFirstName, String paxLastName, String paxEmail,String
paxPhoneNumber, Date checkInDate, Date checkOutDate, String
Productinfo, String additionalInfo, String contractId){
        system.debug('Booking info boardBasis ' + boardBasis);
        system.debug('Booking info bookingAmount ' +
bookingAmount);
        system.debug('Booking info numberOfAdults ' +
numberOfAdults);
        system.debug('Booking info numberOfBabies ' +
numberOfBabies);
        system.debug('Booking info numberOfChilds ' +
numberOfChilds);
        system.debug('Booking info numberOfRooms ' +
numberOfRooms);
        system.debug('Booking info paxFirstName ' +
paxFirstName);
        system.debug('Booking info paxLastName ' + paxLastName);

        system.debug('Booking info paxPhoneNumber ' +
paxPhoneNumber);
        system.debug('Booking info checkInDate ' + checkInDate);
        system.debug('Booking info checkOutDate ' +
checkOutDate);
        system.debug('Booking info Productinfo ' + Productinfo);
        system.debug('Booking info additionalInfo ' +
additionalInfo);
        Booking__c bookingRecord = new Booking__c();
        bookingRecord.Board_Basis__c = boardBasis;
        bookingRecord.Number_of_Adults__c = numberOfAdults;
        bookingRecord.Number_of_Babies__c = numberOfBabies;
        bookingRecord.Number_of_Childs__c = numberOfChilds;
        bookingRecord.Number_of_Rooms__c = numberOfRooms;
        bookingRecord.Pax_First_Name__c = paxFirstName;
        bookingRecord.Pax_Last_Name__c = paxLastName;
        bookingRecord.Pax_Email__c = paxEmail;
        bookingRecord.Pax_Phone_Number__c = paxPhoneNumber;
        bookingRecord.Booking_Check_in_Date__c = checkInDate;
        bookingRecord.Booking_Check_out_Date__c = checkOutDate;
        bookingRecord.Product__c = Productinfo;
        bookingRecord.Additional_information__c =
additionalInfo;
        bookingRecord.Contract__c = contractId;
        List<Booking__c> bkngList = new List<Booking__c>();
        List<Contract> myContracts = new List<Contract>([select
id, AccountId from Contract where id =: contractId ]);
        bookingRecord.Account__c = myContracts[0].AccountId;
        bkngList.add(bookingRecord);
        insert bkngList;
        Contact cont = new Contact();
        cont.Email = paxEmail;
        cont.FirstName = paxFirstName;
        cont.LastName = paxLastName;
```

```
        cont.AccountId = myContracts[0].AccountID;
        cont.BookingIdHelpField__c = bkngList[0].Id;
        insert cont;
        String stringToReturn;
        FindingTheStock(BookingCapacitiesControl(bkngList));
        Booking__c bkng = [select id, booking_status__c,
Booking_Capacity_Status__c, Check_in_out_Status__c from
booking__c where id =: bkngList[0].Id];
        if(bkng.booking_status__c == 'OK'){
            event e = new event();
            e.WhatId = bkng.Id;
            e.Subject = 'Booking';

            e.IsAllDayEvent = true;
            e.WhoId = cont.id;
            e.OwnerId = UserInfo.getUserId();
            e.StartDateTime=checkInDate;
            e.EndDateTime=checkOutDate;
            insert e;

            stringToReturn = bkngList[0].Id;
        }else if(bkng.Booking_Capacity_Status__c == 'KO
capacity' || bkng.Check_in_out_Status__c == 'KO check in/out' ||
bkng.booking_status__c == 'KO'){
            stringToReturn = 'KO';
            system.debug(bkng.Booking_Capacity_Status__c + ' ' +
bkng.Check_in_out_Status__c + ' ' + bkng.booking_status__c);
            delete bkng;
        }
        return stringToReturn;
    }

    public static void FindingTheStock(List<Booking__c>
bookings){
        WrapperBookingContractLines WrapperBookingContractLine =
new WrapperBookingContractLines();
        SET<Id> roomTypes = new SET<Id>();
        SET<Id> contractId = new SET<Id>();
        SET<Id> bookingsID = new SET<Id>();
        SET<ID> selectedStock = new SET<Id>();
        SET<String> boardBasisValues = new SET<String>();
        for(Booking__c bookingItem : bookings) {
            roomTypes.add(bookingItem.Product__c);
            contractId.add(bookingItem.Contract__c);
            bookingsID.add(bookingItem.id);
            boardBasisValues.add(bookingItem.Board_Basis__c);
        }
        List <Contract_Line__c> contractLines = [Select id,
Travel_Period_To__c, Travel_Period_From__c, Board_Basis__c,
Room_Type__c, Name, Length_of_Stay_From__c,
Length_of_Stay_to__c, Contract__c
                                                 from
Contract_Line__c where Contract__c in: contractId AND
```

