Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Терехов Дмитро Андрійович

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

Програмна система керування дій виконуючого персоналу баз відпочинку

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

Зіноватна Світлана Леонідівна

доц. кафедри системного програмного забезпечення

Одеса – 2020

ЗМІСТ

Завдання [на кваліфікаційну роботу 4](#_Toc58965573)

Завдання [на розробку розділу «Охорона праці та безпека в надзвичайних ситуаціях» 6](#_Toc58965576)

[АНОТАЦІЯ 7](#_Toc58965577)

[ВСТУП 9](#_Toc58965578)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОБЛЕМИ РОЗРОБКИ СИСТЕМ ВЗАЄМОДІЇ З КЛІЄНАМИ 12](#_Toc58965579)

[1.1 Теоретичні основи 12](#_Toc58965580)

[1.2 Аналіз існуючих проблем 13](#_Toc58965581)

[1.3 Критичний опис відомих рішень 14](#_Toc58965582)

[1.4 Висновки до розділу 17](#_Toc58965583)

[2 МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМИ ЗВ’ЗНОСТІ КОМПОНЕНТІВ СИСТЕМ ВЗАЄМОДІЇ З КЛІЄНТАМИ 18](#_Toc58965584)

[2.1 Опис децентралізованого підходу при розробці програмного забезпечення 18](#_Toc58965585)

[2.2 Дослідження сумісності мікросервісної та монолітної архітектур при розробці системи керування роботи з населенням 19](#_Toc58965586)

[2.3 Висновки до розділу 20](#_Toc58965587)

[3 АНАЛІЗ ВИМОГ ДО СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ 21](#_Toc58965588)

[3.1 Оцінка тривалості розробки 21](#_Toc58965589)

[3.2 Опис функціональних вимог до системи взаємодії з клієнтами 24](#_Toc58965590)

[3.3 Опис варіантів використання системи взаємодії з клієнтами 26](#_Toc58965591)

[3.4 Нефункціональні вимоги до програмного забезпечення 35](#_Toc58965592)

[3.5 Висновки до розділу 36](#_Toc58965593)

[4 ПРОЕКТУВАННЯ СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ 37](#_Toc58965594)

[4.1 Проектування структури та організації класів 37](#_Toc58965595)

[4.2 Проектування розподіленої реляційної бази даних 49](#_Toc58965596)

[4.3 Висновки до розділу 54](#_Toc58965597)

[5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ 55](#_Toc58965598)

[5.1 Набір інструментальних засобів розробки 55](#_Toc58965599)

[5.2 Алгоритми формування замовлення послуг по розміщенню 57](#_Toc58965600)

[5.3 User-інтерфейс на прикладі «сітки» відображення замовлень клієнтів 59](#_Toc58965603)

[5.4 Висновки до розділу 63](#_Toc58965604)

[6 ТЕСТУВАННЯ СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ 64](#_Toc58965605)

[6.1 Тестування програмних модулів 64](#_Toc58965606)

[6.2 Юзабіліті тестування 66](#_Toc58965607)

[6.3 Тестування сумісності 67](#_Toc58965608)

[6.4 Висновки до розділу 68](#_Toc58965609)

[7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 69](#_Toc58965610)

[7.1 Основні положення 69](#_Toc58965611)

[7.2 Висновки до розділу 70](#_Toc58965612)

[ВИСНОВКИ 71](#_Toc58965613)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 72](#_Toc58965614)

[ДОДАТОК А ЛІСТИНГ ПРОГРАМИ 74](#_Toc58965615)

[ДОДАТОК Б ПИТАННЯ З ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 97](#_Toc58965616)

[Б.1 Аналіз умов праці згідно індивідуального завдання 97](#_Toc58965617)

[Б.2 Індивідуальне завдання з охорони праці. Розрахунок заземлення 98](#_Toc58965618)

[Б.3 Безпека у надзвичайних ситуаціях техногенного характеру. Індивідуальне завдання. Розрахунок безпеки хімічного зараження 101](#_Toc58965619)

[Б.4 Безпека у суспільстві в умовах загрози COVID-19 103](#_Toc58965620)

# 

Міністерство освіти і науки України  
Одеський національний політехнічний університет  
Навчально-науковий інститут комп’ютерних систем  
Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)  
Спеціальність: 121 – Інженерія програмного забезпечення  
Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

проф. Крісілов В.А.

" " 2020 р.

**З А В Д А Н Н Я**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Терехов Дмитро Андрійович

1. Тема роботи: Програмна система керування дій виконуючого персоналу баз відпочинку

Керівник роботи: Зіноватна Світлана Леонідівна, доцент кафедри Системного програмного забезпечення, затверджені наказом ректора ОНПУ від “29” жовтня 2020 року №412-в

2. Строк подання студентом роботи 10.12.2020

3. Вихідні дані по проекту (роботі) Технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належіть розробити) Вступ. Аналіз предметної області та проблеми розробки систем взаємодії з клієнами. Методи вирішення проблеми зв’зності компонентів систем взаємодії з клієнтами. Аналіз вимог до системи взаємодії з клієнтами. Проектування системи взаємодії з клієнтами. Програмна реалізація системи взаємодії з клієнтами. Тестування системи взаємодії з клієнтами. Охорона праці та безпека у надзвичайних ситуаціях

5. Перелік ілюстративного матеріалу Мета та задачі роботи. Порівняння функціональних характеристик продуктів-аналогів. Діаграма варіантів використання. Реляційна модель даних. Приклади віконних форм програми. Висновки

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Назва  розділу | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| ОПтаБНС | Москалюк А.Ю., доц. | 24.11.2020 | 14.12.2020 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 01.09.2020

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Строк виконання  етапів роботи | Примітка |
| 1 | Узгодження теми з керівником роботи | 18.09.2020 | Вик. |
| 2 | Створення архітектури програмного забезпечення | 25.10.2020 | Вик. |
| 3 | Розгляд питань охорони праці безпеки у надзвичайних ситуаціях | 23.11.2020 | Вик. |
| 4 | Розробка програмного забезпечення | 01.12.2020 | Вик. |
| 5 | Перевірка роботи на відповідність вимогам | 09.12.2020 | Вик. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент** **\_\_\_\_\_\_\_\_\_\_\_\_\_** Терехов Д. А.

( підпис ) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_ Зіноватна С. Л.

( підпис ) (прізвище та ініціали)

# **ЗАВДАННЯ**

# на розробку розділу «Охорона праці та безпека в надзвичайних ситуаціях»

здобувач Терехов Дмитро Андрійович

Інститут комп’ютерних систем

Кафедра Системного програмного забезпечення

Тема дипломного проекту Програмна система керування дій виконуючого персоналу баз відпочинку

Тема розділу: Аналіз умов праці та безпеки при загрозі хімічного зараження

Зміст розділу:

1. Охорона праці.

2. Безпека у надзвичайних ситуаціях техногенного характеру.

3. Безпека у суспільстві в умовах загрози COVID-19.

|  |  |
| --- | --- |
| Керівник роботи | Консультант з охорони праці та безпеки в надзвичайних ситуаціях |
| \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ( підпис ) (прізвище та ініціали)  « \_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_ р. | \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ( підпис ) (прізвище та ініціали)  « \_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_ р. |

# АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 76с., 31 рис., 2 табл., 2 додатки, 14 джерел.

Метою даної роботи є зменшення часу взаємодії з клієнтами, а також між представниками різних структурних підрозділів бази відпочинку за рахунок автоматизації робочих процесів бази відпочинку.

Методи розробки базуються на технології RESTful-сервісів, JSON-форматі, а також на використанні технології обробки даних в оперативній пам’яті пристрою, реалізованої в системі управління базою даних H2.

Як результат роботи виконана програмна реалізація системи обліку та передачі даних для курортного комплексу, зокрема, для бази відпочинку «Лада».

Ключові  слова:  система взаємодії з клієнтами, курортний комплекс,  web-сервіс,  JSON-формат,  H2, Java Spring Framework, Firebase.

SUMMARY

Explanatory note to the diploma work: 76p., 31 fig., 2 tables, 2 additional texts, 14 sources.

The purpose of this work is to reduce the time of interaction with customers and between representatives of various structural units of the recreation center by automating the work processes of the recreation center.

Development methods are based on RESTful-services technology, JSON-format and H2 database server.

As a result, the software implementation of the electronic accounting system of the booking process for the resort complex, in particular, for the recreation center "Lada".

Keywords: customer interaction system, resort complex, web-service, JSON-format, H2, Java Spring Framework, Firebase.

# **ВСТУП**

База відпочинку – складний комплекс неоднорідних компонентів, взаємодіючих між собою в процесі проведення підприємницької діяльності. Зачасту такими компонентами виступають складні за своєю внутрішню структурою елементи взаємодії з клієнтами, такі як корпус розміщення клієнтів, зона загальнолюдського харчування та продажу споживчих товарів, розважальна зона та адміністраторський корпус. І при цьому усі існуючі на території бази відпочинку обов’язково мають на меті задоволення потреб клієнта у відпочинку та розвагах.

При цьому кожен підприємець хоче заробляти більше, якщо порівнювати з минулим роком, місяцем, тижнем, а для цього необхідно дотримуватися деяких принципів: швидке та точне обслуговування клієнтів, встановлення та модернізація елементів, наприклад, розміщення нового місця для харчування, а також підтримка робочого стану бази відпочинку.

Так, автоматизація роботи з клієнтами при бронюванні номерів, або при замовлені товарів у магазині на даний момент є більше необхідністю ніж можливістю, якщо власник бази відпочинку хоче бути успішним в цьому бізнесі. І на ринку програмного забезпечення існує багато різних засобів автоматизації роботи компонентів бази відпочинку. Проте існує проблема. З ростом обсягів бази розміри даних для обробки буде збільшуватись, як і зв’язність різних компонентів. Також з ростом обсягів існує проблема несумісності існуючих засобів автоматизації з новими вимогами власника.

Необхідне таке рішення, яке поєднує систему автоматизації роботи з клієнтами для всіх важливих елементів існування курортного комплексу з потребами в швидкості обробки даних, зменшення затрат при модернізації під необхідні конкретному підприємцю потреби. Так, метою даної роботи є зменшення часу взаємодії з клієнтами а також між представниками різних структурних підрозділів бази відпочинку за рахунок автоматизації робочих процесів бази відпочинку.

Об'єктом дослідження є системи взаємодії з клієнтами.

Предметом дослідження є методології розробки архітектури систем взаємодії.

Предметна область розробки – курортний комплекс.

Актуальність: велика зв’язність структурних компонентів курортного комплексу при великих масштабах виробництва.

Вирішувані завдання:

1. розробка метода синхронізації даних різних користувачів одного структурного компоненту комплексу;
2. виправлення моделі централізованого доступу до даних;
3. розробка програмної системи для реалізації методу синхронізації.

Методи дослідження. Для досягнення поставленої мети використані теорія множин для опису моделі бази даних відповідних компонентів а також теорія алгоритмів для розробки програмного продукту.

Науково-практична цінність полягає в тому, що автоматизовано процес передачі та обліку даних по внутрішній мережі.

В розділі «Аналіз предметної області та проблеми розробки систем взаємодії з клієнтами» описані основні структурні компоненти предметної області – курортного комплексу, дані термінологічні роз’яснення для важливих в рамка проведеної роботи понять, та описані існуючі на ринку застосування для реалізації проблем.

В розділі «Методи вирішення проблеми зв’язності компонентів систем взаємодії з клієнтами» розглянуто, як децентралізований підхід дозволяє реалізувати необхідні потреби замовників. Також було проведено питання об’єднання монолітної та мікросервісної архітектури.

В розділі «Аналіз вимог до системи взаємодії з клієнтами» описаний план виконання розробки з урахуванням строків виконання, а також визначено, які актори приймають участь в роботі комплексів, які функції кожен з них виконує, а також описані сценарії використання, за якими мають діяти актори.

В розділі «Проектування системи взаємодії з клієнтами» розглянуто, які архітектурні рішення необхідно запровадити при існуючій децентралізованій мережі. Архітектура системи описана за допомогою діаграм послідовності та класів. Також розглянуто, як виглядають бази даних для різних структурних підрозділів комплексу.

В розділі «Програмна реалізація системи взаємодії з клієнтами» було розглянуто питання взаємодії клієнтської та серверної частини застосування на прикладі процесу бронювання номеру для нового клієнта. Також описано набір робочих інструментів, що використовувались при розробці.

В розділі «Тестування системи взаємодії з клієнтами» проведені процеси модульного тестування за допомогою JUnit, тестування сумісності та тестування зручності використання застосування, а також проведений експеримент, в якому порівнювалась швидкість взаємодії користувача з клієнтом при використанні розробленого програмного забезпечення в порівнянні з ручною обробкою даних.

# **1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОБЛЕМИ РОЗРОБКИ СИСТЕМ ВЗАЄМОДІЇ З КЛІЄНАМИ**

## **1.1 Теоретичні основи**

Система управління взаємовідносинами з клієнтами (CRM) [5] – модель взаємодії, заснована на теорії, що центром всієї філософії бізнесу є клієнт, а головними напрямками діяльності компанії є заходи щодо забезпечення ефективного маркетингу, продажів і обслуговування клієнтів. Підтримка цих бізнес-цілей включає збір, зберігання і аналіз інформації про споживачів, постачальників, партнерів, а також про внутрішні процеси компанії. Функції для підтримки цих бізнес-цілей включають продажу, маркетинг, підтримку споживачів. CRM-система може включати:

* фронтальну частину, що забезпечує обслуговування клієнтів на точках продажів з автономною, розподіленою або централізованою обробкою інформації;
* операційну частину, що забезпечує авторизацію операцій і оперативну звітність;
* сховище даних;
* аналітичну підсистему;
* розподілену систему підтримки продажів: репліки даних на точках продажів або смарт-карти.

CRM-системи класифікують:

* за призначенням існує автоматизована система управління продажами (SFA), управління маркетингом, управління клієнтським обслуговуванням і колл-центрами;
* за рівнем обробки інформації: операційна CRM - реєстрація та оперативний доступ до первинної інформації щодо подій, компаніям, проектам, контактам, аналітична CRM - звітність і аналіз інформації в різних розрізах, коллаборативна CRM – рівень організації тісної взаємодії з кінцевими споживачами, клієнтами, аж до впливу клієнта на внутрішні процеси компанії.

Розподілена база даних – набір логічно пов'язаних між собою поділюваних даних, які розподілені в фізичному просторі. Система управління розподіленою базою даних складається з набору вузлів прийому запитів і набору вузлів збереження даних. Вузли прийому запитів реалізують прозорий інтерфейс доступу до даних, що зберігаються в вузлах збереження даних, та приховують фрагментацію даних між вузлами. Кожен з вузлів може бути представлений незалежним комп'ютером в комп'ютерній мережі з ​​власною операційною системою.

Мікросервісна архітектура – варіант архітектури програмного забезпечення, яка спрямована на взаємодію наскільки це можливо невеликих, слабо пов'язаних і легко змінюваних модулів – мікросервісів [2]. В мікросервісній архітектурі системи шикуються з компонентів, що виконують елементарні функції, і взаємодіючі з використанням економічних мережевих комунікаційних протоколів. За рахунок підвищення розбиття модулів архітектура націлена на зменшення ступеня зачеплення і збільшення зв'язності, що дозволяє простіше додавати і змінювати функції в системі в будь-який час.

## **1.2 Аналіз існуючих проблем**

Для розуміння специфіки розробки програмного забезпечення для курортного комплексу, необхідно зазначити важливі проблеми. Частіше за все, дані для одного структурного компоненту курортного комплексу розташовані в одному місці-сервері, до якого надходять запити і повертаються відповіді. Також дуже часто такі серверні пристрої для різних компонентів не поєднані між собою, тобто є ізольованими одна від іншої. Передача даних між ізольованими компонентами відбувається за рахунок підключення сторонніх користувачів. Для розгляду такого типу взаємодії скористаємось рисунком 1.1.