```
Board_Basis__c in: boardBasisValues ORDER BY
Travel_Period_From__c];
        Set<String> boardBasisValuesToRemove = new
SET<String>();
        Set<String> serviceTypeValuesTocheck = new
SET<String>();
        for(Contract_Line__c cln : contractLines) {
            boardBasisValuesToRemove.add(cln.Board_Basis__c);

            serviceTypeValuesTocheck.add(cln.Room_Type__c);
        }
        Map<Id,Boolean> errorOfBB = new Map<Id,Boolean>();
        for(Booking__c bookingItem : bookings){
            errorOfBB.put(bookingItem.Id, false);

if(!boardBasisValuesToRemove.contains(bookingItem.Board_Basis__c
)){
                errorOfBB.put(bookingItem.Id, true);
                bookingItem.addError('No Contract Lines with
such Board Basis for this Booking. Please check all
parametrs.');
            }
        }
        for(Booking__c bookingItem : bookings){
            if(!errorOfBB.get(bookingItem.Id)){

if(!serviceTypeValuesTocheck.contains(bookingItem.Product__c)){
                    bookingItem.addError('No such Service Type
on the Contract. Please check all parametrs. ');
                }
            }
        }

        SET<Id> contractLinesId = new SET<Id>();
        for(Contract_Line__c clItem : contractLines) {
            contractLinesId.add(clItem.id);
        }

        List <Daily_Stock_data__c> Daily_Stock = [select
Date__c, Available_Stock__c, Stock_Quantity__c,
Stock__c,Product__c, Contract__r.Board_Basis__c,
Booked_Stock_Qty__c,

Length_of_Stay_From__c, Length_of_Stay_To__c,
Contract__r.Contract__c, Contract__r.Travel_Period_To__c,

Contract__r.Travel_Period_From__c
                                                from
Daily_Stock_data__c where (date__c in:
TravelPeriodDates(bookings)

and Product__c in: roomTypes and  Contract__c in:
contractLinesId)];
```

```
        List <Daily_Stock_data__c> Daily_Stock_Booked = [select
Date__c, Available_Stock__c, Stock_Quantity__c,
Stock__c,Product__c,
Booked_Stock_Qty__c,Contract__r.Board_Basis__c,

Length_of_Stay_From__c, Length_of_Stay_To__c,
Contract__r.Contract__c, Contract__r.Travel_Period_To__c,

Contract__r.Travel_Period_From__c

                                                                from
Daily_Stock_data__c
                                                                where
date__c in: TravelPeriodDates(bookings)
                                                                and
Product__c in: roomTypes and  Contract__c in: contractLinesId];
        List<Booking_Stock_Association__c>
Booking_Stock_Association = new
List<Booking_Stock_Association__c>();
        List<Booking_Stock_Association__c>
Booking_Stock_Association_Global = new
List<Booking_Stock_Association__c>();
        List <Daily_Stock_data__c>
List_To_Update_Booked_Daily_Stock = new List
<Daily_Stock_data__c>();
        SET<Date> selectedDates = new SET <Date>();
        for(Contract_Line__c uniCL : contractLines){
            integer i = 0;
            integer summa = 0;
            integer summaDays = 0;
            for(Booking__c booking : bookings) {
                Integer numberDaysBooking =
booking.Booking_Check_in_Date__c.daysBetween(booking.Booking_Che
ck_out_Date__c);
                summaDays =
booking.Booking_Check_in_Date__c.daysBetween(booking.Booking_Che
ck_out_Date__c);
                if((booking.Booking_Check_in_Date__c != null) &&
(booking.Booking_Check_out_Date__c != null)) {
                    for(integer noOfDays = 0; noOfDays <
booking.Booking_Check_in_Date__c.daysBetween(booking.Booking_Che
ck_out_Date__c); noOfDays++) {
                        if(booking.Booking_Check_in_Date__c <=
booking.Booking_Check_out_Date__c) {
                            if((uniCL.Travel_Period_From__c <=
booking.Booking_Check_in_Date__c.addDays(noOfDays))&&
                                (uniCL.Travel_Period_To__c >=
booking.Booking_Check_in_Date__c.addDays(noOfDays)) &&
(booking.Board_Basis__c == uniCL.Board_Basis__c )) {
                                    i = 1 + i;
                                }
                        }
```

```
                    } summa = i;
                }
                WrapperBookingContractLine.wrbooking = booking;
                WrapperBookingContractLine.daysOfBooking =
numberDaysBooking;
            }
            WrapperBookingContractLine.wrContractLine = uniCL;

            WrapperBookingContractLine.num = summaDays;
            if((WrapperBookingContractLine.num >=
WrapperBookingContractLine.wrContractLine.Length_of_Stay_From__c
)&&
               (WrapperBookingContractLine.num <=
WrapperBookingContractLine.wrContractLine.Length_of_Stay_To__c))
{
                    for(Daily_Stock_data__c dailyStock :
Daily_Stock){
                        Booking_Stock_Association__c
Booking_Stock_Association_Item = new
Booking_Stock_Association__c();
                        if(
WrapperBookingContractLine.wrContractLine.id ==
dailyStock.Contract__c){
                            if(dailyStock.Booked_Stock_Qty__c ==
null){
                                dailyStock.Booked_Stock_Qty__c =
0;
                            }

Booking_Stock_Association_Item.Daily_Prices_Stock_data__c =
dailyStock.Id;

Booking_Stock_Association_Item.Booking_Length_of_Periods_Stay__c
= WrapperBookingContractLine.num;

Booking_Stock_Association_Item.Booking__c =
WrapperBookingContractLine.wrbooking.id;

if((!(selectedDates.Contains(Booking_Stock_Association_Item.Date
__c))) &&

(Booking_Stock_Association_Item.Date__c !=
WrapperBookingContractLine.wrbooking.Booking_Check_out_Date__c))
{

Booking_Stock_Association.add(Booking_Stock_Association_Item);
                            }
                        }
                    }
                }
        }
        try{
            Map<Booking__c, List<Booking_Stock_Association__c>>
```

```
bookingWithItsAssosiations = new Map<Booking__c,
List<Booking_Stock_Association__c>>();
            for(Booking__c bkng : bookings){
                List<Booking_Stock_Association__c> assosForMap =
new List<Booking_Stock_Association__c>();
                for(Booking_Stock_Association__c bstckass :
Booking_Stock_Association){
                    if(bstckass.Booking__c == bkng.Id){
                        assosForMap.add(bstckass);
                    }
                }
               bookingWithItsAssosiations.put(bkng,assosForMap);

            }
            for(Booking__c bkng :
bookingWithItsAssosiations.keySet()){
                List<Booking_Stock_Association__c>
assosForCompare = bookingWithItsAssosiations.get(bkng);

if((bkng.Booking_Check_in_Date__c.daysbetween(bkng.Booking_Check
_Out_Date__c)) == assosForCompare.size()){

Booking_Stock_Association_Global.addAll(assosForCompare);
                }
            }
            insert Booking_Stock_Association_Global;
            System.Debug('Insert successful on Booking  Stock
Association');
        }catch (Exception e) {
            System.Debug('The following error occured' + e);
        }
    }

    public static Set<Date> TravelPeriodDates(List<Booking__c>
bookings){
        Set<Date> travelPeriodDates = new Set<Date>();
        for( Booking__c bookingItem : bookings){
            if((bookingItem.Booking_Check_in_Date__c != null) &&
(bookingItem.Booking_Check_Out_Date__c != null)){
                for(integer noOfDays = 0 ; noOfDays <
bookingItem.Booking_Check_in_Date__c.daysBetween(bookingItem.Boo
king_Check_Out_Date__c); noOfDays++){
                    if(bookingItem.Booking_Check_in_Date__c <
bookingItem.Booking_Check_Out_Date__c) {

travelPeriodDates.add(bookingItem.Booking_Check_in_Date__c.addDa
ys(noOfDays));
                    }
                }
            }
        } return travelPeriodDates;
    }
```