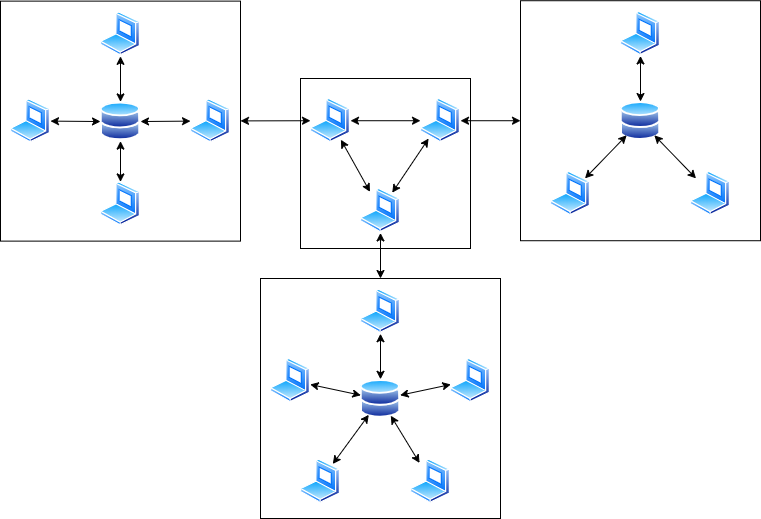


Рисунок 1.1 – Модель взаємодії даних в програмному середовищі курортного комплексу

## **1.3 Критичний опис відомих рішень**

Для опису рішень автоматизації праці службового персоналу, що існують на ринку програмних продуктів. Для розгляду обрано три найбільш відомих з таких рішень – Chameleon POS [10], EasyMS [11] та OtelMS [13]. Для опису їх переваг скористаємось інформацією з офіційних сайтів програмних продуктів.

Chameleon POS – система автоматизованого робочого місця касира, оператора, товарознавця, яке має досить функцій і можливостей, що поєднує в собі простоту в використанні разом з широкими можливостями та зручним інтерфейсом.

Переваги ПЗ:

* робота на ОС Linux і Android, які не потребують ліцензування;
* отримання даних про продажі, залишках в режимі ONLINE;
* кількість касових місць — без обмеження;
* стабільна робота навіть без наявності мережі;
* широкий спектр підтримуваного устаткування;
* висока швидкість організації внутрішнього електронного документообігу;
* одночасний обмін даними між точками та офісом;
* інтуїтивно зрозумілий інтерфейс;
* повна сумісність з будь-якими бухгалтерськими та обліковими системами;
* проведення переобліку, контрольних інвентаризацій без зупинки торгового процесу.

EasyMS – система хмарного менеджменту готельного бізнесу. Спростить життя отельера і заощадить час.

Переваги ПЗ:

* простота і легкість у використанні при наявності широкого функціоналу (менеджер каналів, модуль, інтернет-еквайринг, інтеграція з електронними замками і т.д.);
* доступна швидка технічна підтримка;
* перехід з будь-якої іншої програми за 20 хв. Вам нічого не потрібно робити, все зробимо ми;
* більше 20-ти найбільш ефективних онлайн-каналів. Серед них Booking.com, Airbnb, Hotels24.ua, Expedia і інші;
* вигідний інтернет-еквайринг (оплата для клієнта на вашому сайті) всього за 2,5%;
* безпечне шифрування даних;
* широкий спектр аналітики бронювань, фінансів і т. д.;
* Краща ціна на ринку і перший місяць роботи безкоштовно.

OtelMS – модуль онлайн бронювання для готелів, який задовольнить запити найвибагливіших гостей. Має можливість розмістити кнопку “забронювати” на своєму сайті та приймати замовлення напряму, а також має можливість встановити систему без якихось доробок свого сайту, взаємодіяти з будь-якими сервісами бронювання та примножувати свої продажі.

Для встановлення модулю знадобиться Система управління готелем HMS OtelMS. Доступна 30-денна безкоштовна демо-версія з повним набором функцій, яка позволить на практиці оцінити всі зручності модулю бронювання.

Переваги:

* синхронізація з системою управління готелем та каналами бронювання;
* продаж всіх і лише вільних номерів;
* можливість встановити акції, промокоди;
* 0% комісії за роботу модулю;
* просто й швидко встановлюється на будь-який сайт, Fb, ВК;
* працює на будь-якому пристрої з доступом в інтернет;
* постійно оновлюється.

Для кращого розуміння необхідних функціональних вимог до розробляємого програмного продукту визначені деякі параметри, за якими можна порівняти різні за призначенням системи керування роботи з клієнтами. Всі ці параметри оформлені у таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих програмних систем з розробленою системою

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Шифрування даних | Встановлення акції, промокодів | Сумісність з обліковими системами | Наявність демо-версії |
| Chameleon POS | - | + | + | - |

Продовження таблиці 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Шифрування даних | Встановлення акції, промокодів | Сумісність з обліковими системами | Наявність демо-версії |
| EasyMS | + | - | - | + |
| OtelMS | - | + | - | + |
| Розробка | + | - | + | + |

## **1.4 Висновки до розділу**

Описані основні структурні компоненти предметної області – курортного комплексу, дані термінологічні роз’яснення для важливих в рамках проведеної роботи понять, та описані існуючі на ринку застосування для реалізації проблем.

# **2 МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМИ ЗВ’ЗНОСТІ КОМПОНЕНТІВ СИСТЕМ ВЗАЄМОДІЇ З КЛІЄНТАМИ**

## **2.1 Опис децентралізованого підходу при розробці програмного забезпечення**

Модель централізованої обробки даних має деякі переваги над децентралізованою, в першу чергу тим, що дані для кожного користувача однієї мережі є однаковими, що спрощує вирішення завдання при синхронізації. Однак проблема в тому, що при виходу зі строю головного серверу доступ до даних пропадає у всіх користувачів, а його відновлення може займати багато часу.

Децентралізований підхід до будування моделі вирішує частину цих проблем. Так, пристрій користувача стає індивідуальною базою даних, яка зберігає всі дані, що існують на розгляді елементу курортного комплексу. При цьому, необхідно визначити метод, який дозволить оперувати різним користувачам однаковим набором даних. Модель такої взаємодії зображено на рис. 2.2.

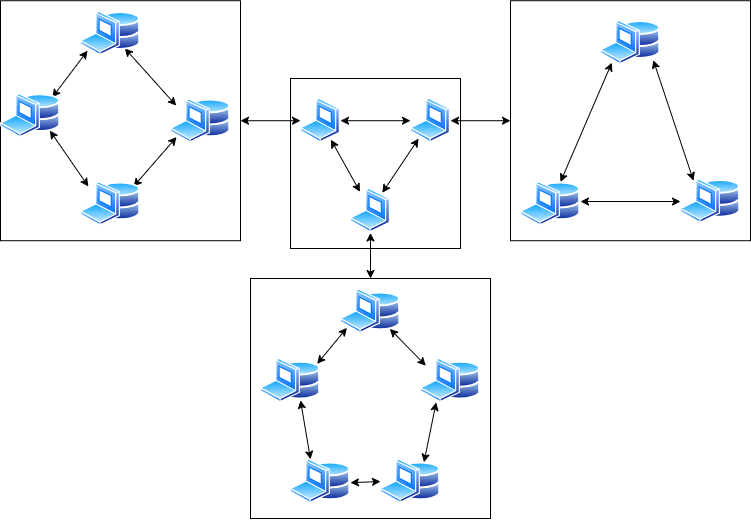


Рисунок 2.2 – Децентралізована модель взаємодії даних в програмному середовищі курортного комплексу

Тепер центральний елемент, хоча все ще є дуже сильно зв’язаним з іншими елементами системи комплексу, став більш стабільним в плані передачі інформації. Кожен пристрій стає самостійною базою даних, який поділяє однорідні дані між іншими елементами, тому, за умови виходу зі строю одного пристрою робота інших пристроїв зберігається в нормальному стані. Проте це несе за собою деякі проблеми, які необхідно вирішувати. Наприклад, ми можемо просто прийняти велику надмірність зберігаємих даних, або намагатись їх розподіляти серед усіма пристроям системи. І в тому і в іншому випадку все одно необхідно замислюватися над синхронізацією та маршутизацією даних.

## **2.2 Дослідження сумісності мікросервісної та монолітної архітектур при розробці системи керування роботи з населенням**

На даний момент при розробці архітектурі майбутнього програмного забезпечення використовують один з двох принципіально різних підходи – монолітний або мікросервісний. Кожен з цих підходів має свої переваги та недоліки.

Так, основним недоліком монолітної архітектури є велика зв’язність компонентів та відсутність ізоляції. Тобто, якщо необхідно буде додати нову функцію до системи, необхідно розуміти, як це вплине на роботу інших компонентів шляхом модульного та інтеграційного тестування. Це може вплинути на час розробки, а при виникненні помилки в будь-якому компоненті такого моноліту може спричинити уповільнення усього додатку. При цьому, монолітну архітекутру легше реалізувати та розгорнути на сервері.

В іншого боку, мікросервісна архітектура пропонує більш просте розуміння роботи її компонентів за рахунок їхнього розміру, відсутність необхідності прив’язання до конкретної технології та більш швидке розвертання та запуску незалежно один від іншого. Але мікросервісна архітектура не є ідеальним рішенням, такий підхід також має свої недоліки. Проведення транзакцій набагато легше проводиться при монолітній архітектурі. Також існують експлуатаційні витрати, а безліч мікросервісів складніше в експлуатації. Крім цього може знадобитися більше обладнання для підтримання роботи всієї системи.

Для потреб замовника-представника курортного комплексу жоден з цих підходів не відповідає в повній мірі для покриття усіх потреб. Монолітна архітектура не дозволяє швидко реагувати на збільшення обсягів, або необхідності введення нових функцій до системи, а мікросервісна – за рахунок необхідності підтримки великої кількості мікросервисів для якісної роботи усіх підрозділів. В такій ситуації можливе рішення – поєднання окремих концепцій двох архітектур.

Наприклад, курортний комплекс обов’язково має працівників розміщення та обробки запитів клієнтів та адміністративний корпус. Кожен з них матиме свою, відмінну одну від іншої систему, яка загалом працюватиме в одному просторі. Такі підсистеми можуть бути представлені мікросервісами, які можуть, наприклад, обробляти роботу бронювання номерів, обліку господарської діяльності, або передачі робочих даних серед працівників комплексу. При цьому, ідея об’єднання двох різних підходів не є невивченою. Так, Мартін Фаулер, один з основоположників ідеї мікросервісов, запропонував об'єднати обидва підходи в один [2].

## **2.3 Висновки до розділу**

Описані особливості моделі децентралізованого підходу обробки даних, розглянуто які є переваги та недоліки при використанні даної моделі, а також проаналізовано питання сумісності монолітного та мікросервісного підходу при вирішенні питання вибору архітектури системи.

# **3 АНАЛІЗ ВИМОГ ДО СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ**

## **3.1 Оцінка тривалості розробки**

Оцінити тривалість майбутньої розробки може бути достатньо складно, адже розробка – доволі складний процес, що складається з декількох етапів, тобто проектування, безпосередньо розробка, тестування та вивід в експлуатацію. Набагато легше оцінити, як довго буде проходити мала частина цієї роботи, наприклад, як довго буде проходити етап, розробки одного модулю системи. І при цьому ми можемо зрозуміти, які етапи ідуть після виконаного.

Так, оскільки дана розробка проводиться індивідуально, життєвий цикл буде дуже схожим з життєвим циклом «водоспад» [7]. Під час цього життєвого циклу розробка проходить наступним чином: першим етапом буде аналіз та збір вимог від замовника, та їхня документація; проектування майбутнього програмного застосування, для нашого випадку модулю бронювання; розробка та тестування забезпечення та перевірка на відповідність записаним вимогам; останній етап – вивід в експлуатацію та його підтримка.

Розробка ПО в рамках цієї моделі дозволяє строго зафіксувати бюджет і терміни. Однак, робота по цій моделі може бути ефективна тільки в тому випадку, якщо замовник детально розуміє цілі і завдання продукту, що розробляється, а також здатний їх сформулювати. Це обумовлено тим, що обсяг роботи теж фіксується - якщо щось не потрапило в ТЗ, то це швидше за все не буде реалізовано в рамках узгодженого бюджету та термінів.

Щоб максимально зменшити вплив цих проблем скористаємося одним з маніфестів гнучких методології розробки програмного забезпечення: «Співробітництво з замовником важливіша узгодження умов контракту». Для цієї розробки нашим замовником може вважатися керівник роботи, тому взаємодія буде проводитись саме з ним. В даному випадку, це передбачає, що необхідно підтримувати зв’язок з керівником для оцінки правильності та спроможності розробки на даному етапі. На основі цього, ми і будемо оцінювати тривалість розробки, а для візуалізації цього скористаємося діаграмою Ганта, що зображена у вигляді табл. 3.1-3.2:

Таблиця 3.1 – Діаграма Ганта для розробки модулю бронювання (частина 1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 01.09.2020 | | 07.09.2020 | | | 14.09.2020 | | | 21.09.2020 | | | 28.09.2020 | | | 05.10.2020 | | | 12.10.2020 | | |
| Визначення предметної області |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Визначення теми з керівником |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Вивчення ринку ПЗ |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Виявлення технік що використовуються найчастіше |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Пошук схожих рішень |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Виявлення унікальних компонент продукту |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Збір даних |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Визначеня особливостей предметної області |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Аналіз даних |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Проведення консультацій |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Виявлення необхідних для реалізації компонентів |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Підготовка до проектування |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Встановлення необхідних для реалізації компонентів |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Установка необхідних сервисів |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Створення документації |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Аналіз вимог |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Запис вимог |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |
| Визначення архітектури ПЗ |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  |

Таблиця 3.2 – Діаграма Ганта для розробки модулю бронювання (частина 2)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 19.10.2020 | | 26.10.2020 | | 02.11.2020 | | 09.11.2020 | | 16.11.2020 | | 23.11.2020 | | 30.11.2020 | |
| Створення архітектури ПЗ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Створення діаграми варіантів використання |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Визначення необхідних архітектурних рішень |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Створення діаграми класів |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Стоврення діаграм взаємодії |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Визначення тактик для вирішуваємих задач |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Аналіз архітектурних рішень |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Переробка існуючої архітектури |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Перевірка сумісності з архітектурою |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1-а ітерація розробки |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1-а ітерація тестування |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2-а ітерація розробки |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2-а ітерація тестування |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3-я ітерація розробки |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Розробка бази даних |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Тестування бази даних |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Тестування інтеграції бази даних з системою |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Підготовка до здачі |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Оформлення документу |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

З цієї діаграми можна побачити, що розробка за планом повинна розпочатись першого березня, йти 4 місяці,  або 18 тижнів. Найбільшими за тривалістю етапами розробки є етапи проектування та безпосередньо розробки застосування. Також можна помітити, що деякі етапи йдуть як паралельні. Для даної розробки паралельна розробка використовується в контексті виконання цих етапів невеликими проміжками з швидким переключенням між завданнями. Розглянемо детальніше найбільш важливі з них:

* визначення предметної області – визначення, що є основними компонентами області, як вони взаємодіють між собою, та що необхідно реалізувати для задоволення потребам користувачам;
* аналіз ринку програмного забезпечення – вивчення ринку на наявність існуючих застосувань, які покривають потреби замовника, з аналізом переваг кожного з них, а також вивчення основних практик, які використовуються для розробки програмного забезпечення;
* пошук унікальних особливостей продукту – визначення тих особливостей, які дозволяють відрізняти розроблене програмне забезпечення від існуючих аналогів, наприклад, більш стабільна робота системи;
* визначення архітектури ПЗ – визначення загальної структури системи з урахуванням потреб предметної області та потреб замовника, використовуючи існуючі архітектурні рішення;
* створення діаграм (класів, послідовності) – створення діаграми для розуміння процесу, який знаходиться у розробці;
* ітерації розробки – основна частина, проходить в три етапи; кожен з таких етапів повинен покрити значну частину розробки;
* ітерації тестування – проходить в три етапи, призваний відловити помилки програми, які допущені на етапі розробки програмного забезпечення;
* розробка бази даних – спеціальний етап розробки, в рамках якого відбувається створення структури бази даних та її інтеграції з існуючою системою.

## **3.2 Опис функціональних вимог до системи взаємодії з клієнтами**

Для предметної області курортного комплексу існує приблизно 15 основних функцій, які мають бути присутні в будь-якому курортному комплексі. Серед них є такі, як: прийняття замовлень на розміщення, організація, повідомлення про закупівлю, та інші. Ці функції розбиті між шістьма основними структурними елементами, серед яких: корпус розміщення та адміністративний корпус, зона продажу продуктів, харчування та розважальна зона, а також службове обслуговування (прокат, прибирання, ремонт, тощо). Так, розглядаючи відповідних акторів для відповідних структурних елементів курортного комплексу визначимо їх особливості, що відрізняють їх один від іншого:

* адміністратор корпусу розміщення несе відповідальність за прийняття замовлень від клієнтів на розміщення; інформування щодо правил, які діють на території комплексу, а також щодо зауважень; прийняття запитів від клієнтів т а їхня передача до відповідальних осіб;
* працівник зони харчування відповідальний за створення технологічних карт різноманітних страв; здійснення продажу страв; ідентифікацію та інвентаризацію майна;
* працівник зони продажу має повідомляти про переміщення товарів; повідомляти про закупівлю товарів; здійснювати продажу продуктів; проводити ідентифікацію та інвентаризацію майна;
* службовий працівник може приймати запити від клієнтів; повідомляти про факт завершення роботи; проводити ідентифікацію та інвентаризацію майна;
* адміністратор (директор) повинен організовувати та проводити контроль роботи робочого колективу; проводити облік та аналіз господарської діяльності;
* працівник розважальної зони відповідальний за повідомлення про нещасні випадки; здійснення продажу послуг; ідентифікацію та інвентаризацію майна.

Для розуміння того, які саме функції комплексу відповідають конкретному користувачеві, зобразимо це у вигляді діаграми (рис. 1.1).



Рисунок 3.1 – Варіанти використання для предметної області курортного комплексу

## **3.3 Опис варіантів використання системи взаємодії з клієнтами**

**Сценарій варіанту використання «Прийняття замовлення на розміщення»**

Передумова: клієнт перейшов до замовлення номеру.

Актори: працівник по розміщенню.

Основний актор: працівник по розміщенню.

Гарантія успіху: замовлення буде створене.

Основний сценарій:

1. Користувач обирає дату заїзду.

2. Користувач обирає дату виїзду.

3. Система перевіряє, чи іде дата заїзду перед датою виїзду.

4. Користувач обирає номер, до якого він буде заселятися.

5. Система перевіряє, чи вільній номер в даний період часу.

6. Користувач вводить своє ім’я, e-mail та телефон.

7. Система перевіряє наявність збігів у даних по іменам та e-mail.

8. Система створює новий запис про замовлення номеру, та додає до нього дату заїзду, дату виїзду, індекс номеру, ім’я користувача, email та телефон. Після цього, система зберігає запис у базі даних, та підтверджує створення.

Альтернативний сценарій:

3.а. Дата заїзду йде після датою виїзду.

3.а.1. Система попереджує про це користувача.

3.а.2. Користувач переходить до пункту 1.

5.а. Номер в даний період часу вже зайнятий.

5.а.1. Система попереджує про це користувача.

5.а.2. Користувач переходить до пункту 4.

**Сценарій варіанту використання «Інформування щодо правил та зауважень»**

Передумова: працівник авторизований у системі взаємодії з клієнтами.

Актори: працівник по розміщенню.

Основний актор: працівник по розміщенню.

Гарантія успіху: пристрої клієнтів знаходяться в робочому стані.

Основний сценарій:

1. Користувач вводить текст, який хоче розповсюдити.

2. Користувач обирає житлові номери, серед яких він хоче розповсюдити цей текст.

3. Користувач визначає, коли від хоче відправити цей текст.

4. Система перевіряє, чи час відправлення не йде перед реальним часом.

5. Система проводить розсилку повідомлення до відповідних пристроїв клієнтів відповідних номерів.

6. Система створює новий запис про відправку повідомлень, та додає до нього текст, індекси номерів та час відправки. Після цього, система зберігає запис у базі даних та підтверджує успішне завершення відправлення.

Альтернативний сценарій:

4.а. Час відправлення йде перед реальним часом.

4.а.1. Система попереджує про це користувача.

4.а.2. Користувач переходить до пункту 3.

5.а. Повідомлення не було відправлено до (мінімум) одного з адресатів.

5.а.1. Система обирає номери, до яких не було доставлене повідомлення.

5.а.2. Система проводить розсилку повідомлення до відповідних пристроїв клієнтів відповідних номерів.

5.а.3. Система переходить до пункту 6.

**Сценарій варіанту використання «Прийняття запитів від клієнтів»**

Передумова: працівник авторизований у системі взаємодії з клієнтами.

Актори: працівник по розміщенню.

Основний актор: працівник по розміщенню.

Основний сценарій:

1. Користувач створює короткий опис запиту.

2. Користувач обирає категорії робіт та послуг, відповідних для отриманого запиту.

3. Користувач вводить номер житла, клієнт якого подав запит.

4. Система перевіряє, чи час запиту знаходиться у проміжку часу роботи працівників та помічає запит як обробляємий.

5. Система проводить розсилку повідомлення до відповідних пристроїв працівників.

6. Система створює новий запис про відправку повідомлень, та додає до нього короткий опис робіт, індекс номеру, категорії робіт а також помітку на запит. Після цього, система зберігає запис у базі даних та підтверджує успішне завершення відправлення.

Альтернативний сценарій:

4.а. Час запиту не знаходиться у проміжку часу роботи працівників.

4.а.1. Система помічає запит як відкладений.

4.а.2. Система переходить до пункту 6.

5.а. Повідомлення не було відправлено до (мінімум) одного з адресатів.

5.а.1. Система обирає пристрої, до яких не було доставлене повідомлення.

5.а.2. Система проводить повторну розсилку повідомлення.

5.а.3. Система переходить до пункту 5.

**Сценарій варіанту використання «Організація роботи робочого колективу»**

Передумова: працівник дає запит до системи на отримання всіх відкладених повідомлень.

Актори: адміністратор.

Основний актор: адміністратор.

Основний сценарій:

1. Користувач проводить процес сортування запитів.

2. Система отримує новий список запитів на виконання.

3. Система перевіряє, чи не порожній отриманий список.

4. Система бере перший запит у списку.

5. Система оновлює помітку запиту як обробляємий.

6. Система проводить розсилку повідомлення до відповідних пристроїв працівників.

7. Система оновлює запис запиту у базі даних та видаляє його зі списку.

8. Система переходить до пункту 3.

Альтернативний сценарій:

3.а. Список запитів порожній.

3.а.1. Система підтверджує успішне завершення відправлення.

6.а. Повідомлення не було відправлено до (мінімум) одного з адресатів.

6.а.1. Система обирає пристрої, до яких не було доставлене повідомлення.

6.а.2. Система проводить повторну розсилку повідомлення.

6.а.3. Система переходить до пункту 7.

**Сценарій варіанту використання «Контроль за роботою робочого колективу»**

Передумова: користувач дає запит на отримання списку всіх клієнтських запитів.

Актори: адміністратор.

Основний актор: адміністратор.

Основний сценарій:

1. Користувач отримує клієнтський запит.

2. Користувач перевіряє, що запит взятий в обробку.

3. Система перевіряє, що всі запити перевірені користувачем.

Альтернативний сценарій:

2.а. Запит не взятий в обробку.

2.а.1. Користувач дає запит на отримання списку всіх службових працівників службового.

2.а.2. Користувач обирає працівника.

2.а.4. Система оновлює у записі статус, міняючи його на виконуємий, а також додає до нього працівника.

2.а.5. Система пересилає дані запиту на пристрій обраного працівника.

**Сценарій варіанту використання «Створення технологічної карти страви»**

Передумова: працівник авторизований у системі взаємодії з клієнтами.

Актори: працівник зони харчування.

Основний актор: працівник зони харчування.

Основний сценарій:

1. Користувач вводить назву технологічної карти.

2. Система перевіряє, що технологічна карта за такою назвою не існує.

3. Користувач додає дані про продукт, який необхідний при створенні страви, а саме, його назву та кількість.

4. Користувач визначає, що він не бажає додати дані про ще один продукт.

5. Система створює запис про нову технологічну карту та додає до нього назву та дані продуктів, необхідні для приготування страви. Після цього, система зберігає запис у базі даних та підтверджує успішне створення.

Альтернативний сценарій:

2.а. Технологічна карта за такою назвою існує.

2.а.1. Система попереджує про це користувача.

2.а.2. Система переходить до пункту 1.

4.а. Користувач бажає додати дані про ще один продукт.

4.а.1. Система переходить до пункту 3.

**Сценарій варіанту використання «Ідентифікація та інвентаризація майна»**

Передумова: працівник авторизований у системі взаємодії з клієнтами.

Актори: службовий працівник.

Основний актор: службовий працівник.

Основний сценарій:

1. Користувач вводить назву локації (номер, службове приміщення або зона продажу товарів/послуг).

2. Система перевіряє, чи існує локація за введеною назвою.

3. Користувач вводить код, що відповідає конкретному предмету, що підлягає інвентаризації.

4. Система перевіряє, що код не зареєстрований.

5. Користувач додає опис про предмет, що підлягає інвентаризації.

6. Користувач визначає, що не хоче більше додавати предметів.

7. Система створює запис про предмети, що підлягають інвентаризації, та додає до кожного з них назву локації, код та дату створення запису. Після цього, система зберігає запис у базі даних та підтверджує успішне створення.

Альтернативний сценарій:

2.а. Локації за такою назвою не існує.

2.а.1. Система попереджує про це користувача.

2.а.2. Система переходить до пункту 1.

4.а. Предмет з таким кодом вже зареєстрований в системі.

4.а.1. Система попереджує про це користувача.

4.а.1. Система переходить до пункту 3.

6.а. Предмет з таким кодом вже зареєстрований в системі.

6.а.1. Система переходить до пункту 3.

**Сценарій варіанту використання «Здійснення продажу»**

Передумова: працівник авторизований у системі взаємодії з клієнтами..

Актори: працівник зони продажу / розважальної зони / зони харчування.

Основний сценарій:

1. Користувач вводить назву продукту.

2. Система перевіряє, що продукт з такою назвою існує.

3. Користувач вводить кількість продукту, який замовив клієнт.

4. Система перевіряє, що введена кількість в правильних межах (від нуля не включно до безкінечності).

5. Користувач визначає, що не хоче більше додавати продуктів.

6. Система видає суму, на яку має бути оплачені товари/послуги.

7. Користувач підтверджує, що товари/послуги оплачені.

8. Система створює запис про оплату, та додає до нього дату проведення транзакції, особу, що провела цю транзакцію, а також товари/послуги, які передані клієнту. Після цього, система зберігає запис у базі даних та підтверджує успішне створення.

Альтернативний сценарій:

2.а. Продукту з такою назвою не існує.

2.а.1. Система попереджує про це користувача.

2.а.2. Система переходить до пункту 1.

4.а. Введена кількість не відповідає правильним межам.

4.а.1. Система попереджує про це користувача.

4.а.2. Система переходить до пункту 3.

5.а. Користувач хоче додати ще один продукт.

5.а.1. Система переходить до пункту 1.

**Сценарій варіанту використання «Повідомлення про переміщення товарів»**

Передумова: працівник авторизований у системі взаємодії з клієнтами.

Актори: працівник зони продажу.

Основний актор: працівник зони продажу.

Основний сценарій:

1. Користувач вводить назву товару.

2. Система перевіряє, що товар з такою назвою існує.

3. Користувач вводить кількість продукту, яка підлягає переміщенню.

4. Система перевіряє, що введена кількість в правильних межах (від нуля виключно до безкінечності).

5. Користувач визначає, що не хоче більше додавати продуктів.

6. Користувач додає дані про працівника комплексу, який відповідальний за приймання товару.

7. Система перевіряє, що дані цього працівника зареєстровані в системі.

8. Система створює запис про переміщення, та додає до нього дату проведення переміщення, особу, відповідальну за переміщення, особу, відповідальну за приймання товару, а також товари. Після цього, система зберігає запис у базі даних та підтверджує успішне створення.

Альтернативний сценарій:

2.а. Продукту з такою назвою не існує.

2.а.1. Система попереджує про це користувача.

2.а.2. Система переходить до пункту 1.

4.а. Введена кількість не відповідає правильним межам.

4.а.1. Система попереджує про це користувача.

4.а.2. Система переходить до пункту 3.

5.а. Користувач хоче додати ще один продукт.

5.а.1. Система переходить до пункту 1.

7.а. Даних про працівника не існує в системі.

7.а.1. Система попереджує про це користувача.

7.а.2. Система переходить до пункту 6.

**Сценарій варіанту використання «Повідомлення про закупівлю товарів»**

Передумова: працівник авторизований у системі взаємодії з клієнтами.

Актори: працівник зони продажу.

Основний актор: працівник зони продажу.

Основний сценарій:

1. Користувач вводить назву товару.

2. Користувач вводить кількість продукту.

3. Система перевіряє, що введена кількість в правильних межах.

4. Користувач визначає, що не хоче більше додавати продуктів.

5. Система створює запис про закупівлю, додає до нього дату проведення переміщення, особу, відповідальну за приймання, а також товари. Після цього, система зберігає запис у базі даних та підтверджує створення.

Альтернативний сценарій:

3.а. Введена кількість не відповідає правильним межам.

3.а.1. Система попереджує про це користувача.

3.а.2. Система переходить до пункту 2.

4.а. Користувач хоче додати ще один продукт.

4.а.1. Система переходить до пункту 1.

**Сценарій варіанту використання «Повідомлення про факт завершення роботи»**

Передумова: користувач дає запит на отримання списку всіх клієнтських запитів для нього.

Актори: службовий працівник.

Основний актор: службовий працівник.

Основний сценарій:

1. Користувач обирає клієнтський запит.

2. Користувач оновлює статус запиту на виконаний.

3. Користувач додає опис проведених робіт.

4. Система оновлює статус запиту, додає до нього опис та дату завершення робіт.

5. Система пересилає дані про виконаний запит до всіх адресатів.

**Сценарій варіанту використання «Повідомлення про нещасний випадок»**

Передумова: користувач зареєстрований у системі взаємодії з клієнтами.

Актори: працівник розважальної зони.

Основний актор: працівник розважальної зони.

Основний сценарій:

1. Користувач додає опис випадку.

2. Система перевіряє, чи хоче користувач додати товар або обладнання, яке постраждало при нещасному випадку.

3. Користувач додає товар/обладнання, яке постраждало при нещасному випадку.

4. Система перевіряє, що такий продукт або обладнання існує серед записів у базі даних.

5. Користувач додає кількість товару/обладнання.

6. Система перевіряє, що введена кількість в правильних межах (від нуля виключно до безкінечності).

7. Користувач додає опис до предмету.

8. Система переходить до пункту 2.

Альтернативний сценарій:

2.а. Користувач не хоче додати предметів, які постраждали при нещасному випадку.

2.а.2. Система створює запис та додає до нього опис випадку, список предметів, які додав користувач, а також дату створення звіту.

4.а. Запис за таким предметом не найдений у базі даних.

4.а.1. Система попереджає про це користувача.

4.а.1. Система переходить до пункту 3.

6.а. Введена кількість предметів є негативною.

6.а.1. Система попереджає про це користувача.

6.а.1. Система переходить до пункту 5.

## **3.4 Нефункціональні вимоги до програмного забезпечення**

Розглянувши відомі рішення для автоматизації роботи визначимо необхідні функціональні та нефункціональні вимоги. Так, нефункціональними вимогами до програмної системи є:

1. дані повинні бути зашифровані так, щоб час на їх розшифрування зайняв не менше 1 місяця;
2. система повинна витримувати обробку 5000 запитів за хвилину при нормальному стані роботи;
3. система повинна витримувати обробку 1000 запитів за хвилину при критичному стані роботи;
4. при запиті необхідних даних відповідь має прийти не більше ніж через п’ять секунд;
5. при створенні нових даних їх обробка повинна займати не більше ніж 2 секунди.

## **3.5 Висновки до розділу**

В розділі визначено, які функціональні та нефункціональні вимоги необхідно задовільнити при реалізації системи взаємодії з клієнтами, які актори приймають участь в діяльності курортного комплексу, та які функції кожен з них виконує. Також описані сценарії варіантів використання для кожної з функцій комплексу.

# **4 ПРОЕКТУВАННЯ СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ**

## **4.1 Проектування структури та організації класів**

Перед тим, як визначитися зі структурою діаграми класів, необхідно визначити її основні структурні компоненти, тобто, визначити, які класи мають бути присутні та які відносини вони мають між собою. Для цього ми можемо скористатися описами варіантів використання. Так, якщо ми хочемо визначити, що саме буде класами для можливості реалізувати збереження клієнту курортного комплексу у вигляді запису у базі даних, ми можемо скористатись діаграмою послідовності. Але для нашої кількості варіантів використання сукупна кількість діаграм послідовностей буде занадто великою для розуміння. Тому ми розглянемо, яка послідовність дій при створенні нового замовлення на розміщення клієнта.