```
    public static Set<Date> datesBetween(Date fromValue, Date
toValue){
        Set<Date> travelPeriodDates = new Set<Date>();
        if((fromValue != null) && (toValue != null)){
            for(integer noOfDays = 0 ; noOfDays <
fromValue.daysBetween(toValue); noOfDays++){
                if(fromValue < toValue) {


travelPeriodDates.add(fromValue.addDays(noOfDays));
                }
            }
        } return travelPeriodDates;
    }

    Public static List<Booking__c>
BookingCapacitiesControl(List<Booking__c> TriggerNew){
        set<Id> relatedProducts = new set<Id>();
        for (Booking__c booking :TriggerNew){
            relatedProducts.add(booking.Product__c);
        }
        List<Product2> prdct = [select Board_Basis__c,X1A__c
,X1A_1B__c,  X1A_1C__c, X2A__c, X2A_1B__c,X2A_1C__c,
X2A_1C_1B__c,X2A_2B__c,X2A_2C__c,
                                X3A__c, X3A_1B__c,X3A_1C__c,
X3A_1C_1B__c, X3A_2B__c, X3A_2C__c, X4A__c from Product2 where
Id in: relatedProducts ];
        Integer Number_of_Childs;
        Integer Number_of_Babies;
        Integer Number_of_Adults;
        for(Booking__c booking : TriggerNew){
            Number_of_Childs = 0;
            Number_of_Babies = 0;
            Number_of_Adults = 0;
            WrapperForCapacity WrcapacityItem = new
WrapperForCapacity();
            List<WrapperForCapacity> WrcapacityList = new
List<WrapperForCapacity>();
            For(Product2 prdtc : prdct){
                if(booking.Product__c == prdtc.id){
                    if(prdtc.X1A__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 1;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X1A_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 1;
                        WrcapacityItem.Number_of_Childs = 0;
```

```
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }

                    if(prdtc.X1A_1C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 1;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }

                    if(prdtc.X2A__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A_1C_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1C_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X1A_1C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 1;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;

                        WrcapacityList.add(WrcapacityItem);
```

```
                    }
                    if(prdtc.X2A_1C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A_1C_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1C_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }

                    if(prdtc.X2A_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;

                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A_1C__c){
                        WrcapacityItem = new
```

```
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A_1C_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A_2B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 2;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X2A_2C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 2;
                        WrcapacityItem.Number_of_Childs = 2;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }

                    if(prdtc.X3A_2B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 2;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_2C__c){

                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 2;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 0;
```

```
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_1C_1B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 1;
                        WrcapacityItem.Number_of_Babies = 1;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X3A_2B__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 2;
                        WrcapacityList.add(WrcapacityItem);
                    }
        Закінчення дод. А

                    if(prdtc.X3A_2C__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 3;
                        WrcapacityItem.Number_of_Childs = 2;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
                    if(prdtc.X4A__c){
                        WrcapacityItem = new
WrapperForCapacity();
                        WrcapacityItem.Number_of_Adults = 4;
                        WrcapacityItem.Number_of_Childs = 0;
                        WrcapacityItem.Number_of_Babies = 0;
                        WrcapacityList.add(WrcapacityItem);
                    }
```

```
                }
                Boolean isBookingOK = false;
                for(WrapperForCapacity wrapperItem :
WrcapacityList){
                    if((wrapperItem.Number_of_Adults <
booking.Number_of_Adults__c)||
                       (wrapperItem.Number_of_Babies <
booking.Number_of_Babies__c)||
                       (wrapperItem.Number_of_Childs <
booking.Number_of_Childs__c )){
                       }
                    else {
                        isBookingOK = true;
                        break;
                    }
                }
                if(booking.Number_of_Rooms__c == 1) {
                    booking.Booking_Capacity_Status__c =
isBookingOK ? Label.OKCapacity : Label.KOCapacity;
                }else if(booking.Number_of_Rooms__c > 1){
                    booking.Booking_Capacity_Status__c =
Label.OKCapacity;
                }
            }
        }
        return TriggerNew;
    }
}
```