Якщо спрощено – працівник комплексу додає дані про клієнта (ім’я, прізвище, стать, вік та телефон), запускає процес збереження у базі даних цих даних, а після цього проводить передачу даних до інших вузлів розгалуженої мережі комплексу, причому лише до тих, які зацікавлені в отриманні даних про клієнта. Всі ці дії також повторюються при процесі створення нового замовлення на розміщення. Також важливо зазначити, що варіанти використання не роз’яснюють, які дії мають відбуватися на стороні отримувача даних, тому на діаграмах ми зобразимо і цей процес.

Для діаграм послідовності важливо відмітити, що послідовність деяких з описаних дій для кінцевого користувача можуть відрізнятись від тієї, що зазначена на рис. 4.1 – 4.5. Проте, частіше за все, це не вплине на результат, або на швидкість виконуємих системою дій. Однак, це не означає, що всі дії алгоритму можуть знаходитись в будь-якому порядку. Так, наприклад, процес відправки нових даних не може йти перед етапом запису цих даних у базі, і навпаки.

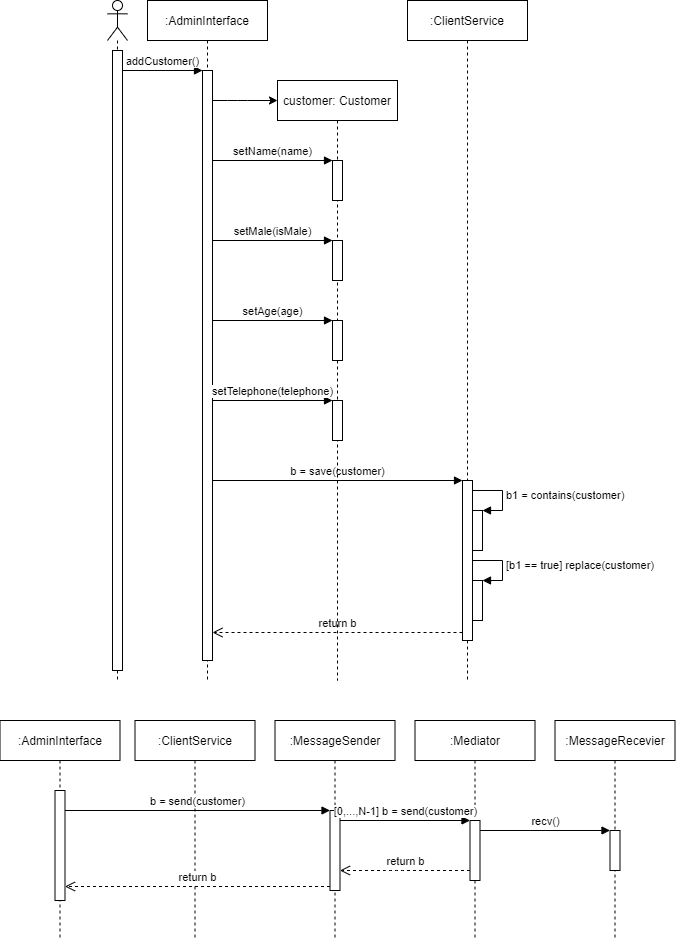


Рисунок 4.1 – Послідовність дій системи при створенні нового запису клієнта



Рисунок 4.2 – Послідовність дій при створенні нового запису про розміщення

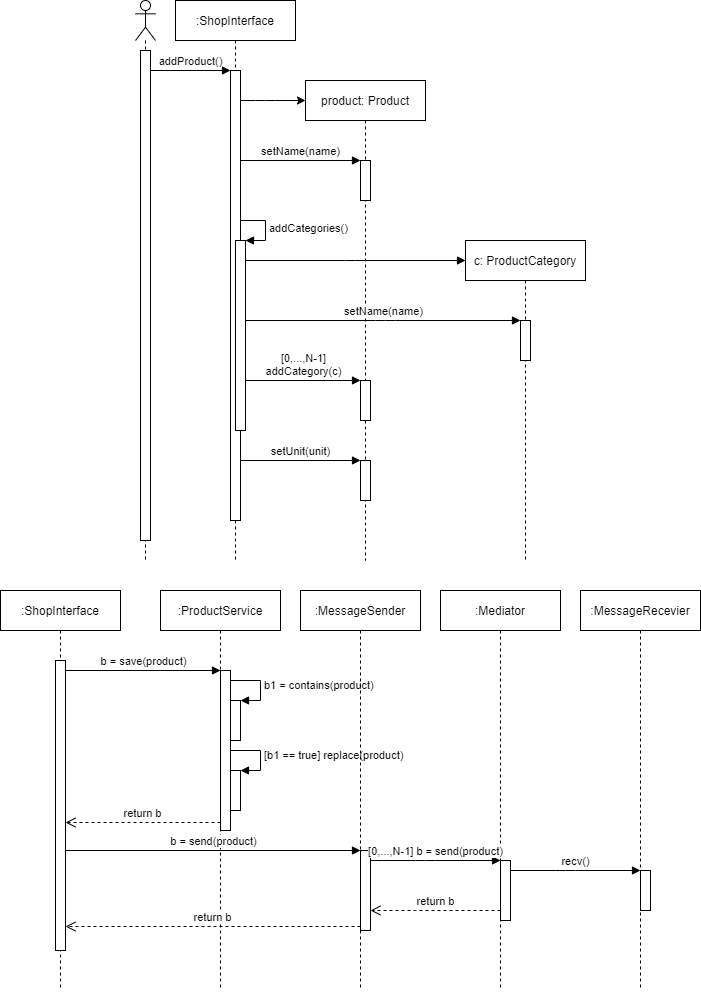


Рисунок 4.3 – Послідовність дій при створенні нового запису продукту

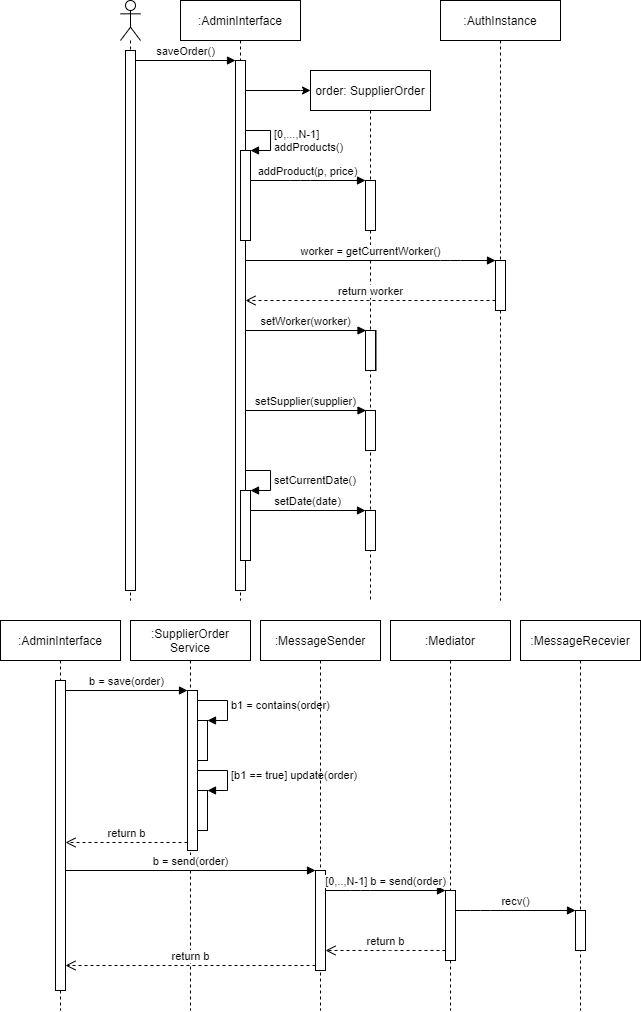


Рисунок 4.4 – Послідовність дій при створенні замовлення продукту

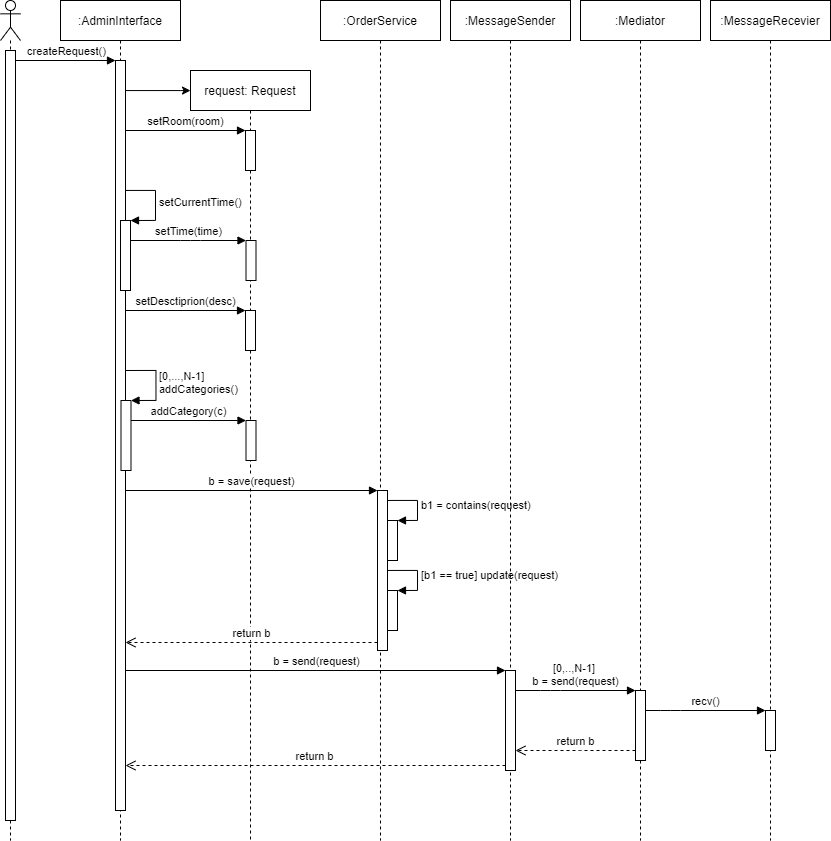


Рисунок 4.5 – Послідовність дій при створенні запиту від клієнту до службового персоналу

Ще раз зазначимо, що ці діаграми презентують лише малу частину загальної системи. Тут не було розглянуто, як користувач може додавати нові номери, як відбувається процес створення технологічних карт страв, що можуть продаватися у місцях людського харчування, або як клієнт створює замовлення для службового персоналу і що робітник повинен зробити для завершення цього замовлення. Загальна діаграма класів буде виглядати, як показано на рис. 4.6.

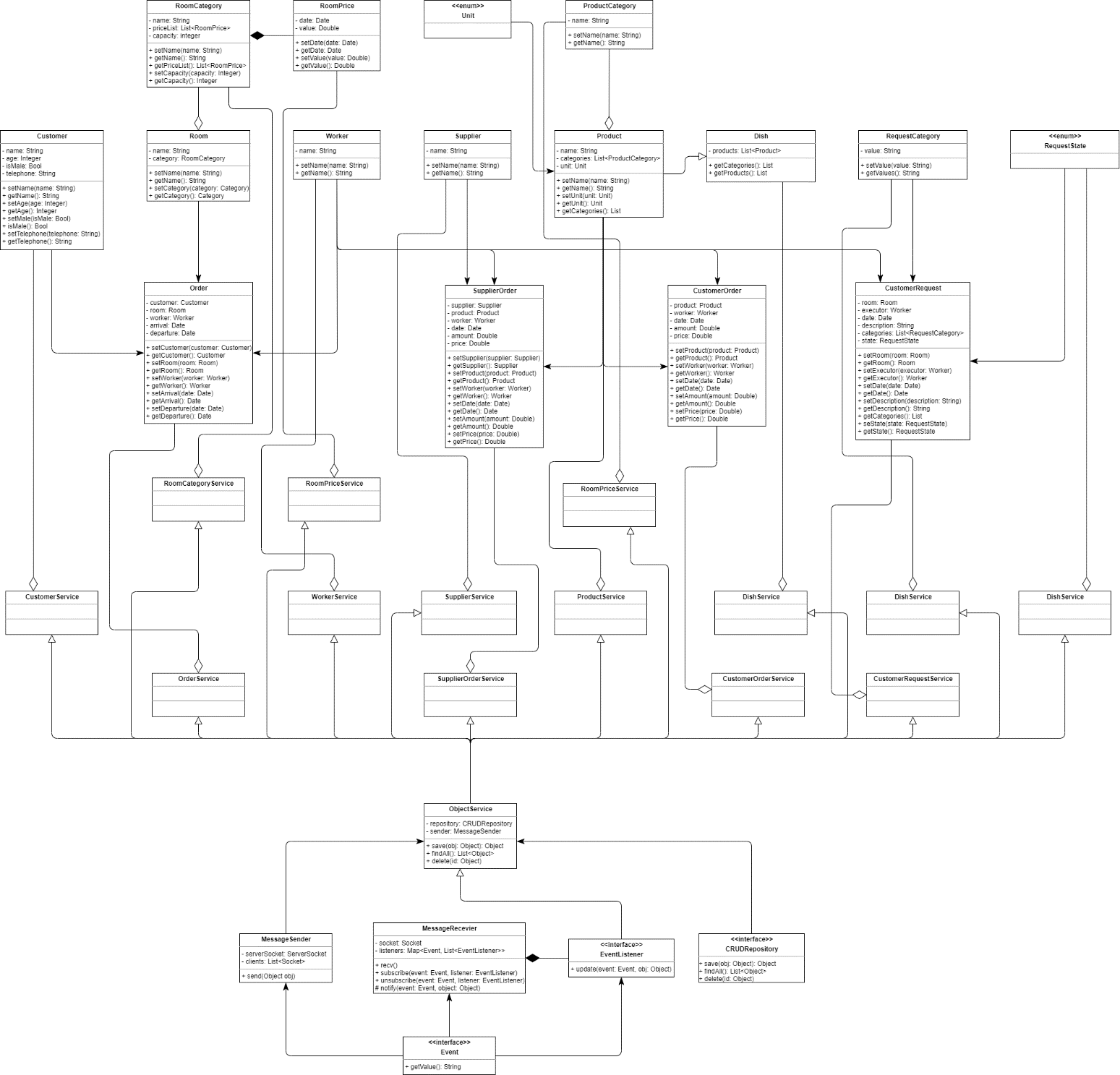


Рисунок 4.6 – Основні класи системи

Можна відмити, що на даній діаграмі існує багато однакових класів, які мають одного і той самого родича-класу. Це – класи, відповідальні за сериалізацію та десериалізацію даних. Їхня структра дуже схожа між різними класами-братами, тому детально розглянемо частину класів, що відповідає за роботу зони розміщення клієнтів а також за передачу даних по внутрішній мережі курортного комплексу (рис. 4.7 – 4.11):

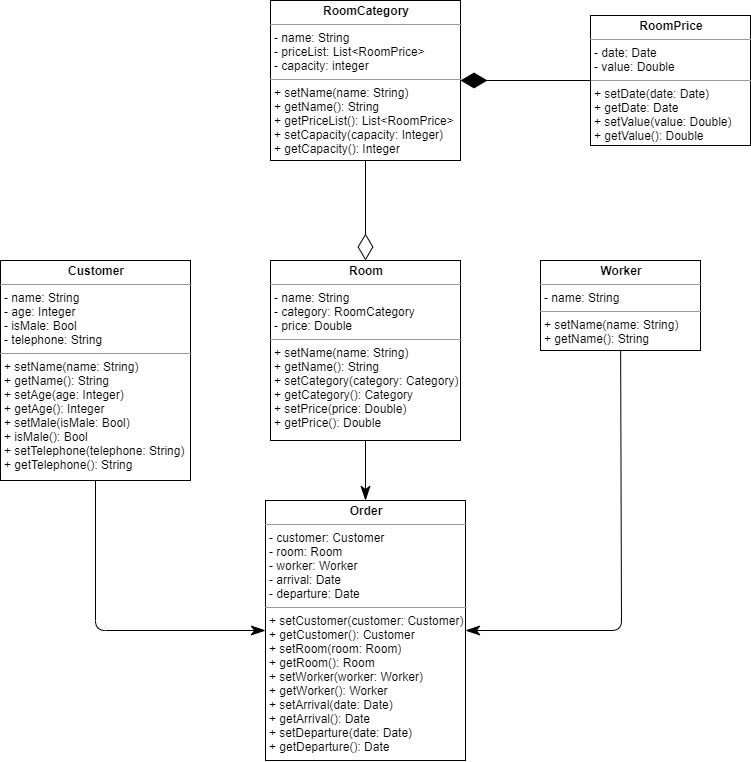


Рисунок 4.7 – Класи системи для зони розміщення клієнтів

Описані класи мають такі властивості: RoomCategory – клас, що відповідає сутності категорії номеру, має свою назву, кількість людей, які можуть проживати в номері такої категорії одночасно, а також список цін. Ціни на номери представлені класом RoomPrice, та мають властивість дати встановлення ціни та суму проживання у номері за один день. Клас Room відповідає сутності номеру в курортному комплексі, та має свою назву та категорію. Клас Customer відповідає за відображення властивостей клієнта, а саме, за відображення імені, віку, статі та номеру телефону. Клас Worker визначає людину, яка є працівником курортного комплексу, та може створювати нові замовлення. За відображення даних про замовлення відповідальний клас Order, властивостями якого є клієнт, що забронював номер, сам номер, працівник, відповідальний за обробка замовлення, а також дата заїзду та виїзду.

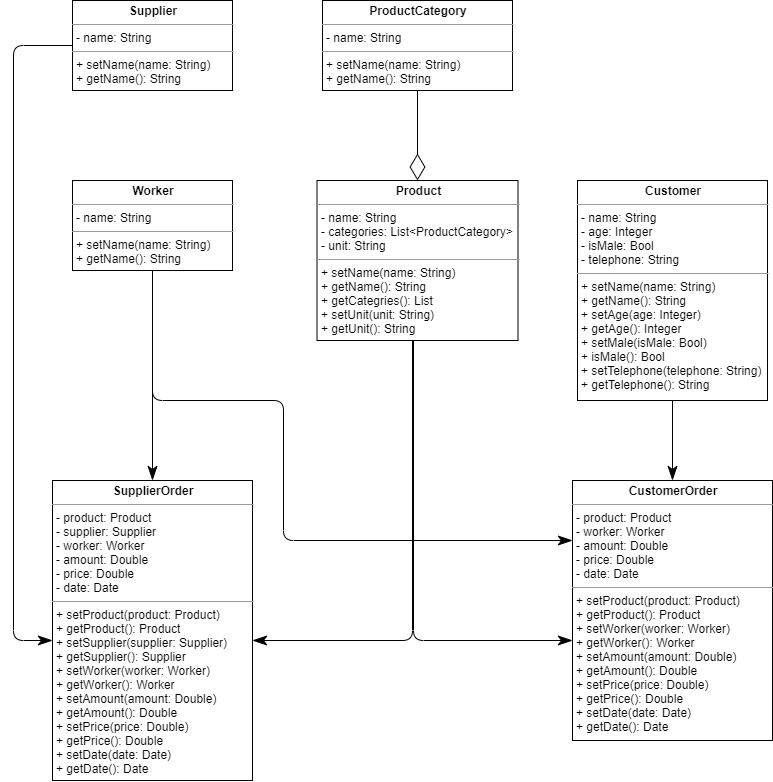


Рисунок 4.8 – Класи системи для зони продажу продуктів

Описані класи мають такі властивості: ProductCategory – клас, що відповідає сутності категорії товару на продаж, та має лише назву як властивість. Клас Product відповідає сутності товару, що продається, або закуповується, в курортному комплексі, має свою назву, категорії товарів а також одиницю вимірювання. Клас Supplier відповідає за відображення властивостей поставника товарів, за відображення імені. Клас Worker визначає людину, яка є працівником курортного комплексу, та може створювати нові замовлення. За відображення даних про замовлення відповідальний клас SupplierOrder та CustomerOrder. Перший відповідає за відображення даних про закупівлю, а другий – продаж покупцю необхідного товару.

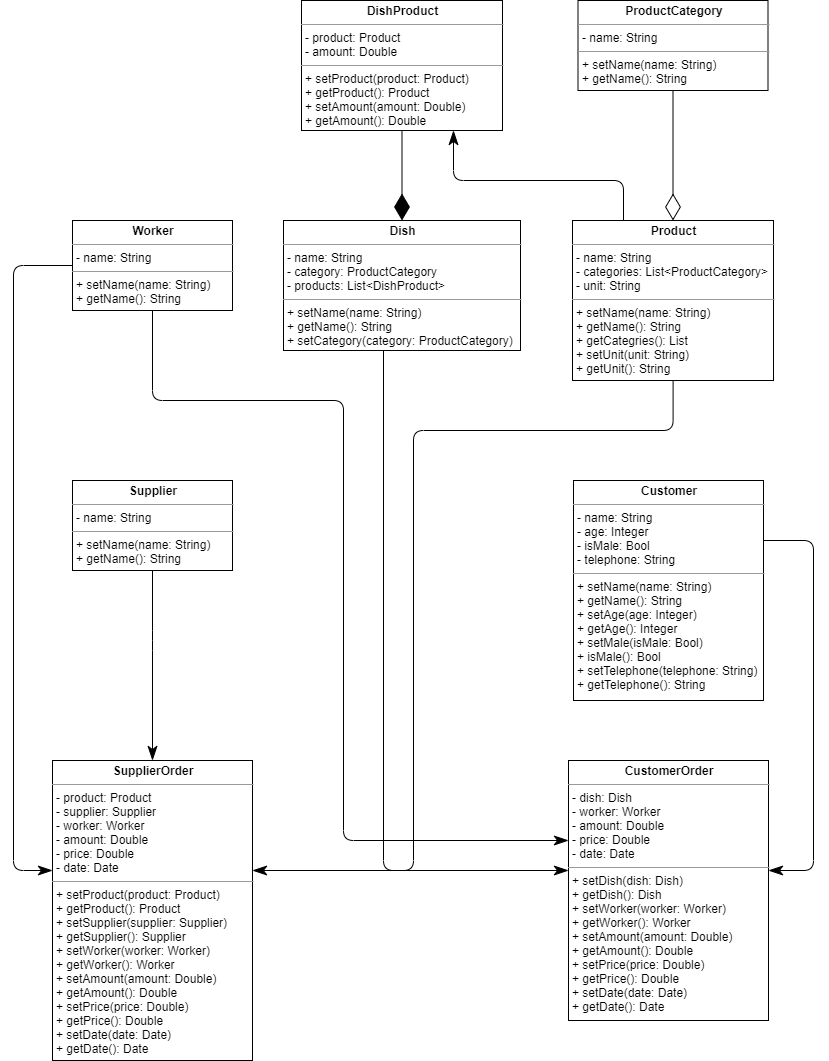


Рисунок 4.9 – Класи системи для зони людського харчування

Описані класи мають практично ті самі властивості, що і класи, описані на рис. 4.8. Однак існує декілька відмінностей: такі властивості: Клас Dish відповідає сутності страви, що продається в курортному комплексі, має свою назву, категорію страви, а також список продуктів, описані класом Product, а також кількість цього самого продукту.

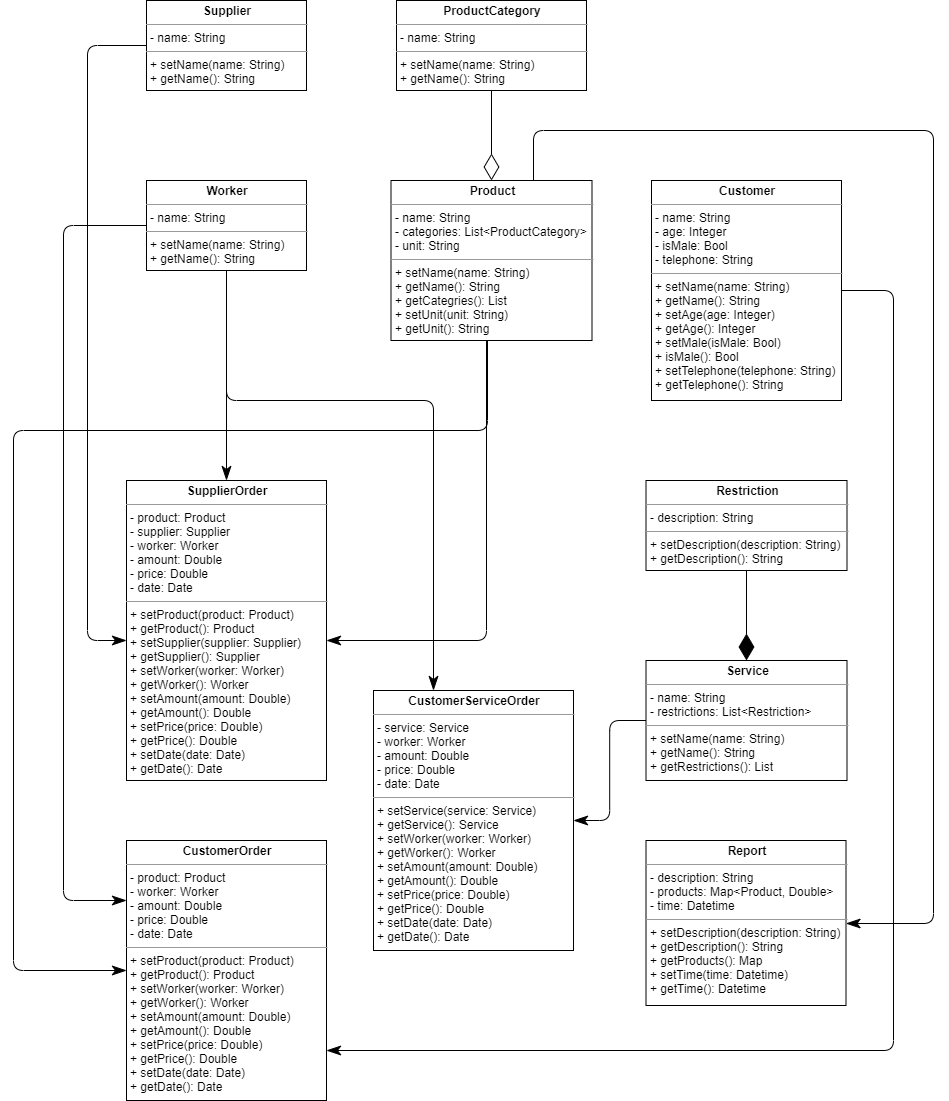


Рисунок 4.10 – Класи системи для розважальної зони

Описані класи також мають багато класів та властивостей, які вже описані на рис. 4.8 – 4.9. Однак вони також мають декілька нових унікальних класів. Клас Service відповідає сутності послуги, яку надають працівники розважальної зони клієнтам. Кожна з послуг має кінцевий список обмежень, які необхідно задовільнити для користування послугами. Також для цього підрозділу існує окремий клас Report, що відображає звіти працівників про нещасні випадки.

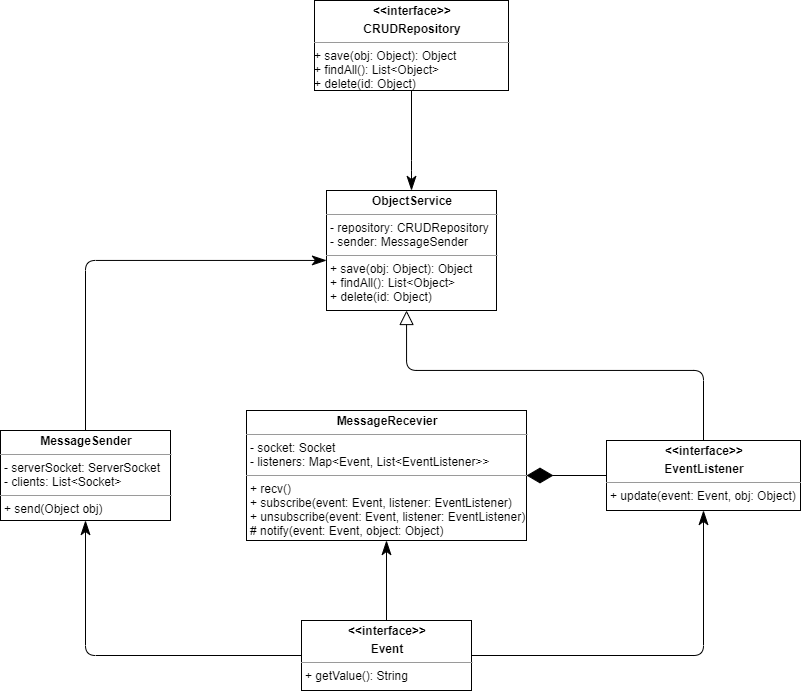


Рисунок 4.11 – Класи системи для роботи передачі даних по внутрішній мережі

Описані класи мають необхідні для маршрутизації даних по внутрішній мережі комплексу, а також для сериалізації та десериалізації об’єктів системи. Так, об’єкт класу MessageSender відповідальний за посилання даних,які має прийняти об’єкт класу MessageReceiver. Після того, як одержувач отримує дані, він проводить передачу даних всередині системи. Так, за структурним шаблоном Observer, клас EventListener і має отримати ці дані, та обробити їх своїм способом, наприклад, зберегти їх у базі даних, для чого існує клас ObjectService.

## **4.2 Проектування розподіленої реляційної бази даних**

Для проектування розподіленої реляційної бази даних ми маємо визначити, які саме об’єкти мають бути реалізовані в якості записів у базі даних. Так, база даних для відповідного структурного елементу повинна зберігати таблиці тільки для тих об’єктів, що мають безпосередню участь при створенні, редагуванні, або читанні. Наприклад, для структурного підрозділу розміщення клієнтів найважливіше зберігати дані про клієнтів, номери та замовлення номерів, та не матимуть значення дані по продажу страв в зоні харчування. Так, ми отримуємо наступні бази даних (рис. 4.12 – 3.18):

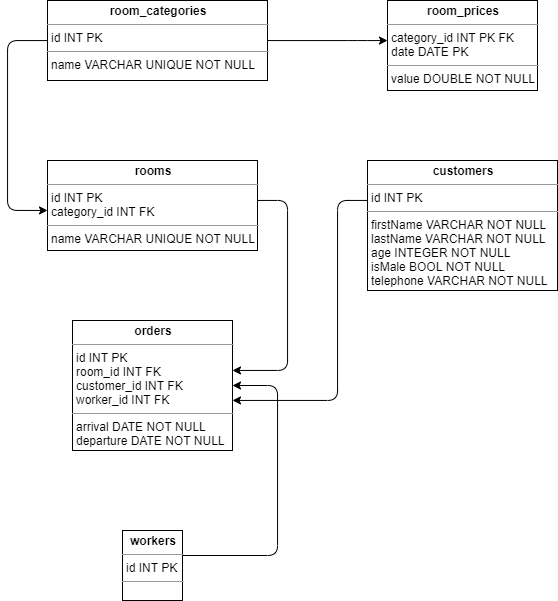


Рисунок 4.12 – Модель бази даних для зони розміщення клієнтів

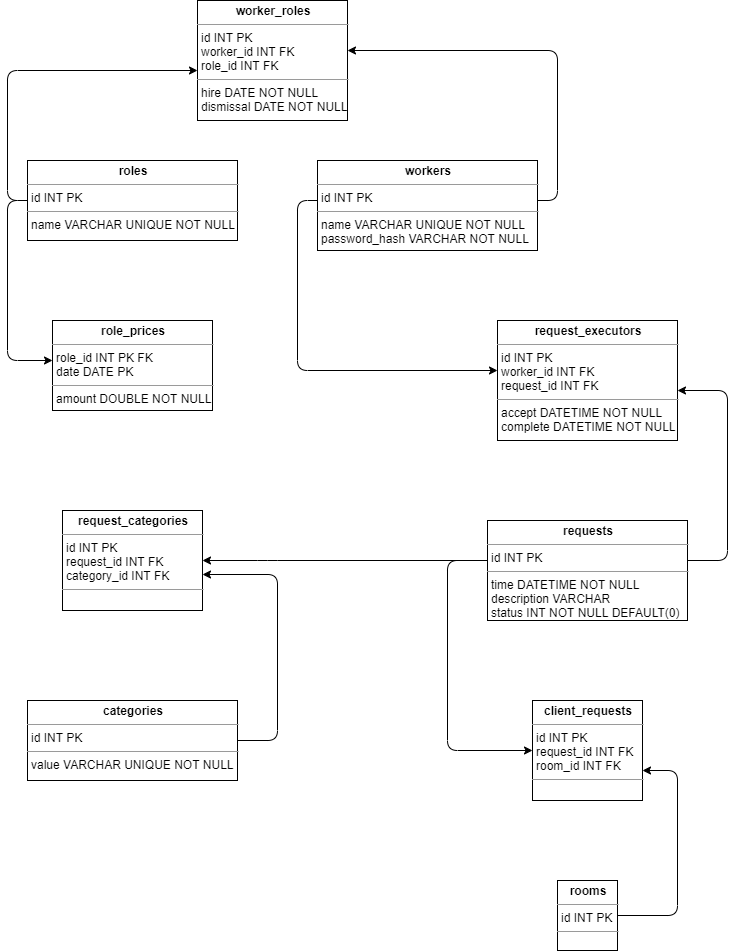


Рисунок 4.13 – Модель бази даних для корпусу службового обслуговування та адміністрування

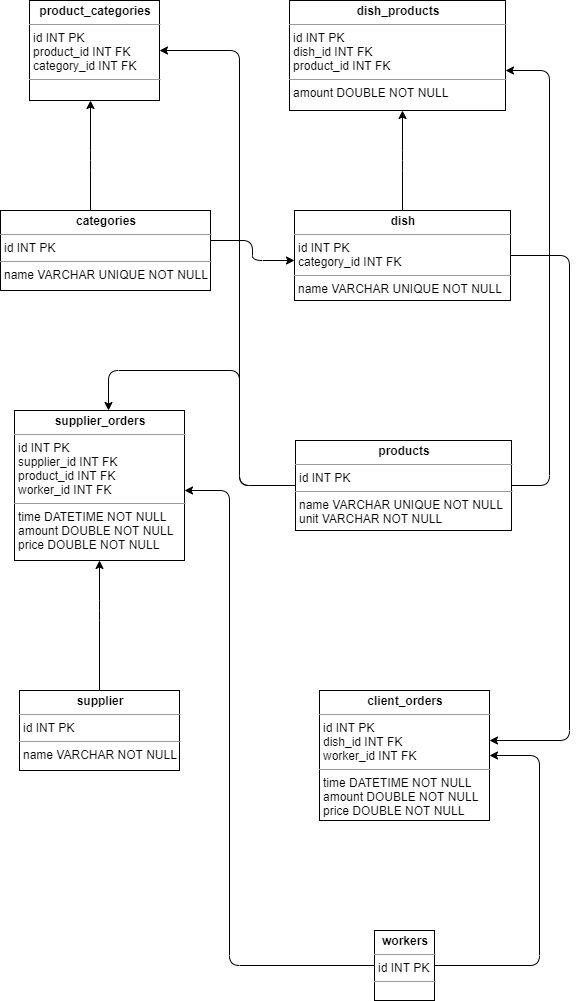


Рисунок 4.14 – Модель бази даних для зони людського харчування

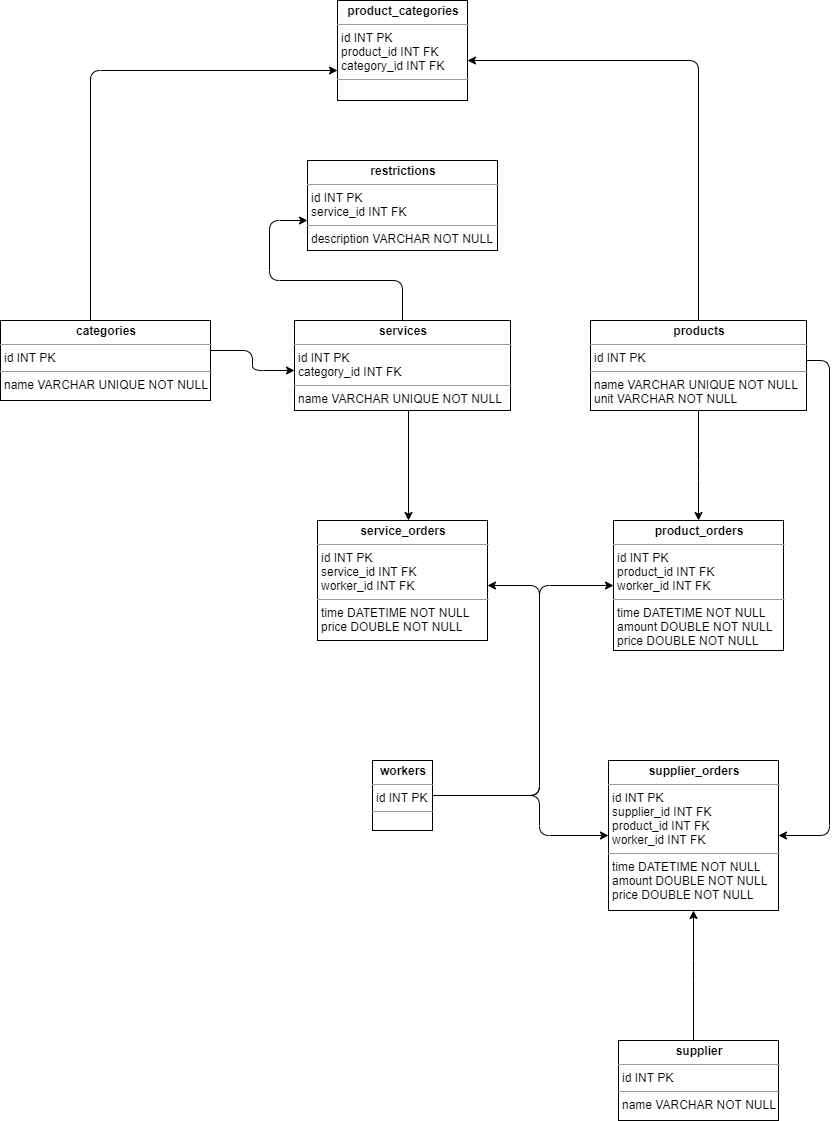


Рисунок 4.16 – Модель бази даних для працівників розважальної зони

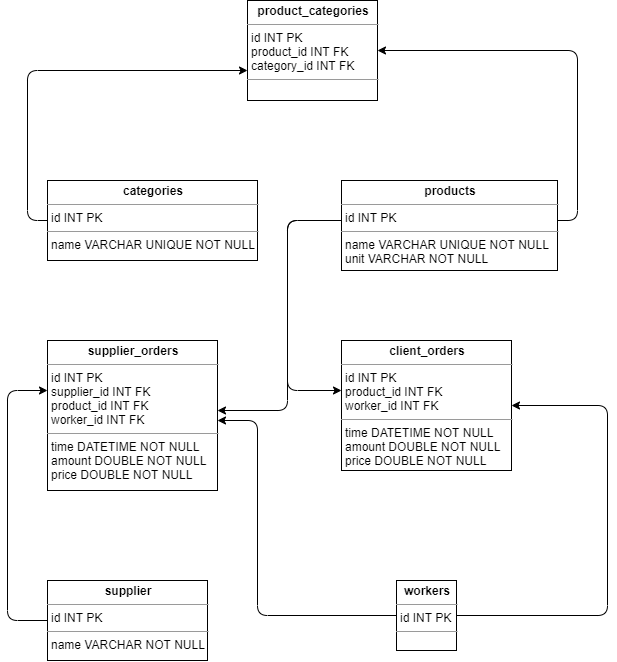


Рисунок 4.15 – Модель бази даних для зони продажу господарських товарів

Кожна з цих таблиці уявляє собою відображення існуючих серед описаних класів зі своїми властивостями. Тут дані властивості представлені як рядки у відповідних таблицях. Стрілками позначено як відносини між об’єктами класів переводяться в записи у таблицях. Наприклад, на рисунку 3.8. позначено, як відносини композиції між класом RoomCategory та RoomPrice відображуються в базі даних.

Також, в кожній базі даних існує хоча б одна таблиця, що має лише один рядок, який зберігає ідентифікатор об’єкту. Дані таблиці необхідні для прискорення роботи системи в різних підрозділах, зменшуючи напругу на внутрішню мережу при передачі даних.

## **4.3 Висновки до розділу**

В розділі розглянуто, які архітектурні рішення використані для задовільнення потреб замовника, яка послідовність дій приводить до бажаного результату, описані класи, які є важливими для окремого структурного елементу комплексу, а також спроектовано моделі реляційної бази даних для кожного елементу.

# **5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ**

## **5.1 Набір інструментальних засобів розробки**

Програмне забезпечення JetBrains IntelliJ IDEA [12] - це провідне середовище швидкої розробки мовою Java. IntelliJ IDEA являє собою високотехнологічний комплекс тісно інтегрованих інструментів програмування, що включає інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації, потужні інструменти рефакторинга коду, вбудовану підтримку технологій J2EE, механізми інтеграції з середовищем тестування Ant / JUnit і системами управління версіями, унікальний інструмент оптимізації та перевірки коду Code Inspection, а також інноваційний візуальний конструктор графічних інтерфейсів.

REST Advanced Client – допоміжна програма веб-розробників для створення та тестування власних запитів HTTP. Єдиний клієнт REST, який здійснює підключення безпосередньо в сокеті, надаючи вам повний контроль над заголовками підключення та запиту / відповіді. Ви можете налаштувати проксі-сервер у налаштуваннях Chrome, якщо у вас виникли проблеми з підключенням до віддаленого комп'ютера.

Особливості: інтегровано з Google Drive; запити, зроблені на сокетах, дають вам більше контролю над HTTP-запитами; встановлення власне кодування форми; пам'ять останніх запитів; історія запитів; імпорт / експорт даних.

Heroku – платформа хмарного контролю сервісів. Це означає, що вам не потрібно турбуватися про інфраструктуру; ви просто зосередитеся на своєму додатку. Особливостями платформи є миттєвий деплоймент з GitHub; push-збірка вашого забезпечення виконується Heroku за допомогою сценаріїв збірки; безліч додаткових ресурсів (додатки, бази даних і т.д.); процес масштабування незалежний для кожного компонента вашої програми без шкоди для функціональності і продуктивності; кожен процес повністю ізольований один від одного; легкий доступ до журналу з кожного компонента вашої програми і кожного процесу.

Heroku надає дуже добре написаний підручник, який дозволяє почати роботу в лічені хвилини. Крім того, вони надають перші 750 годин обчислень безкоштовно, що означає, що ви можете мати один процес (він же Dyno) безкоштовно. Крім того, продуктивність дуже хороша. простий веб-додаток, написаний на мові node.js, може обробляти близько 60-70 запитів в секунду.

Visual Studio Code - це легкий, але потужний редактор коду, який працює на робочому столі та доступний для Windows, macOS та Linux. Він постачається з вбудованою підтримкою JavaScript, TypeScript та Node.js і має багату екосистему розширень для інших мов (таких як C ++, C #, Java, Python, PHP, Go) та середовищ виконання (таких як .NET та Unity).

Firebase - це платформа розробки мобільних додатків з величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих кроссплатформенних додатків. Головне достоїнство платформи в тому, що вона дозволяє розробнику не відволікатися на створення бекенд, тобто прихованої від користувача програмної частини проекту, наприклад, серверного коду.

Це спрощує і прискорює створення мобільних застосувань, дає можливість повністю зосередитися саме на UX / UI, тобто, на призначеному для користувача інтерфейсі і досвіді. Саме зв'язка Firebase з фреймворком Flutter дозволяє програмістам компанії AVADA MEDIA створювати швидкі програми для Android і iOS, що дозволяють вирішувати найрізноманітніші завдання.

Застосування підключається до бази даних через WebSocket, який відповідає за синхронізацію даних протягом усього сеансу.

## **5.2 Алгоритми формування замовлення послуг по розміщенню**

### **Клієнтська частина**

Сам процес відправлення необхідних даних достатньо простий. Для цього необхідно реалізувати декілька функцій з можливістю відправляти запити за необхідною URL адресою, при цьому необхідна можливість відправляти запит з різними заголовками, наприклад, GET, POST, DELETE та PUT. В стандартному наборі JavaScript-бібліотек існує клас XMLHttpRequest,який всі ці описані потреби повністю реалізує. Так для запиту POST та GET ми отримуємо наступний результат:

**const** sendPost = **function**(url, json) {  
 **const** xhr = **new** XMLHttpRequest();  
 xhr.open(**"POST"**, url);  
 xhr.send(json);  
 **return** {code: xhr.responseCode, response: xhr.response};  
}  
  
**const** sendGet = **function**(url) {  
 **const** xhr = **new** XMLHttpRequest();  
 xhr.open(**"GET"**, url);  
 xhr.send(**null**);  
 **return** {code: xhr.responseCode, response: xhr.response};  
}

Наприклад, якщо ми хочемо зберегти відомості про клієнта, це буде виглядати наступним чином:

sendPost(

**"localhost:8080/customers"**,

{firstName:**"name"**, secondName:**"name"**, age: 23, telephone:**"telephone"**});

Знаючи це, послідовність дій для створення замовлення виглядатиме наступним чином:

if (arrival > departure) {

alert(**"Error"**);

}

**const** result = sendPost(

**"localhost:8080/customers"**,

{firstName:**"name"**, secondName:**"name"**, age: 23, telephone:**"telephone"**});

**if** (result.responseCode !== 200) {

alert(result.response);

}

**const** result = sendPost(**"localhost:8080/orders"**, {customer: result.response.id, room: selectedRoomIndex, worker: authWorker, arrival: selectedArrival, departure: selectedDeparture});

**if** (result.responseCode !== 200) { alert(result.response);}

### **Серверна частина**

З серверною частиною все дещо складніше, адже саме на сервер падає основна робота по обчисленню необхідних параметрів та роботи по збереженню та маніпулювання даними. Також на серверну частину лягає відповідальність за передачу необхідних даних по внутрішній мережі комплексу. Як приклад, можна привести програмну реалізацію контролеру, що працює з даними замовлень. Зазначимо, що для даного проекту ми використовуємо фреймворк Spring MVC та JAVA Persistence API.

Розглянемо клас RoomOrderController. За правилами взаємодії з RESTful API та методів, призвані обробляти запити, клас має бути анотований за допомогою інтерфейсу RestController. Після цього, можна визначати методи для обробки запитів за протоколом GET, POST та DELETE. Так, якщо ми хочемо отримати список всіх існуючих замовлень, ми просто даємо get-запит за url-адресою “host/rooms/orders”, після чого ми маємо отримати список. За умови, якщо нам необхідно витягти конкретне замовлення, ми додаємо до url-запиту ідентифікатор запиту, наприклад, “host/rooms/orders/{id}”. Якщо запис в базі даних за таким ідентифікатором існує, ми його і отримуємо, проте за умови помилки або відсутності запису, метод передає сигнал помилки, який потребує подальшої обробки.

Якщо ми хочемо додати інформацію, необхідно виконати декілька дій: створити JSON-тіло запиту, в якому визначено ідентифікатор клієнту (його запис необхідно додати до цього моменту), номер кімнати, ідентифікатор користувача, відповідального за створення запису, дату заїзду та дату виїзду. Після цього, за url-адресою “host/rooms/orders” посилається post-запит. Сам метод, що приймає цей запит, проводить процес зберігання, а після цього посилає дані по внутрішній мережі до всіх слухачів.

Тут важливо описати, що необхідно зробити для того, щоб процес отримання даних пройшов успішно. Для цього система, використовуючи сокетне рішення передачі даних підключається до «серверу» - джерела важливої інформації за заздалегідь визначеною url-адресою та портом прослуховування. Тепер, коли метод «серверу» для пересилання даних буде ініційовано, він відправить їх до всіх під’єднаних «клієнтів», проведши процес серіалізації. На стороні «клієнту» необхідно буде десеріалізувати об’єкт. Подальша робота «клієнта» не має впливати на роботу «сервера».

## **5.3 User-інтерфейс на прикладі «сітки» відображення замовлень клієнтів**

Якщо переглянути описи варіантів використання, можна побачити, що більшість з них вимагає в першу чергу зареєструватись у системі. Якщо враховувати, що розробка пов’язана зі зберіганням конфіденціальних даних як підприємства, то сам процес авторизації буде мати деякі відмінності.

Так, новий користувач сам не може зареєструватись у системі, для цього він повинен звернутись до відповідальної особи, яка має на це дозвіл. Також користувач не може мати можливості замінити/скинути дані для доступу, наприклад, скинути пароль за умови, що користувач забув його. При реалізації це буде виглядати так, як зображено на рисунку 4.1.

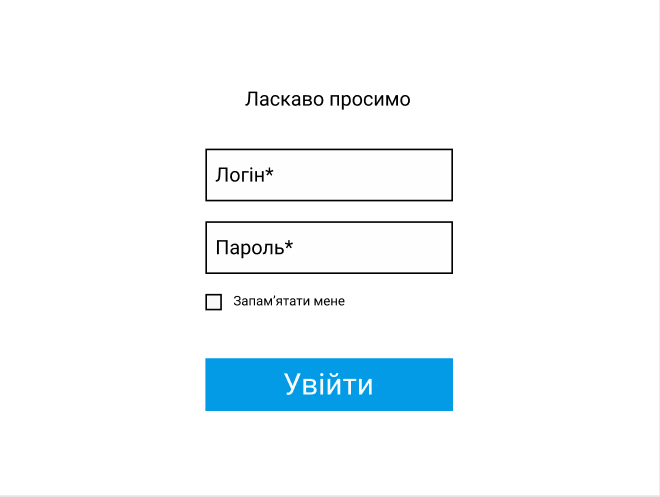


Рисунок 5.1 – Форма входу до системи

Якщо необхідно додати запис про нового користувача, відповідальна особа має отримати від користувача його повне ім’я, дату прийняття на роботу, місце роботи та номер телефону. Тільки після цього, особа зможе додати запис про нового робітника до системи (рис. 4.2).

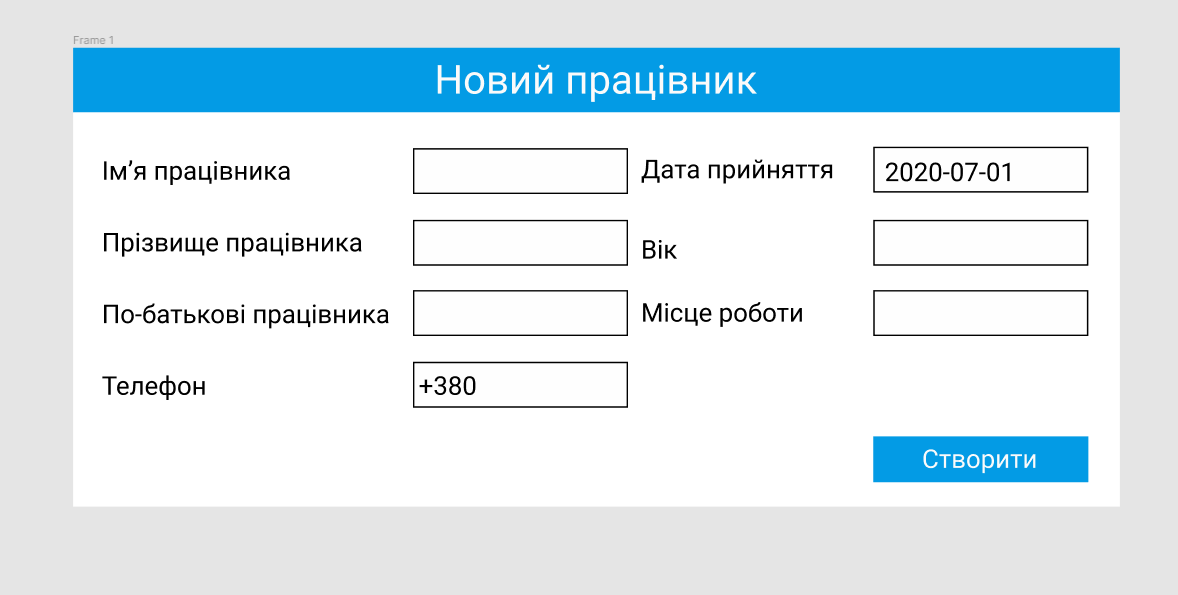


Рисунок 5.2 – Діалог додання нового працівника

Після цього, для користувача повинен бути відкритий доступ до робочих даних курортного комплексу. Наприклад, для робітника зони взаємодії з клієнтами повинен бути відкритий доступ до даних по розміщенню та бронюванню клієнтів в номерах. Також користувач, як відповідальна особа, може тепер редагувати ці дані (створювати, видаляти та оновлювати) за допомогою спеціальних форм взаємодії.

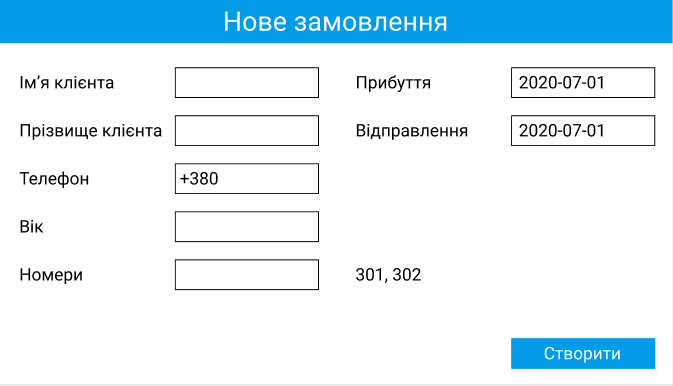


Рис 5.3 – Форма редагування даних бронювання номерів

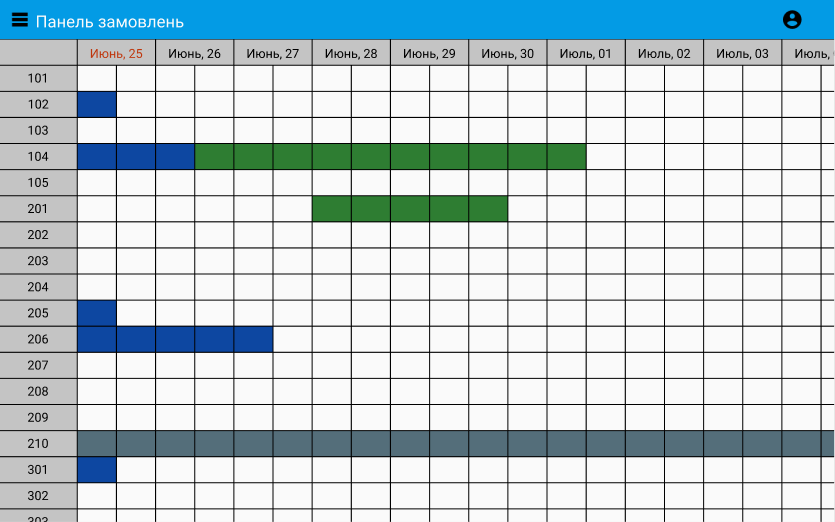


Рис 5.4 – Сітка відображення періодів розміщення клієнтів

Описані вище кроки приємлимі для працівника зони розміщення клієнтів. Проте, в якості акторів, які задіяні в роботі системи, також зазначені працівники зони продажу товарів та зони харчування. Важливо розглянути, як для такого клієнту виглядає процес продажу товарів. Все це можливо реалізувати у вигляді одного діалогового вікна (рис. 4.5).



Рис 4.5 – Форма продажу товарів

Якщо все пройшло успішно, то для користувача в списку чеків з’явиться новий чек з інформацією про усі куплені товари, кількість цих товарів та час продажу.

Для відпочиваючих в комплексі також передбачена взаємодія з системою. В рамках можливостей клієнта доступ до конфіденційних даних інших відпочиваючих не має бути взагалі, при цьому, щоб отримати доступ до особистих даних він має пройти процедуру верифікації розміщення в комплексі. Так, якщо користувач хоче увійти до особистого кабінету, йому необхідно буде ввести номер свого будинку, або, за умови того що клієнт забронював декілька номерів, будь-який з них. Після цього, система має перевірити, чи є заброньований на даний час номер, ким він заброньований. Так, знаючи телефонний номер клієнту, можна провести процес верифікації (рис. 4.5 – 4.6).

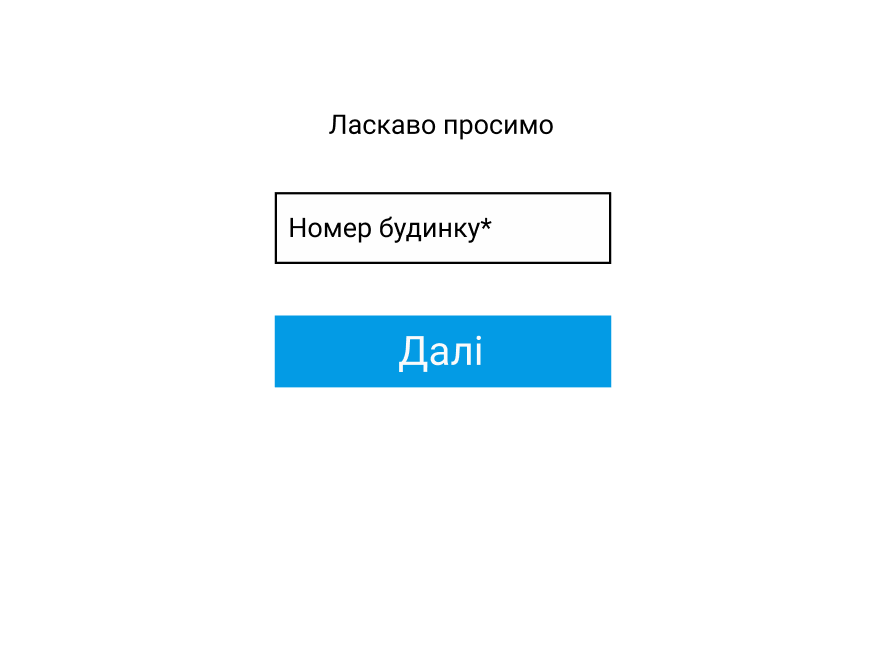


Рис 5.6 – Форма входу до системи відпочиваючих

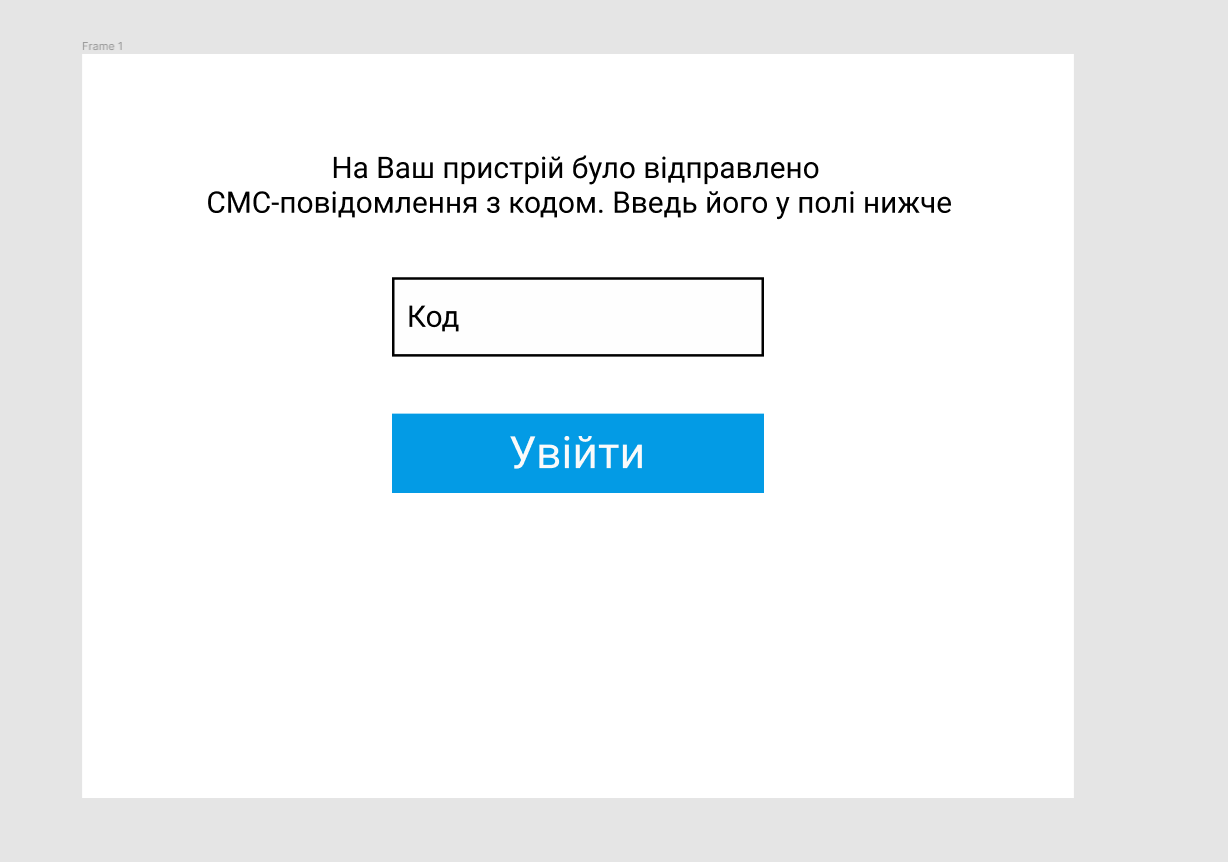


Рис 5.7 – Форма верифікації входу

Необхідно визначити, що саме може зробити відпочиваючий, після того, як він увійшов до особового кабінету. Наприклад, створити запит для службового персоналу, який мають розглянути. Для цього від клієнта потребується додати опис проблеми, з якою він зіткнувся, категорії проблеми (електро-, водопостачання, прибирання, тощо). За умови, що клієнт помилився з обраною категорією, він має можливість відмінити наявність цієї категорії у списку.

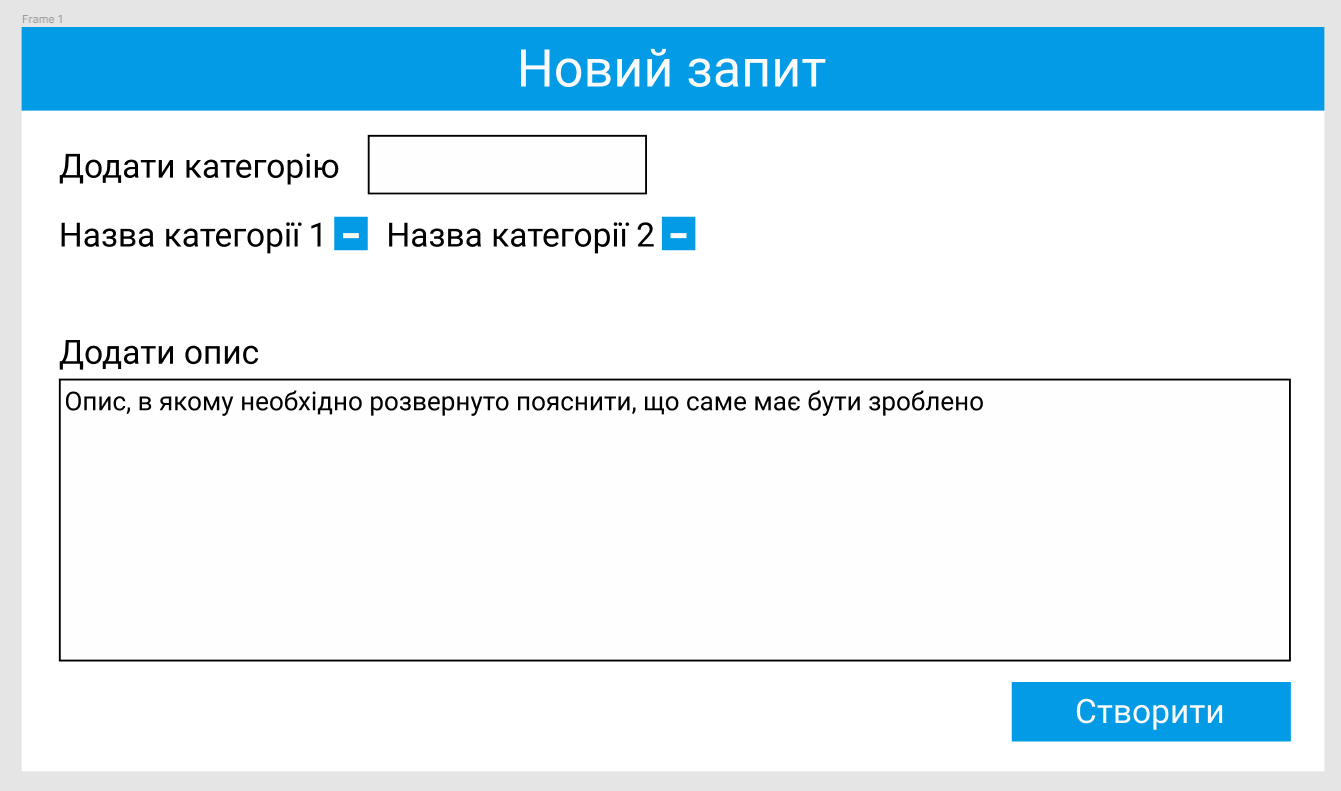


Рис 5.8 – Форма створення запиту до персоналу

## **5.4 Висновки до розділу**

В розділі описано інструментальний набір, за допомогою якого створено систему взаємодії з клієнтами, описано реалізацію взаємодії клієнтської та серверної частини системи, а також user-інтерфейс для різних користувачів.

# **6 ТЕСТУВАННЯ СИСТЕМИ ВЗАЄМОДІЇ З КЛІЄНТАМИ**

## **6.1 Тестування програмних модулів**

Самий простий та доступний метод тестування правильної роботи розробки та її програмних модулів – використання модульного або Unit-тестування [1]. Модульне тестування, як процес, дозволяє перевірити на коректність окремі модулі коду програми.

Ідея полягає в тому, щоб писати тести для кожної нетривіальної функції або методу. Це дозволяє досить швидко перевірити, чи не привела чергова зміна коду до регресії, тобто до появи помилок в протестованих місцях програми, а також полегшує виявлення і усунення таких помилок. При цьому встає декілька важливих питань:

Які функції можна вважати нетривіальними – всі ті функції, які є дуже чуттєвими до вхідних даних, або ті, логіка роботи яких є дуже комплексною.

Скільки тестів необхідно для правильної роботи розробки – стільки, скільки необхідно для покриття вимог, які передав замовник на початку процесу розробки.

Які результати вважати правильними – всі ті, які відповідають вимогам замовника, або, якщо замовник не зазначив, який він хоче отримувати результат – ті, які відповідають здоровому глузду.

Для Java існує багато фреймворків для модульного тестування, серед них одним з найчастіш використовуємих – JUnit. В рамках цього фреймворку ми маємо визначити методи, що анотовані як тест-функція. Після цього, в методі ми можемо проводити необхідні перевірки для цікавих для нас об’єктів. Наприклад, для перевірки роботи розрахунку вартості бронювання номеру Unit-тест виглядатиме так:

Код класу RoomOrderTest

**class** RoomOrderTest {  
  
 **private** RoomOrder **order1**;  
 **private** RoomOrder **order2**;  
  
 @BeforeEach  
 **public void** init() {  
 RoomCategory category = **new** RoomCategory();  
  
 RoomPrice price1 = **new** RoomPrice();  
 price1.setDate(**new** GregorianCalendar(2020, Calendar.***JUNE***, 1).getTime());  
 price1.setValue(1000.0);  
  
 RoomPrice price2 = **new** RoomPrice();  
 price2.setDate(**new** GregorianCalendar(2020, Calendar.***JULY***, 1).getTime());  
 price2.setValue(2000.0);  
  
 category.getPriceList().add(price1);  
 category.getPriceList().add(price2);  
  
 Room room = **new** Room();  
 room.setCategory(category);  
  
 **order1** = **new** RoomOrder();  
 **order1**.setRoom(room);  
 **order1**.setArrival(**new** GregorianCalendar(2020, Calendar.***JUNE***, 1).getTime());  
 **order1**.setDeparture(**new** GregorianCalendar(2020, Calendar.***JUNE***, 8).getTime());  
  
 **order2** = **new** RoomOrder();  
 **order2**.setRoom(room);  
 **order2**.setArrival(**new** GregorianCalendar(2020, Calendar.***JUNE***, 25).getTime());  
 **order2**.setDeparture(**new** GregorianCalendar(2020, Calendar.***JULY***, 5).getTime());  
 }  
  
 @AfterEach  
 **public void** close() {  
 **order1** = **null**;  
 **order2** = **null**;  
 }  
  
 @Test  
 **void** testPriceCalculation() {  
 Assertions.*assertEquals*(7000, **order1**.calculatePrice());  
 Assertions.*assertEquals*(14000, **order2**.calculatePrice());  
 }  
}

Для модульного тестування існує декілька параметрів, за якими можна визначити, чи був даний етап якісно і наскільки можна довіряти результатам цього виду тестування. Один з найбільш важливих з таких параметрів є відсоток покриття коду. Даний параметр визначає, яку обсяг створеного коду дійсно була використана для перевірки правильної роботи системи. Так, для обчислення значення відсотку покриття коду в інтегрованій системі розробки IntelliJ IDEA існує спеціальний плагін Coverage. Приклад роботи цього плагіну розглянуто на рис. 6.1 – 6.2.



Рисунок 6.1 – Ініціалізація роботи плагіну Coverage

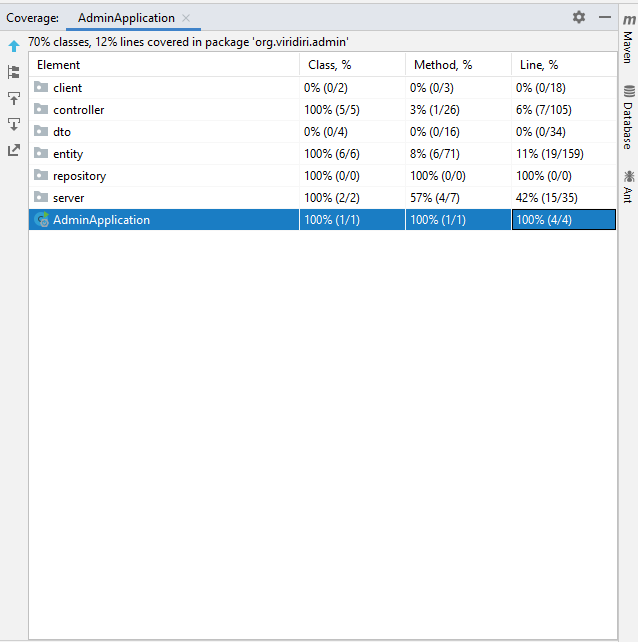


Рисунок 6.2 – Ініціалізація роботи плагіну Coverage

Після того, як користувач проведе усі можливі та необхідні дії у системі, та вимкне її робочий процес, плагін видає відсоток використаних класі та відсоток ліній використаного коду для обраного пакету, а також значення ті ж самих параметрів для усіх підлеглих пакетів та класів. Для нашої розробки кількість використовуємих класів дорівнює 100%, при цьому, значення покриття коду дорівнює 78%.

## **6.2 Юзабіліті тестування**

Те, що програмне забезпечення працює так, як зазначено у вимогах до розробки, не означає, що для кінцевого користувача не стане проблемою легкого та зрозумілого користування. Для перевірки цього існує методологія юзабіліті-тестування – перевірка програмного продукту на відповідність до вимог в плані зручності використання програми. Таким чином, за допомогою юзабіліті-тестування ми можемо визначити ергономічність (пристосованість до використання) програми.

Перевірка юзабіліті додатки полягає в: оцінюванні відповідності дизайну додатки до його функціональності, заданої замовником; аналізу використовуваних графічних елементів, колірного оформлення з точки зору сприйняття; оцінюванні зручності навігації; аналізу текстового наповнення сайту; оцінювання зручності використання сервісами ресурсу.

Так, для того, щоб зареєструватись у системі від користувача необхідно 10,5 секунд для заповнення двох полів для вводу (5 секунди на кожне), та натиск однієї кнопки (0,5 секунди).

Для створення замовлення на розміщення від користувача потребується заповнити 7 полів для вводу, причому два з них – вибір дати заїзду та виїзду відповідно. Час, що має бути затрачений на створення дорівнює 28 секундам.

## **6.3 Тестування сумісності**

Необхідно також враховувати, що сучасне програмне забезпечення дуже вимогливе в плані необхідних для правильної роботи ресурсів. Тому важливо відповісти на питання мінімального апаратного та програмного оточення, необхідного для коректного виконання програмної розробки.

Тестування сумісності – вид нефункціонального тестування, основною метою якого є перевірка коректної роботи продукту в певному оточенні. Оточення може включати в себе наступні елементи: апаратна платформа; операційна система (Unix, Windows, MacOS, ...); браузери (Internet Explorer, Firefox, Opera, Chrome, Safari).

Клієнтська частина нашого програмного забезпечення працює у вигляді веб-застосування, тому нам важливо розглянути, з якими браузерами воно сумісне (перевірка кроссбраузерності). Так, наша система підтримує роботу з Google Chrome версії 65; Mozilla Firefox версії 58; Internet Explorer версії не нижче 10 / Microsoft Edge версії не нижче 13.

При цьому, серверна частина застосування працює як звичайна програма, тому для неї важливим буде перевірити роботу в оточенні операційної системи. Так, для нашого застосування необхідна операційна система, базована на системі Unix, Java Runtime Environment версії не нижче 8.

**6.4 Проведення експерименту визначення ефективності використання системи**

Для того, щоб визначити, наскільки розроблена система відповідає описаним вимогам, ми проведемо декілька експериментів, в яких ми будемо порівнювати параметр часу обробки процесу бронювання номерів.

Для першого параметру ми встановимо декілька початкових значень: кількість вже існуючих записів на бронювання номерів та кількість майбутніх замовлень. Важливо зазначити, що для нашого експерименту нове замовлення потрапляє в обробку одразу після завершення минулого. Так, результати експерименту відображені у табл. 6.1.

Таблиця 6.1 – Результати експерименту

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Кількість оброблених замовлень | 0 | | 50 | | 100 | |
| Кількість нових замовлень | 25 | 50 | 25 | 50 | 25 | 50 |
| Ручна обробка, хв | 180 | 360 | 200 | 370 | 220 | 410 |
| Використання системи, хв | 130 | 250 | 135 | 260 | 140 | 275 |
| Відносна частка використання часу | 1.4 | 1.44 | 1.48 | 1.42 | 1.57 | 1.5 |

Розглянувши результати експерименту, можна побачити, що при використанні системи для обробки процесу бронювання номерів, в порівнянні з ручною обробкою, час обробки зменшився в 1,5 рази, що задовольняє висунутим вимогам до програмної системи.

## **6.4 Висновки до розділу**

В розділі розглянуто результати модульного, юзабіліті тестування, тестуваня сумісності, а також розглянуто результати експерименту, що підтверджує задовільнення мети розробки.

# **7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ**

## **7.1 Основні положення**

У дипломному проекті розглянуті дослідження з питань розробки системи взаємодії з клієнтами, а також взаємодія пристроїв розподіленої системи, вирішення яких здійснюються інженерами-програмістами, а саме їх здійснює розробники програмного забезпечення та системні адміністратори.

Отже, розгляуто питання охорони праці та безпеки на робочому місці, яке розташоване на території курортного комплексу «Лада», у будівлі адміністраторського комплексу, на першому поверху.

Працівник розробки програмного забезпечення, на обраному робочому місці інженера-програміста, виконує свою діяльність на курортному комплексу ТОВ «Лада», отже розглянемо організацію та управління охороною праці на підприємстві комплексу взагалі, та у підрозділі адміністрування більш детально.

Система управління охороною праці (далі – СУОП) — це сукупність органів управління підприємством, які на підставі нормативних документів проводять цілеспрямовану, планомірну діяльність з метою забезпечення здорових, безпечних і високопродуктивних умов праці. Створення СУОП здійснюється шляхом послідовного визначення мети і об'єкта управління, завдань і заходів щодо охорони праці, функцій і методів управління, побудови організаційної структури управління, складання нормативно-методичної документації. Мета СУОП – забезпечення безпечних умов праці, збереження життя та здоров’я людини.

Так, на підприємстві діють такі акти НПАОП – ДНАОП 0.00-4.12-99 “Типове положення про навчання з питань охорони праці, ДНАОП 0.00-1.29-97 «Правила захисту від статичної електрики», ДНАОП 4559-88 «Тимчасові санітарні норми та правила для працівників обчислювальних центрів».

Служба ОП – основа управління охороною праці, підпорядковується керівникові підприємства.  Вирішує завдання забезпечення виконання нормативів стосовно безпечної та здорової праці на підприємстві. Створюється на підприємстві не зважаючи на вид або рід діяльності підприємства. За своїми посадами керівництво служби прирівнюється до головних інженерів.

Інструктажі з ОП – на підприємстві існують багато видів інструктажів, наприклад вступний, первинний та повторний.

Медогляди – медичні огляди працівників проводяться регулярно з періодичністю на рік, та мають на мету визначення стану здоров’я працівника, його спроможність до праці, виявлення захворювань, несумісних з виконанням праці. При цьому, для кожного виду робіт буде свій набір вимог до стану здоров’я працівника, тому і процес медогляду може відрізнятися.

Засоби захисту – на курортному комплексі не відбувається роботи, яка потребувала б додаткових засобів захисту. В якості засобів колективного захисту використовуються вогнегасники, більш за все, порошкоподібні та газові.

## **7.2 Висновки до розділу**

В ході проведення роботи розглянуто питання організації та управління охороною праці в курортному комплексі «Лада», проаналізовано умови праці для інженера-програміста, небезпеки хімічного зараження місцевості та загрози від COVID-19, розраховані параметри безпеки при заземлені та параметри при хімічному зараженню місцевості, запропоновано заходи з охорони праці та безпеки під час загрози COVID-19.

# **ВИСНОВКИ**

В роботі проведений аналіз предметної області курортного комплексу, виділені основні її компоненти та зв’язки між ними. На основі аналізу побудовані вимоги до програмного забезпечення, а також складений план майбутніх робіт, що визначили необхідну для створення тривалість розробки – 3 місяці.

Розроблена децентралізована архітектура, в якій поєднано шаблони інтерактивних систем MVC та клієнт-сервер. Визначено структуру системи, що продемонстровано на діаграмах послідовності та класів, а також наведена структура баз даних. Визначено, які інструменти використовувались для виконання необхідних процесів, наприклад, використання фреймворку Java Spring MVC. Не етапі тестування визначено, що система відповідає заявленим вимогам, зокрема, за допомогою тестування сумісності та unit-тестування. В результаті отриманий продукт, що можна використовувати як система взаємодії з клієнтами.

# **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Каща А. Модульное тестирование: 2+2 = 4? URL: http://rsdn.org/article /testing/UnitTesting.xml Дата звертання - 21.09.2020

2. Беляков М. Микросервисы: как определить, подойдут ли они вашему проекту. URL: https://www.simbirsoft.com/blog/mikroservisy-nachalo-raboty/ Дата звертання - 02.10.2020

3. Бурлаку І. Aspects of the design of distributed databases. URL: http://www. scientificpapers.org/wp-content/files/1111\_Aspects\_of\_the\_design\_of\_distributed\_ databases.pdf – Дата звертання - 02.10.2020

4. Буч Г. The Unified Modeling Language User Guide. — 2-е. — М.: ДМК Пресс, 2006. — 496 с.

5. Грінберг, Пол. CRM at the speed of light. — СПб.: Символ Плюс, 2007. — 528 с.

6. Москалюк А. Методичні вказівки для оформлення розділу «Охорона праці та безпека у надзвичайних ситуаціях" в Дипломному проекті» студентами інституту "Комп’ютерних систем» — Одеса, ОНПУ.2018. —  21 с.

7. Ньюмен С. Building Microservices. — СПб.: Питер, 2016. – 304 с.

8. Ройс, У. Managing the Development of Large Software Systems. URL: https://web.archive.org/web/20160318002949/http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf – Дата звертання - 03.10.2020

9. Юзабилити – QALight. Центр подготовки IT специалистов. URL: https://qalight.com.ua /baza-znaniy/yuzabiliti/ – Дата звертання – 25.11.2020

10. Chameleon POS. URL: https://chm-s.com/ua/produkty/chameleon-pos/ – Дата звертання – 11.09.2020

11. EasyMS – https://easyms.co – Дата звертання – 11.09.2020

12. JetBrains IntelliJ IDEA. URL: https://www.jetbrains.com/ru-ru/idea/ – Дата звертання – 11.09.2020

13. OtelMS. URL: https://www.otelms.com. Дата звертання – 11.09.2020

14. Белл Д. UML basics: The sequence diagram. URL: https://developer.ibm.com /articles/the-sequence-diagram/ – Дата звертання – 04.11.2020